NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS
DEPARTMENT OF MATHEMATICS

PREDICTIVE MODELS FOR FOOTBALL MATCHES

MsC in "Statistics and Operational Research"

Author: Ioannis Vakolas

Supervisor: Dr. Fotios Siannis

Athens, 2022

Acknowledgements

I would like to express my immense gratitude to the leading supervisor, the professor Fotios Siannis, for his trust and his valuable assistance in carrying out this thesis. Additionally, I would like to thank the performance analyst of Asteras Tripolis, Thodoris Tsilimigkras. He contributed to the creation of the data frame for the application of the dissertation, by offering a huge data base which concerns the Greek Superleague. I would also like to express my deepest gratitude to my colleagues for the great collaboration we had with each other during my last academic experience. Finally, my gratefulness is extended to my family and friends for their support and confidence.

Abstract

Football is one of the most popular sports in the world. In recent years, more and more companies have been associated with football depending economically on it. This led to a huge statistical interest in the sport. This thesis constitutes a review on football modeling.

Initially, theory behind bivariate analysis is developed along with properties and extensions of the bivariate distribution. Special attention is paid to the bivariate Poisson distribution which is widely used in football modeling. Regression models constitute another subject of study as they provide functions that describe the relationship between random variables. In that part, count data models are presented such as Poisson regression model and the inflated models which deal with problems with excessive outcomes. As for the parameters estimation, the EM algorithm is considered to be a rational way to find the maximum likelihood estimate when the latter cannot be calculated in straightforward way.

After presenting the theoretical framework on with football modeling is based, several bivariate predictive models are presented in terms of four main categories: naïve models, models with dependence parameter, inflated models, dynamic models.

Finally, analysis of the Greek Superleague is carried out through four bivariate models. After the comparison of the models' fitting, prediction in a playoff match takes place.

Contents

Chapter 1

Bivariate Discrete Distribution

In this chapter, we will present the bivariate discrete distributions as well as their properties. We consider the joint distribution of two random discrete variables X and Y. They are assumed to have the probability mass function $f_{X,Y}(x,y)$ at the point (x,y) with $(x,y) \in T$, where T is a subset of the Cartesian product of the set of nonnegative integers on the real line. In this case the pair (X,Y) is said to have bivariate discrete distribution over T with the probability function $f_{X,Y}(x,y)$.

1.1. Joint distributions

Definition (Joint cumulative distribution function) *Let X and Y be two random variables defined on the same probability space $(\Omega, \mathcal{A}, P[.])$ where Ω is the set of all possible outcomes and $\mathcal{A}$ is a set of events. Then the (X,Y) is called a two-dimensional random variable. The joint cumulative distribution function or joint distribution function of X and Y, denoted by $F_{X,Y}(x,y)$, is defined as*

$$F_{X,Y}(x,y) = P[X \leq x, Y \leq y] \ , x, y \in \mathbb{R}$$

Properties:

1. *If $x_1 < x_2$ and $y_1 < y_2$ then*

$$P[x_1 < X < x_2 \, , y_1 < Y < y_2] =$$
$$= F(x_2,y_2) - F(x_2,y_1) - F(x_1,y_2) + F(x_1,y_1) \geq 0$$

2. ***(i)*** $F(-\infty, y) = \lim_{x \to -\infty} F(x, y) = 0 \quad \forall y \in \mathbb{R}$

(ii) $F(x, -\infty) = \lim_{y \to -\infty} F(x, y) = 0 \quad \forall x \in \mathbb{R}$

(iii) $F(\infty, \infty) = 1$

3. $F(x, y)$ *is right continuous for each argument*:

$$\lim_{h \to 0^+} F(x + h, y) = \lim_{h \to 0^+} F(x, y + h) = F(x, y)$$

Definition (Joint discrete density function) *Let X and Y two random discrete variables. The joint discrete density function of X and Y is defined as*

$$f_{X,Y}(x, y) = P[X = x, Y = y], \quad (x, y) \in T$$

where T is a subset of the Cartesian product of the set of the nonnegative integers on the real line.

1.2. Marginal distributions

When studying bivariate models, it may also be of interest to observe the behavior of the variables independently of each other. Taking the probability function of X and Y as $f_{X,Y}(x, y)$, the marginal probabilities for X and Y are respectively:

$$f_X(x) = \sum_y f_{X,Y}(x, y)$$

and

$$f_Y(y) = \sum_x f_{X,Y}(x, y)$$

It is remarkable that if X and Y are independent then,

$$f_{X,Y}(x,y) = f_X(x) \cdot f_Y(y) = P[X = x] \cdot P[Y = y]$$

Concerning the conditional discrete density functions, they are expressed as follows:

- $f_{Y|X}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)} = \frac{P[X=x,Y=y]}{P[X=x]} \quad if\ f_X(x) > 0$
- $f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} = \frac{P[X=x,Y=y]}{P[Y=y]} \quad if\ f_Y(y) > 0$

Definition (Marginal cumulative distribution function) *If $F_{X,Y}(x,y)$ is the joint cumulative distribution function of two random variables X and Y, then the $F_X(x,y)$ and $F_Y(x,y)$, which are called marginal distribution functions of X and Y respectively, are defined as*

$$F_X(x) = P[X \le x] = P[X \le x, Y < \infty] = \lim_{y \to \infty} F_{X,Y}(x,y) = F_{X,Y}(x,\infty)$$

and

$$F_Y(y) = P[Y \le y] = P[X < \infty, Y \le y] = \lim_{x \to \infty} F_{X,Y}(x,y) = F_{X,Y}(\infty,y)$$

1.3. Generating functions

When studying random variables, there is a variety of generating functions which helps us to point out the properties of the random variables. In this section we will introduce these functions.

1.3.1. Probability generating function

The probability generating function (PGF) $\Pi_{X,Y}(t_1, t_2)$ *of the pair of random variables* (X, Y) *with probability function* $f_{X,Y}(x, y)$ *is the* $\mathbb{E}[t_1^X t_2^Y]$*. So PGF is defined as :*

$$\Pi_{X,Y}(t_1, t_2) = \mathbb{E}[t_1^X t_2^Y] = \sum_{(x,y)\in T} t_1^x t_2^y \, f_{X,Y}(x, y)$$

The marginal PGF′s are

$$\Pi_X(t) = \sum_x f_X(x) t^x = \sum_x t^x \sum_y f_{X,Y}(x, y) = \Pi_{X,Y}(t, 1)$$

$$\Pi_Y(t) = \sum_y f_Y(y) t^y = \sum_y t^y \sum_x f_{X,Y}(x, y) = \Pi_{X,Y}(1, t)$$

1.3.2. Moment generating functions

The moment generating function (MGF) $M_{X,Y}(t_1, t_2)$ *of the pair of random variables* (X, Y) *with probability function* $f_{X,Y}(x, y)$ *is the* $\mathbb{E}[e^{t_1 X + t_2 Y}]$*. So MGF is defined us:*

$$M_{X,Y}(t_1, t_2) = \mathbb{E}[e^{t_1 X + t_2 Y}] = \sum_{(x,y)\in T} e^{t_1 x + t_2 y} \, f_{X,Y}(x, y)$$

By recalling the exponential series,

$$e^{tX} = 1 + tX + \frac{(tX)^2}{2!} + \frac{(tX)^3}{3!} + \cdots$$

in the univariate case we have:

$$M_X(t) = \sum_x e^{tX} f_X(x) = \sum_x (f_X(x) + tX f_X(x) + t^2 X^2 f_X(x) + \cdots) =$$

$$= 1 + \mu_1 t + \mu_2 \frac{t^2}{2!} + \mu_3 \frac{t^3}{3!} + \cdots$$

with $\mu_k = \mathbb{E}[X^k]\ k = 1,2,3 \dots$

So in the bivariate case MGF becomes:

$$M_{X,Y}(t_1,t_2) = \mathbb{E}[e^{t_1X+t_2Y}] = \sum_x \sum_y e^{t_1x+t_2y} f_{X,Y}(x,y) =$$

$$= \sum_x \sum_y \left(1 + tX + \frac{(tX)^2}{2!} + \cdots\right)\left(1 + tY + \frac{(tY)^2}{2!} + \cdots\right) f_{X,Y}(x,y)$$

$$= \sum_{r,s} \frac{t_1^r}{r!}\frac{t_2^s}{s!}\mu'_{r,s}$$

with the coefficients $\mu'_{r,s} = \mathbb{E}[X^r Y^s]$ *.*

The marginal MGF′s are

$$M_X(t) = \sum_x e^{tx} f_X(x) = \sum_x e^{tx} \sum_y f_{X,Y}(x,y) = M_{X,Y}(t,0)$$

$$M_Y(t) = \sum_y e^{ty} f_Y(y) = \sum_y e^{ty} \sum_x f_{X,Y}(x,y) = M_{X,Y}(0,t)$$

1.3.3. Cumulants generating functions

The cumulants generating function (CGF) $K(t_1,t_2)$ *of the pair of random variables* (X,Y) *with probability function* $f(x,y)$ *is the log of MGF. So CGF is defined as:*

$$K_{X,Y}(t_1,t_2) = log M_{X,Y}(t_1,t_2) = \sum_r \sum_s \frac{t_1^r}{r!}\frac{t_2^s}{s!} k_{r,s}$$

where $k_{r,s}$ *is called the cumulant of order* (r,s)*.*

1.4. Trivariate reduction

Suppose that we have X_1, X_2, X_3 which are three independent and maybe identically distributed random variables. We can construct the random variables X and Y as:

$$X = X_1 + X_3$$

$$Y = X_2 + X_3$$

Thus by using convolutions of three independent random variables, bivariate distributions can be generated, where a pair of observations from $f_{X,Y}(x,y)$ is obtained by

$$x = x_1 + x_3$$

$$y = x_2 + x_3$$

The method above is termed the trivariate reduction and it allows for dependence between the random variables of our study.

Now by taking under consideration the generating functions of X_i , $\quad i = 1,2,3$ the joint PGF and MGF of (X,Y) are respectively:

$$\Pi_{X,Y}(t_1,t_2) = \Pi_{X_1}(t_1)\Pi_{X_2}(t_2)\Pi_{X_3}(t_1 t_2)$$

and

$$M_{X,Y}(t_1,t_2) = M_{X_1}(t_1)M_{X_2}(t_2)M_{X_3}(t_1 + t_2)$$

Proof: *Let* $X = X_1 + X_3$ *and* $Y = X_2 + X_3$,

$$\begin{aligned}\Pi_{X,Y}(t_1,t_2) &= \mathbb{E}[t_1^X t_2^Y] = \mathbb{E}\left[t_1^{X_1+X_3} t_2^{X_2+X_3}\right] = \mathbb{E}\left[t_1^{X_1} t_1^{X_3} t_2^{X_2} t_2^{X_3}\right] \\ &= \mathbb{E}\left[t_1^{X_1} t_2^{X_2} (t_1 t_2)^{X_3}\right] = \Pi_{X_1}(t_1)\Pi_{X_2}(t_2)\Pi_{X_3}(t_1 t_2)\end{aligned}$$

$$\begin{aligned}M_{X,Y}(t_1,t_2) &= \mathbb{E}[e^{t_1 X + t_2 Y}] = \mathbb{E}\left[e^{t_1(X_1+X_3)+t_2(X_2+X_3)}\right] \\ &= \mathbb{E}\left[e^{t_1 X_1 + t_2 X_2 + (t_1+t_2)X_3}\right] = M_{X_1}(t_1)M_{X_2}(t_2)M_{X_3}(t_1 + t_2)\end{aligned}$$

1.5. The bivariate binomial distribution

It is widely known that the binomial distribution is the extension of the Bernoulli distribution and counts how many times an event X has occurred in a specific number of trials. Now we will examine the bivariate case of the binomial distribution. To start with, one bivariate Bernoulli trial measures two random variables (I, J), both with outcomes 0 and 1. As a result, each trial has four possible outcomes: $(0,0), (0,1), (1,0), (1,1)$. The probabilities of the outcomes are constant over the trials and the trials are independent. We define

$$p_{ab} = P(I = a, J = b) \qquad\qquad a = 0,1 \quad , b = 0,1$$

Similarly with the univariate case, considering a sequence of n bivariate Bernoulli trials leads to a bivariate binomial distribution. It is defined

$$X = \sum_{i=1}^{n} I_i$$

and

$$Y = \sum_{i=1}^{n} J_i$$

The pair (X, Y) is said to have bivariate binomial distribution.

The PGF of (X, Y) is:

$$\begin{aligned} \Pi_{X,Y}(t_1, t_2) &= \mathbb{E}[t_1^X t_2^Y] = \left\{\mathbb{E}\left[t_1^I t_2^J\right]\right\}^n \\ &= (p_{00} + t_1 p_{10} + t_2 p_{01} + t_1 t_2 p_{11})^n \end{aligned}$$

So, the marginal PGF's are respectively

$$\Pi_X(t) = \Pi_{X,Y}(t, 1) = \{(p_{11} + p_{10})t + (p_{01} + p_{00})\}^n$$

and

$$\Pi_Y(t) = \Pi_{X,Y}(1, t) = \{(p_{11} + p_{01})t + (p_{10} + p_{00})\}^n$$

Reminding that the PGF of the binomial distribution with parameters (n,p) is

$$\Pi_X(t) = (pt + q)^n \; for \; all \; t \in \mathbb{R}$$

we notice that,

$$X \sim Bin(n, p_{11} + p_{10})$$

$$Y \sim Bin(n, p_{11} + p_{01})$$

The bivariate binomial distribution is just an extension of the binomial distribution. In the univariate case we are counting the successes of a fact whereas in the bivariate case we are interested in how many times the events X and Y have occurred.

1.6. The bivariate Poisson distribution

The bivariate Poisson distribution can be defined by taking the limit $(n \to \infty)$ of the bivariate binomial distribution which has PGF

$$\begin{aligned}\Pi_{X,Y}(t_1, t_2) &= (p_{00} + t_1 p_{10} + t_2 p_{01} + t_1 t_2 p_{11})^n \\ &= \{(1 + (p_{11} + p_{10})(t_1 - 1) + (p_{11} + p_{01})(t_2 - 1) \\ &+ p_{11}(t_1 - 1)(t_2 - 1)\}^n\end{aligned}$$

We assume that

$$p_{11} + p_{10} = \frac{\lambda_1}{n}$$

$$p_{11} + p_{01} = \frac{\lambda_2}{n}$$

$$p_{11} = \frac{\lambda_3}{n}$$

where λ_1, λ_2 and λ_3 are positive constants independent of n.

Now, by substituting into the equation of the PGF of the bivariate binomial distribution it is:

$$\Pi_n(t_1,t_2) = \left(1 + \frac{\lambda_1(t_1-1)}{n} + \frac{\lambda_2(t_2-1)}{n} + \frac{\lambda_3(t_1-1)(t_2-1)}{n}\right)^n.$$

Taking into consideration the widely known limit $\lim_{n\to\infty}\left(1+\frac{\lambda}{n}\right)^n = e^{\lambda}$ it is :

$$\lim_{n\to\infty} \Pi_n(t_1,t_2) = \exp\{\lambda_1(t_1-1) + \lambda_2(t_2-1) + \lambda_3\,(t_1-1)(t_2-1)\}$$

So we have

$$\Pi_{X,Y}(t_1,t_2) = \exp\{\lambda_1(t_1-1) + \lambda_2(t_2-1) + \lambda_3(t_1-1)(t_2-1)\}$$

If we set $\lambda_1 = \lambda_1 + \lambda_3$ and $\lambda_2 = \lambda_2 + \lambda_3$ the equation above becomes:

$$\Pi_{X,Y}(t_1,t_2) = \exp\{\lambda_1(t_1-1) + \lambda_2(t_2-1) + \lambda_3(t_1t_2-1)\}$$

Looking at the PGF of the univariate Poisson distribution which is given by $\Pi_X(t) = \exp(\lambda(t-1))$, it is noticeable that this is the PGF of the bivariate Poisson distribution with parameters λ_1, λ_2 and λ_3 for two random variables X and Y.

Probability function

By expanding the joint PGF above we have,

$$\begin{aligned}\Pi_{X,Y}(t_1,t_2) &= \exp\big(\lambda_1(t_1-1) + \lambda_2(t_2-1) + \lambda_3(t_1t_2-1)\big)\\ &= e^{-(\lambda_1+\lambda_2+\lambda_3)} \sum_{i=0}^{\infty}\frac{\lambda_1^i t_1^i}{i!}\sum_{j=0}^{\infty}\frac{\lambda_2^j t_2^j}{j!}\sum_{k=0}^{\infty}\frac{\lambda_3^k t_1^k t_2^k}{k!}\\ &= e^{-(\lambda_1+\lambda_2+\lambda_3)} \sum_{r,s}\sum_{i}\frac{\lambda_1^{r-i}\lambda_2^{s-i}\lambda_3^{i}}{(r-i)!\,(s-i)!\,i!}\; t_1^r t_2^s\end{aligned}$$

As a result, we end up with the mass function,

$$f_{X,Y}(x,y) = e^{-(\lambda_1+\lambda_2+\lambda_3)} \cdot \frac{\lambda_1^x}{x!} \cdot \frac{\lambda_2^y}{y!} \cdot \sum_{k=0}^{\min(x,y)} \binom{x}{k}\binom{y}{k} k! \left(\frac{\lambda_3}{\lambda_1\lambda_2}\right)^k$$

which is the density of the bivariate Poisson distribution $BP(\lambda_1, \lambda_2, \lambda_3)$.

Marginal distributions

The marginal PGF of X is

$$\Pi_X(t) = \Pi_{X,Y}(t,1) = \exp\{(\lambda_1 + \lambda_3)(t-1)\}$$

and the marginal PGF of Y is

$$\Pi_Y(t) = \Pi_{X,Y}(1,t) = \exp\{(\lambda_2 + \lambda_3)(t-1)\}$$

So, respectively

$$X \sim Poisson(\lambda_1 + \lambda_3)$$

$$Y \sim Poisson(\lambda_2 + \lambda_3)$$

1.7. Bivariate correlation

In bivariate analysis, two variables that follow a joint distribution usually interact with each other. This can be described by the correlation coefficient which measures the strength of association between the two variables X, Y and describe the type of their relationship. This coefficient takes values in the interval $[-1,1]$. If the coefficient takes the value $+1$ or the value -1 then there will be a perfect degree of association between the variables whereas when the coefficient takes the value 0, it implies no dependence between the two variables. The sign indicates the direction of the relationship. If we have sign $+$ then there will be positive relationship and if we have sign – then there will be negative relationship between the variables. Two basic types of correlation are Pearson correlation and Kendall correlation each of which adjusts to different occasions.

1.7.1. Pearson correlation coefficient

In statistics, the Pearson correlation coefficient, also known as Pearson's r $(or\ \rho)$, is a measure of linear correlation between two sets of data. It is retrieved when the covariance of two variables X, Y is divided with the product of their standard deviations. That is,

$$r_{X,Y} = \frac{COV(X,Y)}{\sigma_X \sigma_Y} = \frac{n \sum^n x_i y_i - \sum^n x_i \sum^n y_i}{\sqrt{n \sum^n x_i^2 - (\sum^n x_i)^2} \sqrt{n \sum^n y_i^2 - (\sum^n y_i)^2}}.$$

It is essentially a normalized measurement of the covariance.

1.7.2. Kendall rank correlation coefficient

In statistics, the Kendall rank correlation coefficient, also known as Kendall's τ, is a measure of the ordinal association between two quantities. Ordinal data is a statistical data type where the variables have natural, ordered categories and the distances between these categories are unknown.

Let $(\mathrm{x_1}, \mathrm{y_1}), \dots, (\mathrm{x_n}, \mathrm{y_n})$ be a set of observations of the joint random variables X, Y, such that all the value of $\mathrm{x_i}$ and $\mathrm{y_i}$,$\mathrm{i} = 1, \dots, \mathrm{n}$ are unique. Any pair of the observations $(\mathrm{x_i}, \mathrm{y_i})$ and $(\mathrm{x_j}, \mathrm{y_j})$, where $\mathrm{i} < j$, will be said to be concordant if the sort order of $(\mathrm{x_i}, \mathrm{x_j})$ and $(\mathrm{y_i}, \mathrm{y_j})$ is the same. That is, when both $\mathrm{x_i} > \mathrm{x_j}$ and $\mathrm{y_i} > \mathrm{y_j}$ happen or both $\mathrm{x_i} < \mathrm{x_j}$ and $\mathrm{y_i} < \mathrm{y_j}$ happen. On the other hand, if the sort order is opposite the observations will be said to be discordant. In the specific case where $x_i = x_j$ or $y_i = y_j$, then the pair of observations are said to be tied.

Definition (Kendall's τ coefficient) *Let us denote n_c the number of concordant pairs and n_d the number of discordant pairs of n observations of the pair (X, Y) of the random variables X, Y. The Kendall's τ coefficient is defined as*

$$\tau = \frac{n_c - n_d}{\binom{n}{2}}$$

where $\binom{n}{2} = \frac{n(n-1)}{2}$ is the number of pairings between X, Y.

It is reasonable that if all the pairings between X and Y are concordant then the τ coefficient will be equal to 1. On the other side, if all the pairings between X and Y are discordant then the value of τ will be equal to -1.

Actually, the total number of pairings between X and Y is equal to $n_c + n_d + n_0 = \binom{n}{2}$ where n_c, n_d is the numbers of the concordant and the discordant pairs respectively, and n_0 is the number of tied pairs. However, as we can distinguish in the definition above, the tied pairs are not taken into consideration for the calculation of Kendall's τ coefficient.

1.8. Bivariate Copulas

When we have two dependent on each other discrete random variables, we can find their joint cumulative distribution function by using a two-dimensional copula. Copulas are linking functions which link univariate marginal distributions together allowing for dependence between the random variables with a dependence parameter θ. These functions enable us to isolate the dependency structure in a multivariate distribution. So it is easy for us to separate the marginal distributions from the dependence structure of a given multivariate distribution.

1.8.1. Copula

Definition (Copula) *A d-dimensional copula,* $C\colon [0,1]^d \to [0,1]$ *is a cumulative distribution function (CDF) with uniform marginals. For a generic copula we write*

$$C(u) = C(u_1, \dots, u_d) = P(U_1 \le u_1, \dots, U_d \le u_d).$$

Properties:

1. $C(u_1, \dots, u_d)$ *is non-decreasing for each component* u_i*.*

2. The marginal distribution of the i^{th} *component is obtained by setting* $u_k = 1$ *for* $k \neq i$ *in* $C(u)$*.*

3.$C(u_1, \dots, u_{i-1}, 0, u_{i+1}, \dots, u_d) = 0$ *if any one of the components is* 0.

We now recall the definition of generalized inverse for a CDF, F.

Definition (generalized inverse) *Let* F *a cumulative distribution function (CDF). Then, the generalized inverse* F^{-1}*, is defined as*

$$F^{-1}(x) \coloneqq \inf\{u : F(u) \ge x\}.$$

Proposition *If* $U \sim U[0,1]$*and* F_X *is a CDF, then*

$$P(F^{-1}(U) \le x) = F_X(x)$$

In the case of a continuous CDF, then $F_X(X) \sim U[0,1]$

Theorem (Sklar's Theorem) *Consider a d-dimensional CDF, F, with marginals $F_1, \dots, F_d$. Then there exists a copula C, such that*

$$F(x_1, \dots, x_d) = C\big(F_1(x_1), \dots, F_d(x_d)\big)$$

for all $x_i \in [-\infty, +\infty]$ and $i = 1, \dots, d$.

In the bivariate case, a copula function can be expressed as follows:

$$C(u_1, u_2|\theta) = P(U_1 \le u_1, U_2 \le u_2)$$

In this expression, we have two independent and identically distributed standard uniform variables U_1, U_2 and θ is a dependence parameter.

Let X_i with a continuous CDF F_i, then the transform $F_i(X_i)$ must be uniformly distributed. As a result the joint bivariate CDF with marginal CDF's F_1 and F_2 can be written as follows:

$$\begin{aligned} F(x_1, x_2) &= P(X_1 \le x_1, X_2 \le x_2) \\ &= P\big(F_1(X_1) \le F_1(x_1), F_2(X_2) \le F_2(x_2)\big) \\ &= P\big(U_1 \le F_1(x_1), U_2 \le F_2(x_2)\big) \\ &= C(F_1(x_1), F_2(x_2)|\theta) \end{aligned}$$

1.8.2. Types of bivariate discrete copulas

Now we will assume that X_i has a discrete CDF and not a continuous one like the occasion above. In the case of discrete distributions like the Poisson or the negative binomial distribution, the marginal cumulative distribution functions are step functions with jumps at integer values. This results to not having unique F_i^{-1}. For that cases there are several types of copula which have different domains of the dependence parameter θ:

- *Frank Copula*

A basic type of copula for discrete occasions is the Frank copula type where $\theta \in (-\infty, +\infty) - \{0\} = \mathbb{R} - \{0\}$. The Frank copula is expressed as follows:

$$C\big(F_X(x), F_Y(y)\big) = \frac{1}{\theta}\log\left(1 + \frac{((exp(\theta F_X(x)) - 1)(exp(\theta F_Y(y)) - 1)}{exp(\theta) - 1}\right)$$

where F_X, F_Y are the marginal discrete cumulative distribution functions.

- *Gumbel Copula*

A second type of copula is the Gumbel copula where $\theta \in [1, +\infty)$. It is expressed as follows:

$$C\big(F_X(x), F_Y(y)\big) = e^{\left\{\left[-\log\left(F_X(x)\right)\right]^{\theta} + \left[-\log\left(F_Y(y)\right)\right]^{\theta}\right\}^{\frac{1}{\theta}}}$$

- *Joe Copula*

Joe copula is also a type of copula with $\theta \in [1, +\infty)$ and which is expressed as :

$$C\big(F_X(x), F_Y(y)\big) = 1 - \left[\big(1 - F_X(x)\big)^{\theta} + \big(1 - F_Y(y)\big)^{\theta} - \big(1 - F_X(x)\big)^{\theta}\big(1 - F_Y(y)\big)^{\theta}\right]^{\frac{1}{\theta}}$$

- *Clayton Copula*

Another type of copula is Clayton copula, where $\theta \in (0, +\infty)$ and it is expressed as follows:

$$C\big(F_X(x), F_Y(y)\big) = [\big(F_X(x)\big)^{-\theta} + \big(F_Y(y)\big)^{-\theta} - 1]^{-\frac{1}{\theta}}$$

The types of copulas that were mentioned, are some basic bivariate copulas. There are also other types of bivariate copulas such as the *Normal copula*, Student's copula etc.

The proper choice of copula depends a lot on the domain of θ which is connected with the type of the dependence that our variables have each other.

Chapter 2

Regression Models

As it is known, the components of the regression models with i observations are: the dependent variable which is observed and denoted as the observation Y_i, the independent variables which are also observed and denoted as the vector X_i, the unknown parameters (coefficients) which are often denoted as the vector $\boldsymbol{\beta}$ and the error terms ε_i. The general form of a regression model is:

$$Y_i = f(X_i, \beta) + \varepsilon_i$$

The aim of the researchers is to choose the function f that closely fits the data. Several choices of the function f lead to different types of regression.

2.1. Generalized linear models (GLM)

2.1.1. Structure

The basic regression model is the linear regression model which is based on the normal probability function and is expressed as

$$Y = \boldsymbol{\beta}_0 + \boldsymbol{\beta} X + \varepsilon$$

However, linearity cannot deal with a variety of practical situations such as counts (they will be explained in the next paragraph).

The generalized linear model (GLM) is a generalization of the ordinary linear model as it extends the concept of the linear regression model. It generalizes the linear regression by allowing the linear model to be related to the response variable via a link function.

Definition (Link Function) *We assume the regression model with* i *observations. For the* $i-th$ *observation, let* $y_i = f(x_i, \boldsymbol{\beta})$, *where* $x_i^T = (x_{i1}, \dots, x_{ip})$ *is a vector of* p *explanatory variables and* $\boldsymbol{\beta^T} = (\boldsymbol{\beta_1}, \dots, \boldsymbol{\beta_p})$ *is a vector of coefficients. Additionally let* g *be a differentiable function of* $f(x_i, \boldsymbol{\beta})$ *such that* $g(f(x_i, \boldsymbol{\beta}) = x_i^T \boldsymbol{\beta}$. *Then the function* g *is called link function.*

2.1.2. Deviance goodness-of-fit

When a Generalized Linear Model (GLM) is fitted, then a deviance goodness-of-fit test is used to show the explanatory power of the model. In this procedure, the actual model is compared with the saturated model. The saturated model has achieved a perfect fit as the number of the parameters is equal to the number of observations. However, the saturated model isn't actually an excellent choice as it doesn't smooth the data. As a result, a simpler model which uses only a few predictors may have more advantages. Nevertheless, the saturated model is useful for testing the fit of other models. So by denoting as $L(\hat{\lambda}; \boldsymbol{y})$ the maximized log-likelihood for the model being tested and as $L(\boldsymbol{y}; \boldsymbol{y})$ the maximized log-likelihood in the saturated case, we have the following test statistic:

$$D(\boldsymbol{y}; \hat{\lambda}) = -2\left[L(\hat{\lambda}; \boldsymbol{y}) - L(\boldsymbol{y}; \boldsymbol{y})\right]$$

where $\hat{\lambda}$ is a vector of predictors of the observation **y**.
The expression $D(\boldsymbol{y}; \hat{\lambda})$ is called deviance and we have that $D(\boldsymbol{y}; \hat{\lambda}) \sim X_{n-p}^2$ where n is the number of parameters in the saturated model and p is the number of parameters in the model being tested. If the deviance is small then the model will be a good fit for the data. This occurs because the observed values are close to the predicted ones given by the model.

2.1.3. Over-dispersion in GLM

We consider n-dimensional vector of observations $Y = (Y_1, \dots, Y_n)$ and a theoretical model that describes Y. Over-dispersion occurs when the observed variance of the data is higher than it would be expected. In other words, it occurs when the variance of the observations is greater the variance of theoretical model. Some distributions do not have a specific parameter to fit the variation of the observations. A typical example is the Poisson distribution where the mean is described equally to the variance by a parameter λ. In, this case, for an expected value of Y, $\mathbb{E}[Y] = 10$, we expect that the variance of the observed data points is also 10. In contrast, the Normal distribution describes separately the variance through the parameter σ^2.

Let us give an example of over-dispersion. Imagine the number of seedlings in a forest plot. Depending on the distance to the source tree, there may be many hundreds or none. Such data would be over-dispersed for a Poisson distribution.

In statistics, dispersion parameter φ is a parameter which is associated to whether the observed variance of the data is greater than the variance of the theoretical model or not (over-dispersion or under-dispersion).

If the distribution of a variable Y belongs to the exponential family, then its density function can be written as,

$$f(y; \theta, \varphi) = \exp\left(\frac{y\theta - b(\theta)}{a(\varphi)} + c(y, \varphi)\right)$$

where θ is the parameter of interest and φ is the dispersion parameter. In this form the expected value and the variance of Y are expressed,

$$\mathbb{E}[Y] = b'(\theta)$$

$$Var[Y] = b''(\theta)a(\varphi)$$

For instance, in the case of the exponential family form of the Normal distribution we have:

$$f(y;\mu,\sigma^2) = \exp\left\{-\frac{y\mu - \frac{1}{2}\mu^2}{\sigma^2} + \left(-\frac{y^2}{2\sigma^2} + \log\left(\sigma\sqrt{2\pi}\right)\right)\right\}$$

where $\theta = \mu,\ b(\theta) = \frac{1}{2}\mu^2,\ a(\varphi) = \sigma^2,\ \mathbb{E}[Y] = \mu,\ Var[Y] = \sigma^2$

In order to assess whether there is over-dispersion in a model or not, we can evaluate the ratio of the residual deviance divided by the degrees of freedom so that φ is estimated,

$$\hat{\varphi} = \frac{Residual\ deviance}{Degrees\ of\ freedom} = \frac{D\left(\boldsymbol{y},\hat{\lambda}\right)}{n-p}$$

where $n-p$ is the difference between the number of the parameters of the saturated model and the model being tested.
In a Poisson GLM, the estimated variance can be expressed as $Var[Y] = \varphi\mathbb{E}[Y]$. So, the Poisson assumption indicates $\varphi = 1$ which yields that the variance is equal to the expectation. If $\hat{\varphi} > 1$ there is over-dispersion in the model, and if $\hat{\varphi} < 1$, there is under-estimation.
So, it is remarkable that if $\hat{\varphi} > 1$, the Poisson assumption is not correct.

2.2. Count data models

When discussing about modeling count data, it's important to clarify the meaning of count data. Generally, count data refer to observations made about events or items that are enumerated. In statistics, count data refer to observations that have only nonnegative integer values ranging from zero to some undetermined value. Theoretically, counts can range from zero to infinity. However, they are always limited to a distinct maximum value. There are many count data examples such as the number of children that a couple has, the number of someone's doctor visits, the number of goals achieved by a football team etc.

2.2.1. Poisson regression

Poisson regression model is the basic model which a variety of count models are based on. It is derived by the Poisson probability mass function, which can be expressed as

$$f(y_i;\lambda_i) = \frac{e^{-\lambda_i t_i}(\lambda_i t_i)^{y_i}}{y_i!} \quad ,y_i = 0,1,2,\ldots$$

where y_i is the i-th observation-count response, λ_i is the mean number of events in a time period of length t_i. When λ_i is understood as applying to individual counts without consideration of size or time, then $t_i = 1$. The mean number of the evens λ_i is modeled as follows:

$$\lambda_i = t_i f(x_i, \boldsymbol{\beta}) \quad ,i = 1,\ldots,n$$

where $x_i^{\boldsymbol{T}} = (x_{i1},\ldots,x_{ip})$ is a vector o p explanatory variables, $\boldsymbol{\beta^T} = (\beta_1,\ldots,\beta_p)$ is a vector of coefficients and f is the rate function.

The distributions of the exponential family have corresponding link functions that are called *canonical links.* In the case of Poisson regression, we have a log-link function. Moreover, since the Poisson distribution values are nonnegative, using a link function whose inverse function takes only nonnegative numbers is purposeful.

By using the log-link function, we have,

$$log(\lambda_i) = log(t_i) + x_i^T \beta \,, \qquad i = 1, \dots, n$$

where the $\log(t_i)$ can be transferred to the left side of the equation above. This will finally lead to the consideration of the $\log\left(\frac{\lambda_i}{t_i}\right)$ as the response variable.

2.2.2. Inflated Models

Many times, when modeling the outcomes of a variable we notice underestimation over a specific outcome. Quite often, this specific outcome is zero. Count data with many zeros are common in a wide variety of experiments. In order to manage this occurrence, it is often useful to use a mixture of models in order to correct this underestimation. A specific kind of mixture distribution is the inflated model, which inflates the probability of this underestimated outcome in our study.
Random variables are usually considered as a sample from a distribution. However, there are random variables that cannot be described from one single distribution alone. Most of real-life random variables are generated from a mixture of distributions.

Definition (Mixture Distribution) *Let us consider k distributions $\{g_1(x;\theta_1), \dots, g_k(x;\theta_k)\}$ and k coefficients $\{w_1, \dots, w_k\}$. Then the mixture distribution f of the densities g_i with the weights w_i for $i = 1, \dots, k$ is defined as:*

$$f(x;\theta_1, \dots, \theta_k) = \sum_{i=1}^{k} w_i g_i(x;\theta_i),$$

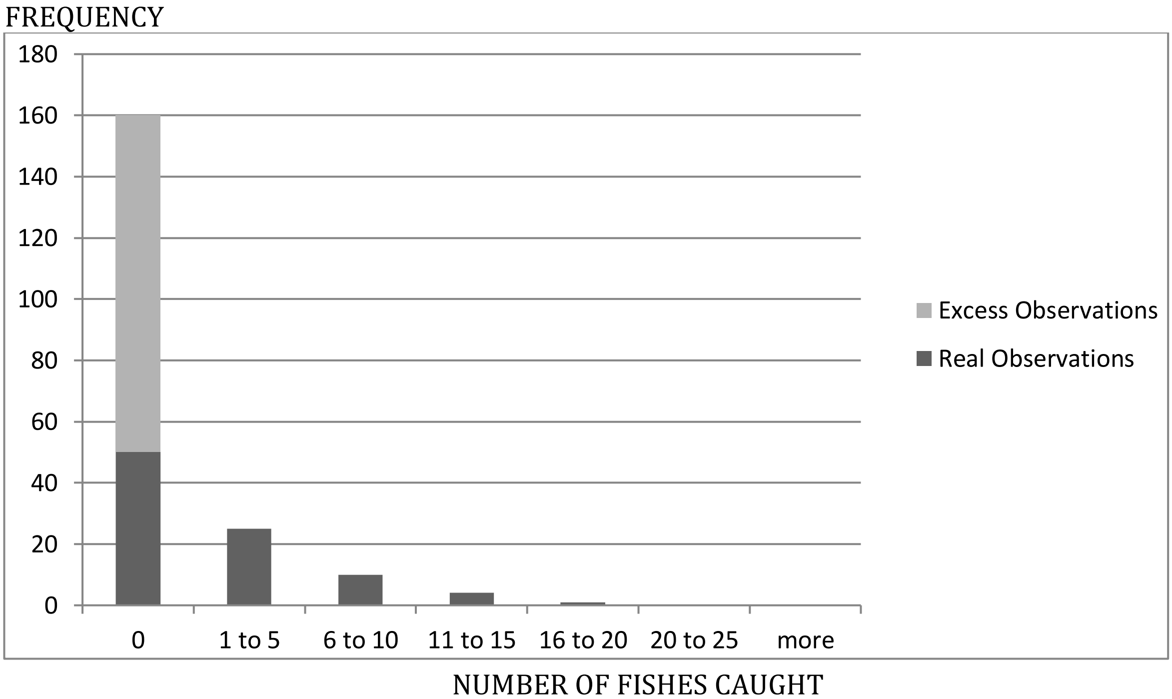
subject to $\sum_{i=1}^{k} w_i = 1$.

The densities g_i from the definition above are not necessarily from the same family. However, this makes the problem sometimes complex.

Zero-Inflated models

In many real life statistical experiments we often observe many zeros. This is something that cannot be modeled using standard modeling approaches for count data. Let us give a simple example:

We consider 200 people in a large boat and we want to see their success in fishing. We take observations about how many fishes each one caught and so we have the following graph of frequency:



In the graph above we can distinguish a large amount of zeros in which some are real and some excess. Real zeros are connected with people who fish but did not manage to catch any fish. Excess zeros are associated with people that may not even fish, for instance some women or little children. However, all 200 people of this boat are included in our study so it is necessary to deal with this.

Zero-inflated models take into account excess zeros data. They estimate two equations: a count model and a model for the excess number of zeros.

Definition (Zero-Inflated Model) *Assume the state X_0 which is 0 with probability 1 and the state X_1 which is a random variable taking nonnegative integers with probability function $P(X_1 = x) = g(x, \lambda)$ for $x = 0,1, \ldots$, where $\lambda = (\lambda_1, \ldots, \lambda_s)^T$ is an unknown parameter vector in an open subset D of s-dimensional space $\mathbb{R}^s$. Now consider the mixture of X_0 and X_1 with the Bernoulli(p) where $0 \leq p < 1$. Zero-inflated model is defined as*

$$f_{ZIM}(x, \boldsymbol{\theta}) = \begin{cases} p + (1-p)g(0, \lambda), & for\ x = 0 \\ (1-p)g(x, \lambda), & for\ x = 1,2, \ldots \end{cases}$$

where $\boldsymbol{\theta} = \binom{p}{\lambda} \in \boldsymbol{\Theta} = (0,1] \times D$. The mixture above is denoted as $X \sim ZIM(\boldsymbol{\theta}, g)$ or simply $X \sim ZIM(\boldsymbol{\theta})$.

The mean of the zero-inflated count data model is:

$$\mathbb{E}(X) = \sum_{k=0}^{+\infty} (1-p)g(k, \lambda) = (1-p)\mathbb{E}_g(X)$$

where $\mathbb{E}_g(X)$ denotes the mean of g.

A common type of zero-inflated model is the Poisson zero-inflated regression model.

- **Zero-Inflated Poisson regression**

When the Poisson regression model is applied to the count outcome data in real world, it is not rare to see the poor model fit indicated by a deviance. Most of the real data violate the assumption of the standard Poisson model, which is called equidispersion (the variance of the count outcome is equal to the mean). In most of the real data over-dispersion is observed (Sun Y. Jeon 2013). Ignoring over-dispersion and applying the standard Poisson regression for this data can cause underestimation of standard errors and p-values.

The zero-inflated Poisson (ZIP) is an alternative that can be considered in this case. This model allows for over-dispersion assuming that there are two types of individuals in the data (Sun Y. Jeon 2013):

1) those who have a zero count with probability of 1 (“always 0 group”)
2) those who have counts predicted by the standard Poisson. (“not always 0 group”)

Observed zero could be either from the zero count or the standard Poisson.

The observation i is in “always 0 group” with probability p_i and the latter can be predicted by a logit or probit model (these models will be presented in the paragraph 2.3.). The probability that observation i is in “not always 0 group” becomes $1 - p_i$. For observations in the second group, their positive count outcome is predicted by the standard Poisson (λ_i). The overall model is a mixture of the probabilities from the two groups above. As a result, for the i-th observation:

$$f_{ZIP}(y_i) = \begin{cases} p_i + (1 - p_i)e^{-\lambda_i}, & if\ y_i = 0 \\ (1 - p_i)\dfrac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}, & if\ y_i > 0 \end{cases}$$

where f_{ZIP} the density of the zero-inflated Poisson model.

The mean and the variance of the model above are,

$$\mathbb{E}[Y_i] = 0 \cdot p_i + \lambda_i \cdot (1 - p_i) = \lambda_i \cdot (1 - p_i)$$

and

$$Var[Y_i] = \lambda_i(1 - p_i)(1 + p_i\lambda_i)$$

respectively.

2.3. Logit and probit models

The logistic models (logit models) and the probit models are the statistical models that model the probability μ (expected value) of one event taking place out of two alternatives. They are among the most widely used members of the family of GLM models in the case of binary dependent variables.
Let $\eta = x\boldsymbol{\beta}$ a linear model where η is a response variable, x is vector of explanatory variables and $\boldsymbol{\beta}$ is a vector of coefficients.

In the ***logit models*** the link function relating the linear predictor $\eta = x\boldsymbol{\beta}$ to the expected value μ is the logit transform,

$$\log\left(\frac{\mu}{1-\mu}\right) = \eta = x\boldsymbol{\beta}$$

Solving μ in the equation above results to the logistic function,

$$\mu(x) = \frac{e^{x\boldsymbol{\beta}}}{1+e^{x\boldsymbol{\beta}}} = \frac{1}{1+e^{-x\boldsymbol{\beta}}}$$

In the ***probit models*** the link function that relates the linear predictor $\eta = x\boldsymbol{\beta}$ to the expected value μ is the inverse normal cumulative distribution function,

$$\Phi^{-1}(\mu) = \eta = x\boldsymbol{\beta}$$

Suppose a response variable Y is binary (1 or 0) and we consider a vector of regressors X that influence the outcome Y. The model takes the form,

$$P[Y = 1|X] = \Phi(X^T\boldsymbol{\beta})$$

where Φ is the cumulative distribution function of the standard normal distribution.
Considering a latent variable $Y^* = X^T\boldsymbol{\beta} + \varepsilon$ where $\varepsilon \sim N(0,1)$, the probit model above may transformed to the model,

$$Y = \begin{cases} 1, & Y^* > 0 \\ 0, & otherwise \end{cases}$$

As a result,

$$\begin{aligned} P[Y = 1|X] &= P[Y^* > 0] \\ &= P[X^T\boldsymbol{\beta} + \varepsilon > 0] \\ &= P[\varepsilon > -X^T\boldsymbol{\beta}] \\ &= P[\varepsilon < X^T\boldsymbol{\beta}] \\ &= \Phi(X^T\boldsymbol{\beta}) \end{aligned}$$

2.4. Ordinal regression models

In statistics, ordinal regression, also called ordinal classification, is a type of regression analysis used for the prediction of an ordinal variable. The value of an ordinal variable exists on an arbitrary scale where only the relative ordering between different values is significant. A typical example of ordinal regression is ordered probit.

Ordered Probit Model

Let Y_i be individual i's response variable and assume that this can take an integer value on the set $[0, J]$. Let y_i^* be the underlying latent variable representing i's tendency to agree with the statement advanced. The ordered probit model is based on the assumption that y_i^* depends linearly on x_i:

$$y_i^* = x_i\beta + e_i\,, \qquad i = 1, \dots, n$$

where $e_i \sim N(0,1)$ and β is a vector coefficients not containing an intercept.

The relationship between y^* and the observed variable Y is expressed as follows:

$$Y = 1 \quad if \; -\infty < y^* < \kappa_1$$

$$Y = 2 \quad if \quad \kappa_1 < y^* < \kappa_2$$

$$Y = 3 \quad if \quad \kappa_2 < y^* < \kappa_3$$

$$.$$

$$.$$

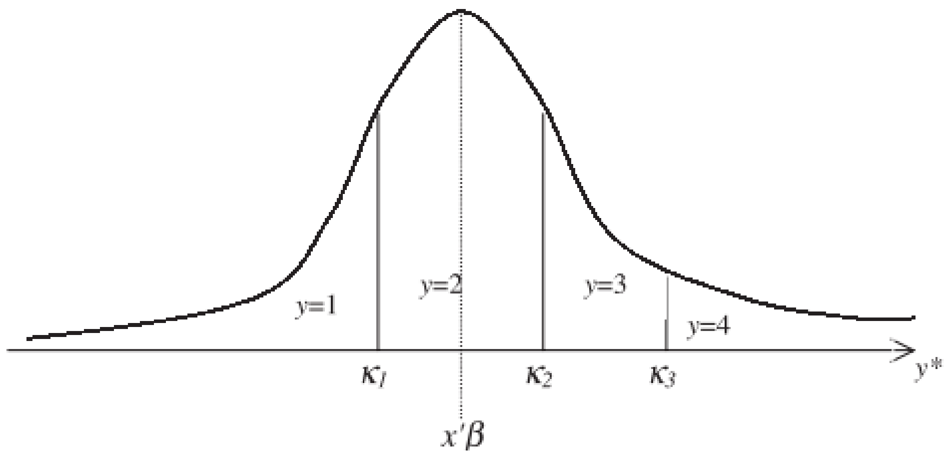
$$Y = J \quad if \quad \kappa_{J-1} < y^* < \infty$$

The parameters $\kappa_j = 1, \dots, J-1$ are known as cut points or threshold parameters.

As a result, the probability of each ordinal outcome is expressed,

$$\begin{aligned} P[Y_i = j] &= P\left[\kappa_{j-1} < y_i^* < \kappa_j\right] = P\left[\kappa_{j-1} < x_i\beta + e_i < \kappa_j\right] \\ &= P\left[\kappa_{j-1} - x_i\beta < e_i < \kappa_j - x_i\beta\right] \\ &= \Phi\left(\kappa_j - x_i\beta\right) - \Phi(\kappa_{j-1} - x_i\beta) \end{aligned}$$

The figure below depicts the density function of $y^*$for the case of $J = 4$ (Anne R. Daykin , Peter G. Moffatt).



The absence of the intercept parameter is a consequence of the $J-1$ cut points all being free parameters; they are not predefined by the model but they can be chosen or estimated experimentally or theoretically. If one of the cut points were normalized to zero, then the intercept parameter would become identified and would appear in the model.

2.5. Auto-regressive processes

The most common model for correlated data is a class of time series models which are called auto-regressive processes. These processes are used a lot in the football dynamic models where the abilities of the teams change over time. These models will be presented in Chapter 4 (paragraph 4.4.).

Definition (Time series process) *A time series process is stochastic process* $\{X_t | t \in T\}$*, which is a collection of random variables ordered in time. The set* T *is called index set and it determines the set of times at which the process is defined and observations are made.*

There are two sets of conditions under which the theory is built:

- Stationary process (the mean and the variance don't change over time)
- Ergodic process (the statistical properties of the process can be deduced from a single, sufficiently long, random sample of the process)

Definition (Auto-regressive process) *Let* Z_t *be a random process with mean* 0 *and variance* σ_z^2 *where each* Z_t *is independent. An auto-regressive process of order* p*, denoted* $AR(p)$*, is given by*

$$X_t = a_1 X_{t-1} + \cdots + a_p X_{t-p} + Z_t$$

where $X_0 = X_{-1} = \cdots = X_{1-p} = 0$

In the expression above, correlation is introduced between the random variables by the regression of X_t on past values $X_{t-1}, \dots, X_{t-p}$. The parameters $\alpha_1, \dots, \alpha_p$ are the coefficients of the auto-regressive process where a_i is called the lag i coefficient.

The $AR(1)$ process

An $AR(1)$ process is given by

$$X_t = aX_{t-1} + Z_t$$

In order to calculate the **mean** and **variance** of the process:

$$X_t = aX_{t-1} + Z_t = a(aX_{t-2} + Z_{t-1}) + Z_t = \cdots = \sum_{j=0}^{\infty} a^j Z_{t-j}$$

As a result,

$$\mathbb{E}[X_t] = \mathbb{E}\left[\sum_{j=0}^{\infty} a^j Z_{t-j}\right] = \sum_{j=0}^{\infty} a^j \mathbb{E}[Z_{t-j}] = \sum_{j=0}^{\infty} a^j \cdot 0 = 0$$

and

$$Var[X_t] = Var\left[\sum_{j=0}^{\infty} a^j Z_{t-j}\right] = \sum_{j=0}^{\infty} Var[a^j Z_{t-j}] = \sum_{j=0}^{\infty} a^{2j} \cdot \sigma_Z^2$$

The variance is comprised of an infinite sum, so its value depends on a.

- If $|a| \geq 1$ (non-stationary) then $Var[X_t] = \infty$
- if $|a| < 1$ (stationary) then it is known for a geometric series:

$$\sum_{j=0}^{\infty} a^{2j} = 1 + a^2 + a^4 + \cdots = \frac{1}{1 - a^2}$$

As a result,

$$Var[X_t] = \sigma_Z^2 \sum_{j=0}^{\infty} a^{2j} = \frac{\sigma_Z^2}{1-a^2}$$

2.6. Model selection criteria

In many statistical problems, obtaining the optimal model is the main good. For this purpose, some selection model criteria have been developed, which are based on the maximum likelihood of the model and the number of the parameters estimated. All these criteria are based on the Kullback-Leibler divergence.

Definition (Kullback-Leibler divergence) *Let us consider the probability measures P, Q defined in the same space $(\mathcal{X}, \mathcal{A})$ where $\mathcal{X}$ is the set of all possible outcomes, $\mathcal{A}$ is a set of events and P is absolutely continuous on Q $\quad (Q(A) = 0 \Rightarrow P(A) = 0\,, \forall A \in \mathcal{A})$. The Kullback-Leibler divergence (or relative entropy) from Q to P is defined to be*

$$D_{KL}(P||Q) = \int_{\mathcal{X}} \log\left(\frac{dP}{dQ}\right) dP.$$

For discrete cases the KL-distance is expressed as,

$$\sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right) = \mathbb{E}_P[logP(x)] - \mathbb{E}_P[logQ(x)].$$

In an actual problem, we have a sample of observations from the unknown mass function P which is modeled by the mass function $Q(\cdot\,|\theta)$. If we want to compare different models with respective mass functions $Q_i(\cdot\,|\theta_i)$, this can take place through an equivalent comparison of the divergences $D_{KL}(P||Q_i)$, where the best model is that with the shortest divergence from the actual mass function P. From the equation above, it is clear that the best model is that with the largest $\mathbb{E}_P[logQ(x|\theta)] = \mathbb{E}_P[l(\theta)]$.

The theory above leads to the following definitions of model selection criteria.

Definition (AIC and BIC) *Let us consider a sample of observations, a model with vector of parameters* $\theta \in \Theta \subseteq \mathbb{R}^k$ *and the maximum likelihood estimator* $\hat{\theta}$*. The Aikake Information Criterion and the Bayesian Information Criterion are defined to be*

$$AIC = -2logL(\hat{\theta}) + 2k \quad and \quad BIC = -2logL(\hat{\theta}) + klogn$$

respectively.

These criteria contain a “penalty” for the number of the model parameters. The BIC has greater “penalty” for the parameters than AIC, which also increases according to the sample size.

Chapter 3

The EM algorithm

The Expectation-Maximization (EM) algorithm is a broadly applicable type of iterative computation of maximum likelihood (ML) estimates. It is mainly used in incomplete-data problems. Its basic idea is to solve a succession of simpler problems which occur when we augment the observed variables (incomplete data) with a set of additional variables (missing data) that are unobservable or unavailable.

3.1. Theoretical Framework

Maximum likelihood estimation (MLE) is a widely known method of estimating the parameters of a probability function, given some observed data. In this procedure, the aim is to obtain the point of the parameter space that maximizes the likelihood function so that the observed data is most probable. This point is called maximum likelihood estimate. Specifically, our objective is to maximize the likelihood $L(\theta) = g(x;\theta)$ as a function of θ, after assuming the observed data x with probability density function $g(x;\theta)$, and with θ being a vector of unknown parameters in the parameter space. In order to maximize the likelihood,

$$\frac{\partial L(\theta)}{\partial \theta} = 0$$

or equivalently,

$$\frac{\partial logL(\theta)}{\partial \theta} = 0$$

However, in many statistical problems where the likelihood or log-likelihood is not quadratic, due to missing data, dependence or non-normal errors, the maximum likelihood estimate cannot be obtained by solving a simple equation or a linear system. In these situations, ML estimate is obtained by using numerical iterative methods of solution of equations such as Newton-Raphson approach. In the next paragraph, we will present an additional iterative method, the EM algorithm, which offers an attractive alternative in a variety of settings.

The EM algorithm is an iterative method which deals with estimating parameters in problems where the likelihood is complicated in structure resulting in difficult-to-compute maximization problems. A typical case is that of missing data problems. In such problems we can formulate an associated statistical problem with augmented data from which it is possible to work out the MLE. The augmented data is often called 'complete' data and the available data is called 'incomplete' data, and the corresponding likelihoods are the 'complete-data likelihood' and the 'incomplete-data' likelihood respectively. The EM algorithm is a generic method that computes the MLE of the incomplete-data problem by formulating a complete data problem. Basically it takes advantage of the simplicity of the MLE of the complete-data problem and it finally computes the MLE of the incomplete-data problem.

Let us give an example (Maya R. Gupta, Yihua Chen 2010).

"Consider the temperature outside your window for each of the 24 hours of a day, represented by $x \in \mathbb{R}^{24}$*, and say that this temperature depends on the season* $\theta \in \{summer, autumn, winter, spring\}$*, and that you know the seasonal temperature distribution* $p(x|\theta)$*. But what if you could only measure the average temperature* $y = \bar{x}$ *for some day, and you would like to estimate what season* θ *it is. In particular, you might seek the maximum likelihood estimate of* θ*, that is the value* $\hat{\theta}$ *that maximizes* $p(y|\theta)$*."*

The EM algorithm is a suitable technique that can deal with the problem above.

3.2. The EM Method

In order to use EM, we have to be given some observed data y, a parametric density function $f(y|\boldsymbol{\theta})$, a description of some complete data x that we don't have. We assume that the complete data can be modeled as continuous random vector X with density $f_c(x;\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \Omega$ for some set Ω.

Definition (complete-data log-likelihood) *We let $f_c(x;\boldsymbol{\theta})$ denote the probability density function of the random vector X which corresponds to the complete-data vector x, with $\boldsymbol{\theta} \in \Omega$ where Ω a parameter space. Then the complete-data log-likelihood function is given by*

$$logL_c(\boldsymbol{\theta}) = \log f_c(x;\boldsymbol{\theta})$$

The EM algorithm deals with the problem of solving the incomplete-data likelihood equation indirectly via iterative calculations of $logL_c(\boldsymbol{\theta})$. As it is unobservable, it is replaced by its conditional expectation given observable data y every time.
The procedure is described as follows:

- Firstly, let $k = 0$ and make an initial estimate $\boldsymbol{\theta}^{(k)}$ for $\boldsymbol{\theta}$.
- Given the observed data y and pretending for the moment that our current guess $\boldsymbol{\theta}^{(k)}$ is correct, we formulate the conditional probability distribution $f_c(x|y,\boldsymbol{\theta}^{(k)})$ for the complete data x.
- Using the probability distribution $f_c(x|y,\boldsymbol{\theta}^{(k)})$, we form the conditional expected log-likelihood, which is called Q-function:
$$Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}\right) = \mathbb{E}_{\theta^{(k)}}\{logL_c(\boldsymbol{\theta})|y\}$$
- We find the value of $\boldsymbol{\theta}$ that maximizes the Q-function, $\boldsymbol{\theta}^{(k+1)}$. This is the new estimate.
- Let $k \coloneqq k + 1$ and we go back to the second "bullet".

The traditional description of the EM algorithm consists of two main steps.

On the $(k+1)$-th iteration, the steps are, the ***Expectation Step (E-Step)*** *and the* ***Maximization Step (M-Step)****.*

E-STEP*: Compute the expected value of $logL_c(\boldsymbol{\theta})$ given the observed data y, and the current parameter estimate $\boldsymbol{\theta}^{(k)}$. It is defined,*

$$Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}\right) = \mathbb{E}_{\theta^{(k)}}\{logL_c(\boldsymbol{\theta})|y\}$$

M-STEP*: Choose $\boldsymbol{\theta}^{(k+1)}$to be any value of $\boldsymbol{\theta} \in \Omega$ so that:*

$$Q\left(\boldsymbol{\theta}^{(k+1)};\boldsymbol{\theta}^{(k)}\right) \geq Q\left(\boldsymbol{\theta};\boldsymbol{\theta}^{(k)}\right) \quad \forall \boldsymbol{\theta} \in \Omega$$

In other words, the M-Step consists of maximizing over θ the expectation computed in the E-Step.

The E-steps and the M-steps are alternated repeatedly until the procedure stops due to convergence.

Let us give an example from Maya R. Gupta and Yihua Chen (2010) to illustrate the use of the method above.

Let us consider n kids which choose one toy out of four choices. Let $y = (y_1, y_2, y_3, y_4)^T$ denote the histogram of their n choices, where y_i the number of kids that chose toy i, for $i = 1,2,3,4$. We can model this random histogram y as being multinomially distributed. In this case, the multinomial density function is expressed as,

$$f(y|p) = \frac{n!}{y_1!\,y_2!\,y_3!\,y_4!} p_1^{y_1} p_2^{y_2} p_3^{y_3} p_4^{y_4},$$

where n is the number of kids asked, that is $n = y_1 + y_2 + y_3 + y_4$ and $p = (p_1, p_2, p_3, p_4)$ is vector of probabilities with p_i being the probability that toy i is chosen, $i = 1,2,3,4$.

By assuming that the probability p of choosing each of the toys is parameterized by some value $\boldsymbol{\theta} \in (0,1)$ we have,

$$p_\theta = (p_1, p_2, p_3, p_4)^T = \left[\frac{1}{2} + \frac{1}{4}\boldsymbol{\theta}, \frac{1}{4}(1-\boldsymbol{\theta}), \frac{1}{4}(1-\boldsymbol{\theta}), \frac{1}{4}\boldsymbol{\theta}\right]^T$$

The estimation problem is to guess the value of θ that maximizes the probability of the observed histogram y. According to the parameterization above the multinomial function in our case becomes,

$$f(y|p) = \frac{n!}{y_1!\,y_2!\,y_3!\,y_4!}\left(\frac{1}{2} + \frac{1}{4}\boldsymbol{\theta}\right)^{y_1}\left(\frac{1-\boldsymbol{\theta}}{4}\right)^{y_2}\left(\frac{1-\boldsymbol{\theta}}{4}\right)^{y_3}\left(\frac{\boldsymbol{\theta}}{4}\right)^{y_4}.$$

For this simple example, the MLE can be easily found but we will instead illustrate how to use the EM algorithm to find the MLE of $\boldsymbol{\theta}$.

To illustrate the EM algorithm, we represent y as incomplete data from a five-category multinomial distribution (complete data) where the cell probabilities are,

$$q_\theta = \left[\frac{1}{2}, \frac{1}{4}\boldsymbol{\theta}, \frac{1}{4}(1-\boldsymbol{\theta}), \frac{1}{4}(1-\boldsymbol{\theta}), \frac{1}{4}\boldsymbol{\theta}\right]^T, \boldsymbol{\theta} \in (0,1).$$

The idea is to split the first of the original four categories into two categories. Thus, the complete data is $x = (x_1, x_2, x_3, x_4, x_5)$ where $y_1 = x_1 + x_2,\quad y_2 = x_3\,,\quad y_3 = x_4\,,\quad y_4 = x_5$ and the complete data density function is,

$$f_c(x|\theta) = \frac{n!}{x_1!\,x_2!\,x_3!\,x_4!\,x_5!}\left(\frac{1}{2}\right)^{x_1}\left(\frac{\theta}{4}\right)^{x_2}\left(\frac{1-\theta}{4}\right)^{x_3}\left(\frac{1-\theta}{4}\right)^{x_4}\left(\frac{\theta}{4}\right)^{x_5}$$

Our aim is to maximize the Q-function, that is to find $\boldsymbol{\theta}^{(\kappa+1)}$ so that, $\boldsymbol{\theta}^{(\kappa+1)} = argmax\ Q\big(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}\big) = argmax\mathbb{E}_{X|y,\boldsymbol{\theta}^{(k)}}\{log f_c(x|\boldsymbol{\theta})\}$.

As stated above, two steps are required.

Expectation Step: The E-Step estimates the sufficient statistics of the complete data x, given the observed data y. In our case, (x_3, x_4, x_5) are known to be (y_2, y_3, y_4). The only sufficient statistics that need to be estimated are x_1 and x_2 where $x_1 + x_2 = y_1$. After all, despite the fact that the value of $y_1$is known, x_1 and x_2 remain unknown. Estimating x_1 and x_2 using the current estimate of $\boldsymbol{\theta}$ leads to,

$$x_1^{(k)} = y_1 \cdot \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}\boldsymbol{\theta}^{(\boldsymbol{k})}} = \frac{2}{2 + \boldsymbol{\theta}^{(\boldsymbol{k})}} y_1$$

and

$$x_2^{(k)} = y_1 \cdot \frac{\frac{1}{4}\boldsymbol{\theta}^{(\boldsymbol{k})}}{\frac{1}{2} + \frac{1}{4}\boldsymbol{\theta}^{(\boldsymbol{k})}} = \frac{\boldsymbol{\theta}^{(\boldsymbol{k})}}{2 + \boldsymbol{\theta}^{(\boldsymbol{k})}} y_1$$

As a result,

$$x|y = \left(x_1^{(k)}, x_2^{(k)}, x_3, x_4, x_5\right) = \left(\tfrac{2}{2+\boldsymbol{\theta}^{(k)}} y_1, \tfrac{\boldsymbol{\theta}^{(k)}}{2+\boldsymbol{\theta}^{(k)}} y_1, y_2, y_3, y_4\right)$$

and

$$Q\big(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}\big) = \left(\frac{\boldsymbol{\theta}^{(\boldsymbol{k})}}{2 + \boldsymbol{\theta}^{(\boldsymbol{k})}} y_1 + y_4\right) log\theta + (y_2 + y_3)\log(1 - \boldsymbol{\theta})$$

Maximization Step: The M-Step becomes:

$$\begin{aligned}
\boldsymbol{\theta}^{(k+1)} &= argmax_{\boldsymbol{\theta}\in(0,1)} Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)}\right) \\
&= argmax_{\boldsymbol{\theta}\in(0,1)} \left[\left(\frac{\boldsymbol{\theta}^{(\boldsymbol{k})}}{2+\boldsymbol{\theta}^{(\boldsymbol{k})}} y_1 + y_4\right) log\boldsymbol{\theta} + (y_2+y_3)\log(1-\boldsymbol{\theta})\right] \\
&= \frac{\dfrac{\boldsymbol{\theta}^{(k)}}{2+\boldsymbol{\theta}^{(\boldsymbol{k})}} y_1 + y_4}{\dfrac{\boldsymbol{\theta}^{(k)}}{2+\boldsymbol{\theta}^{(k)}} y_1 + y_2 + y_3 + y_4}
\end{aligned}$$

The procedure above is repeated till the convergence of $\boldsymbol{\theta}$ to a $\boldsymbol{\theta}^*$ which is considered to be the MLE of $\boldsymbol{\theta}$.

3.3. Convergence of the EM algorithm

While the EM algorithm is in progress, the $(k+1)$th guess $\boldsymbol{\theta}^{(k+1)}$ is never found to be less than the kth guess $\boldsymbol{\theta}^{(k)}$. This property is called *monotonicity* of the EM algorithm (Maya R. Gupta, Yihua Chen 2010). The *monotonicity* of the EM algorithm guarantees that while the EM algorithm is in progress the guesses-values of $\boldsymbol{\theta}$ won't get any worse in terms of their likelihood, but it cannot guarantee the convergence of the sequence $\{\boldsymbol{\theta}^{(k)}\}$. Actually, there is no general convergence theorem for the EM algorithm; the convergence of the sequence $\{\boldsymbol{\theta}^{(k)}\}$ depends on the characteristics of the log-likelihood and $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(k)})$.

The convergence of the EM algorithm is determined by using a suitable stopping rule like,

$$|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}| < \varepsilon$$

for some $\varepsilon > 0$.

So when the rule above happens then the procedure stops with $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(k+1)}$ being the result-estimate of the incomplete-data problem.

Theorem (EM algorithm inequality) *If the observable likelihood $L(\boldsymbol{\theta}|y)$ is bounded, then the value of $\boldsymbol{\theta}^*$to which the algorithm converges, is a local maximum of $L(\boldsymbol{\theta}|y)$.*

Proof: *Initially, we have that*

$$L(\theta|y,x) = f_\theta(x,y) = f_\theta(y)f_\theta(x|y) = L(\theta|y)f_\theta(x|y)$$

and with logarithm in the equation above it is

$$\ell(\theta|y,x) = \ell(\theta|y) + log f_\theta(x|y)$$

If X is an absolutely continuous random variable, by multiplying the equality members with the density $f_{\theta^{(0)}}(x|y)$ and by integrating by x:

$$\int \ell(\theta|y,x)\, f_{\theta^{(0)}}(x|y)dx = \int \ell(\theta|y) f_{\theta^{(0)}}(x|y)dx + \int log f_\theta(x|y) f_{\theta^{(0)}}(x|y)dc$$

Respectively, if X were a discrete random variable, we would multiply with the probability $\mathbb{P}_{\theta^{(0)}}(X = x|y)$ and we would take the sum by x instead of integrating.

We observe that:

$$\int \ell(\theta|y,x)\, f_{\theta^{(0)}}(x|y)dx = \mathbb{E}[\,\ell(\theta|y,X)|y] = Q_{\theta^{(0)}}(\theta)$$

$$\int \ell(\theta|y)\, f_{\theta^{(0)}}(x|y)dx = \ell(\theta|y) \int f_{\theta^{(0)}}(x|y)dx = \ell(\theta|y)$$

Now, we set,

$$\mathcal{H}_{\theta^{(0)}}(\theta) = -\int log f_\theta(x|y) f_{\theta^{(0)}}(x|y)dx = -\mathbb{E}_{\theta^{(0)}}[\,log f_\theta(X|y)|y].$$

So the observable likelihood is analytically written as:

$$\ell(\theta|y) = Q_{\theta^{(0)}}(\theta) + \mathcal{H}_{\theta^{(0)}}(\theta).$$

Now by using Jensen inequality we have:

$$
\begin{aligned}
&\mathcal{H}_{\theta^{(0)}}\left(\theta^{(1)}\right) - \mathcal{H}_{\theta^{(0)}}\left(\theta^{(0)}\right) \\
&\qquad = -\mathbb{E}_{\theta^{(0)}}[logf_{\theta^{(1)}}(X|y)|y] + \mathbb{E}_{\theta^{(0)}}[logf_{\theta^{(0)}}(X|y)|y] \\
&\qquad = -\mathbb{E}_{\theta^{(0)}}\left[\frac{logf_{\theta^{(1)}}(X|y)}{logf_{\theta^{(0)}}(X|y)}\middle|y\right] \\
&\qquad \geq -log\mathbb{E}_{\theta^{(0)}}\left[\frac{logf_{\theta^{(1)}}(X|y)}{logf_{\theta^{(0)}}(X|y)}\middle|y\right] \\
&\qquad = -log\int \frac{logf_{\theta^{(1)}}(x|y)}{logf_{\theta^{(0)}}(x|y)} logf_{\theta^{(0)}}(x|y)dx \\
&\qquad = -log\int logf_{\theta^{(1)}}(x|y) = -log1 = 0
\end{aligned}
$$

This inequality is known as the fundamental inequality of EM algorithm. If $\theta^{(0)}$ is the current estimate of θ, then this inequality shows us that for any value of our next estimate $\theta^{(1)}$, the function $\mathcal{H}_{\theta^{(0)}}(\cdot)$ will not be smaller than the current value $\mathcal{H}_{\theta^{(0)}}(\theta^{(0)})$. As the function $\mathcal{H}$ is increased in every step of the EM algorithm, we can ignore $\mathcal{H}$ and focus on the function Q.

If we select any value $\theta^{(1)}$ that increases the value of the function $Q_{\theta^{(0)}}(\cdot)$, that is $Q_{\theta^{(0)}}\left(\theta^{(1)}\right) > Q_{\theta^{(0)}}(\theta^{(0)})$, then we will have $\ell\left(\theta^{(1)}\middle|y,x\right) > l\left(\theta^{(0)}\middle|y,x\right)$. By repeating this procedure, we produce a sequence of estimates which increases the value of the observable likelihood in every step of the algorithm and finally converges to a local maximum.

It is clear that, if we select precisely the value that maximizes the function $Q_{\theta^{(0)}}(\cdot)$ as $\theta^{(1)}$, that is $\theta^{(1)} = argmax_{\theta}Q_{\theta^{(0)}}(\theta)$, then the algorithm will have the maximum speed of convergence. This is the aim of the EM algorithm. However, even if the analytical maximization of the function $Q_{\theta^{(0)}}(\cdot)$ is not feasible, the algorithm will anyway converge to a local maximum of the observable likelihood if we select in every step a new estimate that increases, even a little, the current value of the function Q.

Chapter 4

Football Modeling

It is true that football is probably the most popular sport in the world. Football's history began in England in 1863 where people were kicking a leather ball filled with feathers and hair in their neighborhoods and, after a continuous evolution, it became an international attraction. In recent years, more and more companies have been associated with football depending economically on it and more and more staff has been working on it. Moreover, the sport has become extremely competitive and complicated. These facts have led to a huge statistical interest in the sport. Visualizations, performance analytics, outcome prediction etc, came to improve players and teams making their performance more effective.

Football is a low-score sport with a lot of surprises and changes during a match which make it hard to predict the final outcome. A lot of statistical modeling has been developed in order to assist professionals of all kinds to improve their influence on the sport. In this chapter we will show different types of statistical models like win-draw-loss models and score models which are used in predicting the outcome of football matches.

4.1. Naive Models

In this paragraph we will present some basic and easy-to-use statistical models with their characteristics that can be used in predicting football outcomes. Although these models do not have specific properties that are essential in football modeling, they are an obvious initial approach.

4.1.1. The Bradley-Terry ordinal model

The Bradley-Terry model is a preliminary simplistic model which can predict the outcome of a paired comparison. Given a pair of individuals i and j drawn from some population with X_i, X_j being variables relating to i and j respectively, it estimates the probability that the pairwise comparison $X_i > X_j$ turns out true, as

$$P(X_i > X_j) = P(Y_{ij} = 1) = \frac{p_i}{p_i + p_j}$$

where p_i is a positive real-valued score assigned to individual i and Y_{ij} is a binary variable; if $Y_{ij} = 1$ then $X_i > X_j$ and if $Y_{ij} = 0$ then $X_i < X_j$. In the case of a football game, i is the home team, j is the away team, X_i denotes the goals that team i achieved in the match and p_i represents the ability of team i. Actually, $P(X_i > X_j)$ is the probability of team i prevailing over team j.The Bradley-Terry model uses exponential score functions $p_i = e^{\gamma_i}$ so it can be written as

$$P(X_i > X_j) = P(Y_{ij} = 1) = \frac{e^{\gamma_i}}{e^{\gamma_i} + e^{\gamma_j}} = \frac{e^{\gamma_i - \gamma_j}}{1 + e^{\gamma_i - \gamma_j}}$$

where the parameters γ_i are associated with the ability of the teams and need to be estimated. For example, γ_i could have information about the rate of chances that team i generally creates during a match. It is clear that the outcome of the game is determined by the difference $\gamma_i - \gamma_j$. For identifiability, a sum-to-zero constraint to the parameters is needed, $\sum_i \gamma_i = 0$.

We can notice that the model above has only two outcomes (win or lose) and that is the reason that the Bradley-Terry model can be preferably used in basketball games rather than football games, where one of the two teams win in the end. For football matches we have to extend the model above by taking into consideration the case of a draw. An early approach on such modeling was the Rao-Kupper model (1967) which consists of two types of models:

- Model A:

$$P(X_i > X_j) = \frac{p_i}{p_i + \theta p_j}$$

$$P(X_i < X_j) = \frac{p_j}{p_j + \theta p_i}$$

$$P(X_i = X_j) = \frac{p_i p_j(\theta^2 - 1)}{(p_i + \theta p_j)(p_j + \theta p_i)}$$

- Model B:

$$P(X_i > X_j) = \frac{p_i}{p_i + p_j + \nu\sqrt{p_i p_j}}$$

$$P(X_i < X_j) = \frac{p_j}{p_j + p_i + \nu\sqrt{p_i p_j}}$$

$$P(X_i = X_j) = \frac{\nu\sqrt{p_i p_j}}{p_i + p_j + \nu\sqrt{p_i p_j}}$$

For $\theta = 1$ and $\nu = 0$ respectively we get no draws.

By extending the binary Bradley-Terry model to a model with three categories; the variable Y_{ij} is coded as 2 if the home team wins, 1 in the case of draw and 0 in the case of victory of the visiting team, we lead to the cumulative probabilities in the form

$$P\left(Y_{ij} \leq k\right) = \frac{\exp(\mu_k + \gamma_i - \gamma_j)}{1 + \exp(\mu_k + \gamma_i - \gamma_j)} \; , k \in \{0,1,2\}$$

where $\mu_0 < \mu_1 < \mu_2$ are unknown cut-point parameters-thresholds which determine the preference for each specific category.
The probability for a single response category can be derived as follows,

$$P\left(Y_{ij} = k\right) = P\left(Y_{ij} \leq k\right) - P(Y_{ij} \leq k - 1)$$

By slight abuse of notation, in the pursuit of completeness we define the threshold of the last category $\mu_2 = +\infty$ so that $P\left(Y_{ij} \leq 2\right) = 1$.
The model is over-parameterized in the sense that it is exactly the same even if we add a fixed constant α to all values γ_i because the differences $\gamma_i - \gamma_j$ remain unchanged. The constant a may denote the home advantage. Therefore,

$$P\left(Y_{ij} \leq k\right) = \frac{\exp(\mu_k + \alpha + \gamma_i - \gamma_j)}{1 + \exp(\mu_k + \alpha + \gamma_i - \gamma_j)} \; , k \in \{0,1,2\}$$

The constant parameter α can be replaced by α_i so that home effects are team-specific instead of being equal for all teams. Concerning the ability γ_i of team i, it is given by

$$\gamma_i = \beta_i z_i$$

where z_i is a vector of covariates and β_i is vector of coefficients.

By assuming the latent linear predictor of the ordered model,

$$Y_{ij}^{*} = \alpha_i + \beta_i z_i - \beta_j z_j + \varepsilon$$

where $\varepsilon \sim N(0,1)$ represents the error term, the ordinal categories re

$$\begin{aligned} Y_{ij} &= 0\,, \qquad \infty < Y_{ij}^{*} \leq \mu_0 \\ Y_{ij} &= 1\,, \qquad \mu_0 < Y_{ij}^{*} \leq \mu_1 \\ Y_{ij} &= 2\,, \qquad \mu_1 < Y_{ij}^{*} < \infty \end{aligned}$$

Estimation

Maximum likelihood estimation is applied to estimate the value for the parameters β_i, β_j and the thresholds $\mu_k\,, k = 0{,}1$. The log-likelihood function lnL of the model is,

$$lnL = \sum_{i,j,Y_{ij}=0} \left(lnF_{ij0}\right) + \sum_{i,j,Y_{ij}=1} \left(lnF_{ij1} - lnF_{ij0}\right) + \sum_{i,j,Y_{ij}=2} \left(-lnF_{ij1}\right)$$

where $F_{ijk}\,, k = 0{,}1$ are the cumulative probabilities of the model.
By maximizing the equation of the log-likelihood for each parameter, the estimates for the parameters are obtained.

4.1.2. The Double Poisson model

The Poisson distribution has been widely accepted as a simple modeling approach for the distribution of the number of goals in sports involving two competing teams.
We assume for the i-th match, $i = 1, \dots, n$ that (X_1, X_2), which denote the achieved goals by the two opponents, are modeled as two conditionally independent Poisson,

$$X_{1i} \sim Poisson(\lambda_{1i})$$

$$X_{2i} \sim Poisson(\lambda_{2i})$$

with joint density function the Double Poisson probability function f_{DP},

$$f_{DP}(x_1, x_2) = e^{-\lambda_1} \frac{\lambda_1^{x_1}}{x_1!} \cdot e^{-\lambda_2} \frac{\lambda_1^{x_2}}{x_2!}$$

The parameters $\lambda_{1i}, \lambda_{2i}$ represent the scoring rates, that is the expected number of goals for the home and the away team respectively in the i-th observation-game.
Starting with the probability Poisson mass function in order to obtain the exponential dispersion form and indentify the link function for the parameters estimation we have the following steps:

$$f(x_i; \lambda) = \frac{e^{-\lambda_i} \lambda^{x_i}}{x_i!} = \exp(x_i log(\lambda_i) - \lambda_i - log(x_i!)) \Rightarrow$$

$$a(\varphi) = 1, \qquad \theta_i = \log(\lambda_i) \Leftrightarrow \lambda_i = e^{\theta_i}, \qquad b(\theta_i) = \lambda_i = e^{\theta_i},$$

$$c(x_i, \varphi) = \log\left(\frac{1}{x_i!}\right)$$

In the exponential dispersion family, θ_i is the canonical parameter which depends on a model of linear predictors. Therefore, the log of the expectation, $\log(\lambda_i)$ can be modeled by Poisson regression as,

$$g(\lambda_i) = \theta_i = \log(\lambda_i) = \boldsymbol{\beta} x$$

where x is a vector of explanatory variables and $\boldsymbol{\beta}$ is a vector of coefficients.
The scoring rate of the kth team in the ith match λ_{ki} depends on the attacking ability of the team k as well as on the defensive ability of the opponent (Joel Liden 2016). As a result,

$$\log(\lambda_{1i}) = \mu + home + att_{h_i} + def_{a_i},$$

$$\log(\lambda_{2i}) = \mu + att_{a_i} + def_{h_i},$$

where att_k and def_k are the attack and defense parameters of the team k respectively, h_i and a_i are the home and the away team in the i-th match, $home$ represents the home advantage and μ represents the constant intercept.
In order to achieve identifiability, we use sum-to-zero constraints for attacking and defensive abilities,

$$\sum_{k=1}^{n} att_k = 0$$

$$\sum_{k=1}^{n} def_{\mathrm{k}} = 0$$

Estimation

Considering the Poisson regression form,

$$\log(\lambda_{1i}) = \boldsymbol{\beta}_1{}^T w_{1i}$$
$$\log(\lambda_{2i}) = \boldsymbol{\beta}_2{}^T w_{2i}$$

where $\lambda_{1i}, \lambda_{2i}$ are the scoring rates of the home and away team respectively in the ith match and $w_{1i}, w_{2i} \in \mathbb{R}^d$ the respective vectors of covariates with $\beta_1{}^T, \beta_2{}^T \in \mathbb{R}^d$ coefficients, the log-likelihood function is:

$$logL = \sum_{i=1}^{n}[-\lambda_1 - \lambda_2 + x_{1i}\log(\lambda_{1i}) + x_{2i}\log(\lambda_{2i}) - \log(x_{i1}!) - \log(x_{2i}!)]$$
$$= \sum_{i=1}^{n}\left[-e^{\beta_1^T w_{1i}} - e^{\beta_2^T w_{2i}} + x_{1i}\beta_1^T w_{1i} + x_{2i}\beta_2^T w_{2i} - \log(x_{i1}!) - \log(x_{2i}!)\right]$$

The maximum likelihood estimation for parameters w_{1k} and w_{2k}, $k = 1, \dots, d$ is carried out through the following equations:

$$\frac{\partial logL}{\partial \beta_{1k}} = 0, \qquad for\ k = 1, \dots, d$$

$$\frac{\partial logL}{\partial \beta_{2k}} = 0, \qquad for\ k = 1, \dots, d$$

For the solution of these equations the Newton-Raphson method is suggested which is presented in the Appendix. For this method , the matrix of second derivatives is needed.

4.1.3. The Negative Binomial model

Just like the Poisson case, a negative binomial model can be used for count data such as the number of goals for two opponents. In the world of football, empirical evidence has shown over the years that there is over-dispersion in the number of teams' goals in most leagues. An important characteristic of negative binomial distribution is that allows for over-dispersion as it has larger variance than the mean, something that can be seen as a disadvantage in Poisson distribution, where the mean is equal to the variance,.

The negative binomial distribution is a discrete probability distribution that models the number of successes in a sequence of independent and identically distributed Bernoulli trials before a specified number of failures (denoted r) occurs. Thus, the negative binomial mass function is derived as,

$$f(k;r,p) = P(X=k) = \binom{k+r-1}{r-1}(1-p)^k p^r \quad ,k = 0,1,2,\ldots$$

In our case, we have

$$f(y_i) = \frac{\Gamma(y_i+r)}{\Gamma(r)\Gamma(y_i+1)}\left(\frac{r}{\lambda_i+r}\right)^r\left(\frac{\lambda_i}{\lambda_i+r}\right)^{y_i}$$

where Γ is the gamma distribution, λ_i denotes the scoring rate of team i.

By obtaining the exponential dispersion form,

$$f(y_i;\theta_i;\varphi) == \exp\left(log\left(\left(\frac{r}{\lambda_i+r}\right)^r\right) + y_i\, log\left(\frac{\lambda_i}{\lambda_i+r}\right) + log\left(\frac{\Gamma(y_i+r)}{\Gamma(r)\Gamma(y_i+1)}\right)\right) \Rightarrow$$

$$a(\varphi) = 1, \qquad \theta_i = \log\left(\frac{\lambda_i}{\lambda_i+r}\right), \qquad b(\theta_i) = -rlog\left(\frac{r}{\lambda_i+r}\right),$$

$$c(y_i,\varphi) = \log\left(\frac{\Gamma(y_i+r)}{\Gamma(r)\Gamma(y_i+1)}\right)$$

As a result, the expected value and the variance in the negative binomial case can be retrieved by the following procedure:

$$\theta_i = \log\left(\frac{\lambda_i}{\lambda_i + r}\right) \Leftrightarrow \frac{\lambda_i}{\lambda_i + r} = e^{\theta_i} \Rightarrow \lambda_i = \frac{re^{\theta_i}}{1 - e^{\theta_i}}$$

$$b(\theta_i) = -rlog\left(\frac{r}{r + \lambda_i}\right) = -rlog(1 - e^{\theta_i})$$

So it is,

$$\mathbb{E}[Y_i] = b^{'}(\theta_i) = \frac{re^{\theta_i}}{1 - e^{\theta_i}} = \lambda_i$$

$$Var[Y_i] = b^{''}(\theta_i)a(\varphi) = \frac{re^{\theta_i}}{(1 - e^{\theta_i})^2} = \lambda_i + \frac{1}{r}\lambda_i^2$$

For $r \to \infty$ we can see that we get a Poisson model. As for the link function in the negative binomial case we have $g(\lambda_i) = \theta_i = \log\left(\frac{\lambda_i}{\lambda_i + r}\right) = x_i\beta$.

Since $\lambda_i > 0$ the image of $g(\lambda_i) \in (-\infty, 0)$. Therefore, the canonical link function is not a good choice. On the other side a log-link function (similarly to the case of Poisson model) is a better choice as it allows for positive values.

So, similarly to the Double Poisson model we have:

$$log(\lambda_{1,i}) = \alpha + \beta_1 att_{h,i} + \beta_2 def_{g,i}$$

$$log(\lambda_{2,i}) = \alpha + \beta_1 att_{g,i} + \beta_2 def_{h,i}$$

where att and def are the attack and defense parameters, h and g are the indicators of the home and the guest team respectively, i is the number of our observation-game and a is a constant parameter. The estimation of the parameters is similar to the Double Poisson model (paragraph 4.1.2.).

The models that were presented above are simple and easy-to-use. However they do not have some important properties that we need in studying football results. **Dependence** between the opponents, **excess** of some specific outcomes and **dynamic abilities** are some specifications that need to be taken into consideration as they play an important role in the quality of our model we use.

4.2. Models with dependence parameter

We saw some simple approaches in studying football results which do not contain dependence between the random variables. However, several researchers have shown the existence of a correlation between the numbers of goals scored by the two opponents. In team sports, such us football, it is reasonable to consider that the two random variables are correlated (either positively or negatively) as the two teams interact during the game. For example, if a team loses during a game, then it will try to score as soon as possible which affects the speed of the game as well as the rate of the chances of the opponent too. On the other hand, when a team has a totally offensive style of playing making many chances, this may affect negatively the net scoring of the opponent team which may only defend during the game. In this paragraph we will present models that contain dependence between the outcome variables.

4.2.1. Two-dimensional copula model

Copulas are very fashionable multivariate distributions contributing in application to many disciplines, like biostatistics, finance etc. Thus, one way to study football outcomes and insert a correlation between the two opponents is a two-dimensional copula. Two-dimensional copulas can produce flexible bivariate distributions with flexible marginal distributions and flexible dependence structure.

In our case, we want to predict the outcome in football games with the goal scoring of each team being a discrete random variable. As it is mentioned in **Chapter 1**, there are specific types of copulas dealing with discrete cases.

Provided that between the two opponents in a football match there is not only positive but also negative correlation, the Frank copula is a reasonable choice as $\theta \in (-\infty, +\infty) \setminus \{0\}$. So it is,

$$C(u_1, u_2|\theta) = \frac{1}{\theta}\log\{1 + \frac{\left(e^{-\theta u_1} - 1\right)\left(e^{-\theta u_2} - 1\right)}{e^{-\theta} - 1}\ , \quad \theta \in \mathbb{R} \setminus \{0\}$$

If we consider $F_X(x), F_Y(y)$ the cumulative distribution functions for the number of goals of the home and the away team respectively, our copula is expressed as follows:

$$C(F_X(x), F_Y(y)|\theta) = \frac{1}{\theta}\log\{1 + \frac{\left(e^{-\theta F_X(x)} - 1\right)\left(e^{-\theta F_Y(y)} - 1\right)}{e^{-\theta} - 1}$$

with $\theta \in \mathbb{R} \setminus \{0\}$.

- In the Poisson case, we have,

$$u_1 = F_X(x) = P(X \leq x) = \sum_{k=0}^{x} \frac{\lambda_1^k e^{-\lambda_1 x}}{k!}$$

$$u_2 = F_Y(y) = P(Y \leq y) = \sum_{k=0}^{y} \frac{\lambda_2^k e^{-\lambda_2 y}}{k!}$$

where λ_1, λ_2 denote the rate of scoring of the home and the away team respectively.

- In the negative binomial case,

$$u_1 = F_X(x) = P(X \leq x) = \sum_{k=0}^{x} \frac{\Gamma(k+r)}{\Gamma(r)\Gamma(k+1)} \left(\frac{r}{\lambda_1 + r}\right)^r \left(\frac{\lambda_1}{\lambda_1 + r}\right)^k$$

$$u_2 = F_Y(y) = P(Y \leq y) = \sum_{k=0}^{y} \frac{\Gamma(k+r)}{\Gamma(r)\Gamma(k+1)} \left(\frac{r}{\lambda_2 + r}\right)^r \left(\frac{\lambda_2}{\lambda_2 + r}\right)^k$$

where λ_1, λ_2 denote the rate of scoring of the home and the away team respectively.
Since the copula function is actually the cumulative distribution function (cdf) and not the joint probability mass function (pmf), the probabilities of specific outcomes can be retrieved as follows:

- $P(X = 0, Y = 0) = C(F_X(0), F_Y(0))$
- $P(X = x, Y = 0) = C\big(F_X(x), F_Y(0)\big) - C\big(F_X(x-1), F_Y(0)\big)\ , x = 1,2, \ldots$
- $P(X = 0, Y = y) = C\big(F_X(0), F_Y(y)\big) - C\big(F_X(0), F_Y(y-1)\big)\ , y = 1,2, \ldots$
- $P(X = x, Y = y) = C\big(F_X(x), F_Y(y)\big) - C\big(F_X(x-1), F_Y(y)\big) - \big(F_X(x), F_Y(y-1)\big) + C\big(F_X(x-1), F_Y(y-1)\big)\ ,$ $x, y = 1,2, \ldots$

Dependence parameter θ

As it is mentioned, the dependence parameter θ in the Frank copula allows for negative correlation between the home goals and the away goals which is appropriate, since historic data suggests this. Moreover, in our case, the dependence parameter is not the Pearson type of correlation in which the interval of θ would be $[-1,1]$. Kendall's τ is a measure of correlation-concordance that works in our case. For the Frank copula, Kendall's τ can be expressed as follows:

$$\tau = f(\theta) = 1 + \frac{4}{\theta}\left[\int_0^{\theta} \frac{\alpha}{\theta(e^{\alpha} - 1)} d\alpha - 1\right]$$

Since the function f is invertible, θ can be easily estimated using an estimate of Kendall's τ. So it is

$$\tau = f(\theta) \Leftrightarrow \theta = f^{-1}(\tau)$$

where τ is a Kendall's estimate. It is noticeable that θ can be estimated through τ.

Estimation

Assume that we have a set of n observed match results,

$$(x_{11}, x_{21}), (x_{12}, x_{22}), \dots, (x_{1n}, x_{2n})$$

where x_{1i} and x_{2i} are the number of goals scored by the home and the away team respectively in the ith match, and that we also have corresponding explanatory varialbles-vectors w_{1i}, w_{2i} for each match.

The log-likelihood of the model is,

$$\ell(\boldsymbol{\beta}, \theta) = \sum_{i=1}^{n} \log[h_\theta(x_{1i}, x_{2i})].$$

where $\boldsymbol{\beta}$ is a vector of coefficients. Concerning the function h_θ:

$$h_\theta(x_{1i}, x_{2i}) = C_\theta\big(F_1(x_{1i}), F_2(x_{2i})\big) - C_\theta\big(F_1(x_{1i}-1), F_2(x_{2i})\big) - C_\theta\big(F_1(x_{1i}), F_2(x_{2i}-1)\big) + C_\theta\big(F_1(x_{1i}-1), F_2(x_{2i}-1)\big), \quad x_{1i}, x_{2i} = 1,2, \dots$$

where F_1, F_2 the marginal cumulative functions of the goals achieved by the home and the away team respectively.

The parameter estimates $\widehat{\boldsymbol{\beta}}$ and $\hat{\theta}$ can be found by the maximum likelihood estimation as $\widehat{\boldsymbol{\beta}}, \hat{\theta} = argmax_{\boldsymbol{\beta},\theta} \ell(\boldsymbol{\beta}, \theta)$.

In practice, the numerical computations required to find the maximum are very heavy. Instead we use inference for the margins to estimate the marginal parameters and copula parameters separately.

4.2.2. The bivariate Poisson model

In the paragraph **4.1.2** we presented the double Poisson approach on football modeling which is a simple approach consisting of two independent and Poisson distributed random variables. In this paragraph we will show the bivariate Poisson distribution which is an advanced Poisson-model version allowing also for dependence between the random variables. After all, as it is mentioned, in team sports like football, there is correlation between the two opponents during the game.

The bivariate Poisson distribution

Consider three random variables X_1, X_2, X_3 which follow independent Poisson distributions with parameters $\lambda_1, \lambda_2, \lambda_3$ respectively. As we want to construct a bivariate model we will apply trivariate reduction. We create X, Y such as,

$$X = X_1 + X_3$$
$$Y = X_2 + X_3$$

The random variables X, Y follow jointly the bivariate Poisson distribution $BP(\lambda_1, \lambda_2, \lambda_3)$ with joint probability function f_{BP},

$$f_{BP}(x, y) = \exp\{-(\lambda_1 + \lambda_2 + \lambda_3)\} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_{k=0}^{\min(x,y)} \binom{x}{k} \binom{y}{k} k! \left(\frac{\lambda_3}{\lambda_1 \lambda_2}\right)^k .$$

This bivariate distribution allows for dependence between the random variables. As for the marginal distributions, it is obvious that:

$$X \sim Poisson(\lambda_1 + \lambda_3) \quad with \quad \mathbb{E}[X] = \lambda_1 + \lambda_3$$
$$Y \sim Poisson(\lambda_2 + \lambda_3) \quad with \quad \mathbb{E}[Y] = \lambda_2 + \lambda_3$$

Moreover, $Cov(X, Y) = \lambda_3$ which leads to the consideration that λ_3 is a measure of dependence between the two random variables. If $\lambda_3 = 0$, then the two random variables are independent and the bivariate Poisson distribution reduces to the product of two independent Poisson distributions which is the double Poisson distribution that we presented in **4.1.2**.

When using this bivariate Poisson distribution to model football outcomes, it is obvious that X_1 and X_2 denote the goals of the home and the away team respectively, with λ_1 and λ_2 reflecting the scoring rates of the two teams. The variable X_3 denotes the goals from common cause, so λ_3 reflects game conditions such as the stadium, the weather, the speed of the game etc.

Estimation

In football modeling, we have to use realistic models where the parameters are expressed through covariates. In the case of the bivariate Poisson model, we have the regression form as follows:

$$
\begin{gathered}
(X_i, Y_i) \sim BP(\lambda_1, \lambda_2, \lambda_3), \\
\log(\lambda_{1i}) = w_{1i}\boldsymbol{\beta}_1, \\
\log(\lambda_{2i}) = w_{2i}\boldsymbol{\beta}_{2,} \\
\log(\lambda_{3i}) = w_{3i}\boldsymbol{\beta}_3,
\end{gathered}
$$

where $i = 1, \ldots, n$ denotes i-th observation-match, w_{ki} is a vector of explanatory variables for the i-th match used to model λ_{ki} and $\boldsymbol{\beta}_k$ are the regression coefficients, $k = 1,2,3$.

It is clear that the explanatory variables that are used to model each parameter λ_{ki} , $k = 1,2,3$, $i = 1, \ldots, n$, are different as each parameter may be influenced by different characteristics and variables. For that reason the estimation of the parameters cannot be accomplished straightforwardly. Thus, in order to obtain maximum likelihood estimates, we make use of the EM algorithm. To construct the EM algorithm for the bivariate Poisson regression model, we make use of the trivariate reduction. Suppose that for the i-th observation, X_{1i}, X_{2i}, X_{3i} represent the unobserved data, whereas $X_i = X_{1i} + X_{3i}$ and $Y_i = X_{2i} + X_{3i}$ are the observe data. Initially, we need to estimate the unobserved data through their conditional expectations and then fit the Poisson regression models to the pseudo-values obtained by the E- step. The complete data log-likelihood is given by

$$
L(\varphi) = -\sum_{i=1}^{n}\sum_{k=1}^{3}\lambda_{ki} + \sum_{i=1}^{n}\sum_{k=1}^{3}x_{ki}\log(\lambda_{ki}) - \sum_{i=1}^{n}\sum_{k=1}^{3}\log(x_{ki}!),
$$

where $\varphi = (\boldsymbol{\beta}'_1, \boldsymbol{\beta}'_2, \boldsymbol{\beta}'_3)$.

The EM algorithm for the bivariate Poisson model is:

E-step: We calculate the conditional expected values of X_{3i}, $i = 1, \dots, n$ by using the current parameter values of k iteration $(\varphi^{(k)}, \lambda_{1i}^{(k)}, \lambda_{2i}^{(k)}, \lambda_{3i}^{(k)})$:

$$s_i = \mathbb{E}\left[X_{3i}\middle|X_i, Y_i, \varphi^{(k)}\right) = \begin{cases} \lambda_{3i}^{(k)} \cdot \dfrac{f_{BP}\left(x_i - 1, y_i - 1\middle|\lambda_{1i}^{(k)}, \lambda_{2i}^{(k)}, \lambda_{3i}^{(k)}\right)}{f_{BP}\left(x_i, y_i\middle|\lambda_{1i}^{(k)}, \lambda_{2i}^{(k)}, \lambda_{3i}^{(k)}\right)} & , \min(x_i, y_i) > 0 \\ 0 & , \min(x_i, y_i) = 0 \end{cases}$$

where f_{BP} the mass function of the bivariate Poisson distribution.

M-step: We update the estimates:

$$\beta_1^{(k+1)} = \hat{\beta}(x - s, W_1),$$
$$\beta_2^{(k+1)} = \hat{\beta}(y - s, W_2),$$
$$\beta_3^{(k+1)} = \hat{\beta}(s, W_3),$$
$$\lambda_{ki}^{(k+1)} = \exp(W_{ki}^T\, \hat{\beta}_k^{(k+1)})\,, k = 1,2,3$$

where $s = (s_1, \dots, s_n)^T$ is the $n \times 1$ vector calculated in the E-step and $\hat{\beta}(x, W)$ are the maximum likelihood estimates of a Poisson model with response vector x and W data matrix.

Model specification

A simple regression form of the model above is :

$$(X_i, Y_i) \sim BP(\lambda_1, \lambda_2, \lambda_3)$$
$$\log(\lambda_{1i}) = \mu + home + att_{h_i} + def_{a_i}$$
$$\log(\lambda_{2i}) = \mu + att_{a_i} + def_{h_i}$$

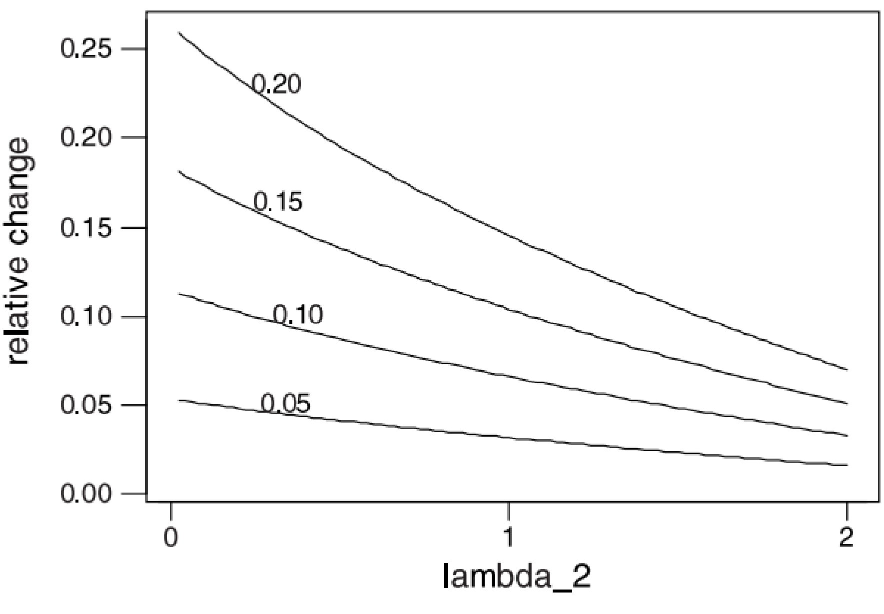
For ease of interpretation we choose sum-to-zero constraints on the explanatory variables.

For the covariance parameter λ_{3i} we may assume the general form:

$$\log(\lambda_{3i}) = \beta_0 + \gamma_1 \beta_{h_i}^{home} + \gamma_2 \beta_{a_i}^{away} + \gamma \boldsymbol{\beta} w_i$$

where β_0 is a constant parameter, $\beta_{h_i}^{home}$ and $\beta_{a_i}^{away}$ are the parameters that depend on the home and the away team respectively, w_i is a vector of covariates for the i-th match and $\boldsymbol{\beta}$ a vector of coefficients. The parameters γ_1 and γ_2 are dummy binary indicators taking values 0 or 1 as well as γ is a parameter-vector that that takes also values 0 or 1. These values of parameters γ_1, γ_2 and γ depend on the model that we consider. Usually, we consider models with constant λ_3, that is $\gamma_1 = \gamma_2 = 0 \; and \; \gamma = 0$ which makes the models easier to use. However, using covariates on λ_3 helps us to have more insight on the influence of λ_{3i} in each observation i.

The effect of λ_3 in draws



The figure above is an output presented by Karlis and Ntzoufras (2003) which shows the relative change in the probability of a draw for different values of the parameter λ_3 $(0.05, 0.10, 0.15, 0.20)$ when the two competing teams have marginal means equal to $\lambda_1 = 1$ and $\lambda_2 \in [0.1,2]$ respectively.

4.2.3. The bivariate Conway-Maxwell Poisson model

The bivariate Poisson distribution is widely used for modeling bivariate count data. However, its marginal equi-dispersion may prove limiting in some cases such us football outcomes where, as it is mentioned, there is over-dispersion.
The bivariate Conway-Maxwell Poisson (COM-Poisson) distribution includes three bivariate discrete distributions: bivariate Poisson, bivariate Bernoulli, bivariate geometric. It also contains an added dispersion parameter and as a result, the bivariare COM-Poisson distribution deals with bivariate count data in the presence of data dispersion (over-dispersion or under-dispersion).
Before presenting the bivariate Conway-Maxwell Poisson distribution and its properties we will show the univariate case.

Conway-Maxwell Poisson distribution
The COM-Poisson distribution was introduced by Conway and Maxwell and its mass function is,

$$f(x;\lambda,\nu) = P(X = x|\lambda,\nu) = \frac{\lambda^x}{(x!)^\nu}\frac{1}{Z(\lambda,\nu)} \quad , \quad x \in \mathbb{N}, \lambda > 0, \nu \geq 0$$

where $Z(\lambda,\nu) = \sum_{k=0}^{\infty}\frac{\lambda^k}{(k!)^\nu}$ *is the normalizing constant and* $\lambda = \mathbb{E}[X^\nu]$.

The expected value and the variance of the COM-Poisson distribution are (Kimberly F. Sellers 2011) :

$$\mathbb{E}_\lambda[X] = \lambda\frac{\partial lnZ(\lambda,\nu)}{\partial\lambda} = \frac{\partial lnZ(\lambda,\nu)}{\partial ln\lambda} \approx \lambda^{\frac{1}{\nu}} - \frac{\nu-1}{2\nu}$$

$$Var[X] = \frac{\partial\mathbb{E}_\lambda[X]}{\partial ln\lambda} \approx \frac{1}{\nu}\lambda^{\frac{1}{\nu}}$$

After all, $\frac{\partial}{\partial\lambda} = \frac{\partial ln\lambda}{\partial\lambda}\frac{\partial}{\partial ln\lambda} = \frac{1}{\lambda}\frac{\partial}{\partial ln\lambda}$ and $\lambda\frac{\partial}{\partial\lambda} = \frac{\partial}{\partial ln\lambda}$.

It is clear that $\nu \geq 0$ is a dispersion parameter such that $\nu = 1$ denotes equi-dispersion, $\nu > 1$ denotes under-dispersion and $\nu < 1$ denotes overdispersion.

The COM-Poisson distribution is a generalization of well-known distributions:

1. If $\nu = 1$ then $X \sim Poisson(\lambda)$
2. If $\nu = 0$ and $0 < \lambda < 1$, then $X \sim Geom(1 - \lambda)$
3. If $\nu \rightarrow \infty$ then $X \sim Bernoulli(\frac{\lambda}{1+\lambda})$

The bivariate Conway-Maxwell Poisson distribution

Let us consider two random variables $X$and Y denoting the goals achieved by the home and the away team in a football game, which follow univariate COM-Poisson distribution of mass functions,

$$P(X = x)|\lambda_1, \nu_1) = \frac{\lambda_1^x}{(x!)^{\nu_1}} \frac{1}{Z(\lambda_1, \nu_1)}, \quad x \in \mathbb{N}, \;\; \nu_1 \in \mathbb{R}_+, \;\; \lambda_1 \in \mathbb{R}_+^*,$$

$$P(Y = y)|\lambda_2, \nu_2) = \frac{\lambda_2^y}{(y!)^{\nu_2}} \frac{1}{Z(\lambda_2, \nu_2)}, \quad x \in \mathbb{N}, \;\; \nu_2 \in \mathbb{R}_+, \;\; \lambda_2 \in \mathbb{R}_+^*$$

where λ_1,λ_2 the respective scoring rates of the two teams.
The couple (X, Y) follows the bivariate COM-Poisson distribution if and only if its mass function is,

$$P(X = x, Y = y|\lambda_1, \nu_1, \lambda_2, \nu_2) = \frac{\lambda_1^x}{(x!)^{\nu_1}} \frac{\lambda_2^y}{(y!)^{\nu_2}} \frac{1}{Z(\lambda_1, \nu_1)} \frac{1}{Z(\lambda_2, \nu_2)} \;,$$

where $x, y \in \mathbb{N}$, $\nu_1, \nu_2 \in \mathbb{R}_+$, $\lambda_1, \lambda_2 \in \mathbb{R}_+^*$ under the conditions

$$log\lambda_1 = \boldsymbol{\beta_1} w$$

$$log\lambda_2 = \boldsymbol{\beta_2} w + \boldsymbol{\eta} x$$

where $w \in \mathbb{R}^p$ is the vector of explanatory variables and $\boldsymbol{\beta_1}, \boldsymbol{\beta_2} \in \mathbb{R}^p$ are the vectors of coefficients.

From the last condition above, we notice that

$$P(Y = y|\lambda_2, v_2) = P(Y = y|X = x).$$

As a result, it is,

$$P(X = x, Y = y)|\lambda_1, v_1, \lambda_2, v_2) = P(X = x|\lambda_1, v_1) \cdot P(Y = y|X = x)$$

It is clear that $\boldsymbol{\eta}$ is a measure of dependence between the two random variables, which is actually introduced through the dependence of the model parameters. When $\boldsymbol{\eta} = 0$ the variables X and Y are independent. After all, the covariance of X and Y in the bivariate COM-Poisson model is expressed as,

$$COV(X, Y) = \mathbb{E}_{\lambda_1}[X]\mathbb{E}_{\lambda_2}[Y](e^{\boldsymbol{\eta}} - 1)$$

Estimation

The estimation of the parameters β_1, β_2, η takes place through the maximum likelihood estimation. The log-likelihood of the bivariate COM-Poisson distribution is expressed as,

$$\ell = \sum_{i=1}^{n} \{x_i log\lambda_1 + y_i log\lambda_2 - \widehat{v_1}\log(x_i!) - \widehat{v_2}\log(y_i!) - log\sum_{x=0}^{\infty}\left[\frac{\lambda_1^{x_i}}{(x_i!)^{\widehat{v_1}}}\right] - log\sum_{y=0}^{\infty}\left[\frac{\lambda_1^{y_i}}{(y_i!)^{\widehat{v_2}}}\right]\} =$$

$$\sum_{i=1}^{n} \{x\boldsymbol{\beta_1}w + y(\boldsymbol{\beta_1}w + \eta x) - \widehat{v_1}\log(x!) - \widehat{v_2}\log(y!) - log\sum_{x=0}^{\infty}\left[\frac{e^{x\boldsymbol{\beta_1}w}}{(x!)^{\widehat{v_1}}}\right] - log\sum_{y=0}^{\infty}\left[\frac{e^{y\boldsymbol{\beta_1}w+\eta xy}}{(y!)^{\widehat{v_2}}}\right]\}$$

4.3. Models with inflation

Many statistical models that are used to predict football outcomes often underestimate some particular scores such as 0-0, 1-1 etc. In order to deal with this underestimation, it is necessary to inflate the probability of these scores. Inflated models are a good choice to deal with this problem.

4.3.1. Diagonal inflated bivariate Poisson model

The bivariate Poisson model (2.2.2) can explain quite properly a football game and its probable results. However, there is an underestimation in the "draw" results such as 0-0, 1-1, 2-2, 3-3, 4-4 etc. As a remedy to this occurrence, we may consider the diagonal inflated bivariate Poisson model. The latter is an extension of the simple zero-inflated model which allows for an excess only in (0,0) cell.

Considering that the starting model is the bivariate Poisson model, a diagonal inflated model is expressed as,

$$f_{IBP}(x,y) = \begin{cases} (1-p)f_{BP}(x,y|\lambda_1,\lambda_2,\lambda_3)\ , & x \neq y \\ (1-p)f_{BP}(x,y|\lambda_1,\lambda_2,\lambda_3) + pf_D(x;\theta)\ , & x = y \end{cases}$$

where D is a discrete distribution defined on the set $\{0,1,2,,\dots\}$ with parameter θ and $p \in (0,1)$.

We notice that if $p = 0$ we have the simple bivariate Poisson model.

The distribution $D(x;\theta)$

The distribution D that we mentioned above could be Poisson, geometric or other simple discrete distributions denoted by $D(m)$. As $D(m)$ we consider the distribution with the following probability function:

$$f(x|\theta, m) = \begin{cases} \theta_x, & x = 0,1,\dots,m \\ 0, & x \neq 0,1,\dots,m \end{cases}$$

where $\sum_{x=0}^{m} \theta_x = 1$.

We notice that if m = 0 we have a zero-inflated model that inflates only the 0-0 score. The geometric distribution might be of great interest as it decays quickly. After all, in football the most frequent draw results are 0-0 and 1-1 and, additionally, the more goals a draw outcome has, the less probable it is.

The marginal distributions

The marginal distributions of X and Y of the diagonal inflated bivariate Poisson model are not Poisson distributions, but mixtures of distributions:

$$f_{IBP}(x) = (1-p)f_{Poisson}(x|\lambda_1+\lambda_3) + pf_D(x|\theta)$$

$$f_{IBP}(y) = (1-p)f_{Poisson}(y|\lambda_2+\lambda_3) + pf_D(y|\theta)$$

As a result, the marginal means are:

$$\mathbb{E}[X] = (1-p)(\lambda_1+\lambda_3) + p\mathbb{E}_D[X]$$

and

$$\mathbb{E}[Y] = (1-p)(\lambda_2+\lambda_3) + p\mathbb{E}_D[Y]$$

where $\mathbb{E}_D[X]$ denotes the expected value of the distribution D.

As for the variance, we have:

$$Var[X] = (1-p)\{(\lambda_1+\lambda_3)^2 + (\lambda_1+\lambda_3)\} + p\mathbb{E}_D[X^2] - \{(1-p)(\lambda_1+\lambda_3) + pE_D[X]\}^2$$

and

$$Var[Y] = (1-p)\{(\lambda_2+\lambda_3)^2 + (\lambda_2+\lambda_3)\} + p\mathbb{E}_D[Y^2] - \{(1-p)(\lambda_2+\lambda_3) + pE_D[Y]\}^2$$

Since the marginal distributions are not Poisson distributions, they can be either under-dispersed or over-dispersed. It depends on the distribution D.

Correlation

In general, in the simple bivariate Poisson model, it is $\mathbb{E}_{BP}[XY] = \lambda_3 + (\lambda_1+\lambda_3)(\lambda_2+\lambda_3)$*. So, in the case of the respective inflated model we have,*

$$\begin{aligned} COV_{IBP}(X,Y) &= (1-p)\{\lambda_3 + (\lambda_1+\lambda_3)(\lambda_2+\lambda_3)\} + p\mathbb{E}_D(X^2) \\ &- (1-p)^2(\lambda_1+\lambda_3)(\lambda_2+\lambda_3) \\ &- (1-p)p\mathbb{E}_D(X)(\lambda_1+\lambda_2+2\lambda_3) - p^2\{\mathbb{E}_D[X]\}^2 \end{aligned}$$

We note that the covariance can either positive or negative depending on the choice of distribution D.

We conclude that, except for inflating the draw results, the diagonal inflated bivariate Poisson model also allows for over-dispersion as well as negative correlation in contrast with the simple bivariate Poisson model. These characteristics are necessary when modeling football results. However, the inflated model may sometimes be more difficult in computations than the simple Poisson model.

Estimation

Similarly to the simple bivariate Poisson distribution the estimation of the parameters will take place through the EM algorithm. In the diagonal inflated case of the bivariate Poisson model, the complete data log-likelihood takes the form,

$$L(\varphi, p, \theta) = \sum_{i=1}^{n} u_i\{\log(p) + log f_D(x_i; \theta)\}$$
$$+ \sum_{i=1}^{n} (1 - u_i)\{\log(1 - p)$$
$$- \sum_{i=1}^{n}\sum_{k=1}^{3} \lambda_{ki} + \sum_{i=1}^{n}\sum_{k=1}^{3} x_{ki}\log(\lambda_{ki}) - \sum_{i=1}^{n}\sum_{k=1}^{3} \log(x_{ki}!)\},$$

where u_i take values 1 or 0 depending on whether the observation comes from the inflation or the basic component. At the E-step u_i have to be estimated through their conditional expectations.

The EM algorithm for the diagonal inflated model is expressed as follows:

E-step: (a) We calculate the conditional expected values of the latent binary variable V_i, $i = 1, \dots, n$ by using the current parameter values of k iteration $(\varphi^{(k)}, \lambda_{1i}^{(k)}, \lambda_{2i}^{(k)}, \lambda_{3i}^{(k)}, p^{(k)}, \theta^{(k)})$:

$$u_i = \mathbb{E}\left[V_i \middle| X = X_i, Y = Y_i, \varphi^{(k)}, p^{(k)}, \theta^{(k)}\right)$$
$$= \begin{cases} \dfrac{p^{(k)} f_D(x_i|\theta^{(k)})}{p^{(k)} f_D\left(x_i\middle|\theta^{(k)}\right) + (1 - p^{(k)}) f_{BP}(x_i, y_i|\lambda_{1i}^{(k)}, \lambda_{2i}^{(k)}, \lambda_{3i}^{(k)})} & , x_i = y_i \\ 0 & , x_i \neq \mathrm{y_i} \end{cases}$$

where f_D the mass function of the inflation distribution with parameter vecror θ.

(b) Similarly to the occasion of the simple bivariate Poisson model, for $i = 1, \dots, n$ we calculate s_i.

M-step: We update the estimates:

$$p^{(k+1)} = \frac{1}{n}\sum_{i=1}^{n} u_i,$$
$$\beta_1^{(k+1)} = \hat{\beta}_{\tilde{u}}(x - s, W_1),$$
$$\beta_2^{(k+1)} = \hat{\beta}_{\tilde{u}}(y - s, W_2),$$
$$\beta_3^{(k+1)} = \hat{\beta}_{\tilde{u}}(s, W_3),$$
$$\theta^{(k+1)} = \hat{\theta}_{u,D},$$
$$\lambda_{ki}^{(k+1)} = \exp(W_{ki}^T\, \hat{\beta}_k^{(k+1)})\ , k = 1,2,3$$

where $x, y, s, u, \tilde{u} = 1 - u$ are $n \times 1$ vectors, $\hat{\beta}_u(x, W)$ are the weighted maximum likelihood estimates β of a Poisson regression model with response x and data matrix W, and $\hat{\theta}_{u,D}(x, W)$ are the weighted maximum likelihood estimates of θ for the distribution $D(x; \theta)$.

For specific choices of the inflation distribution that are used in the application of this dissertation :

- Geometric distribution

The parameter θ is updated by,

$$\theta^{(k+1)} = \frac{\sum_{i=1}^{n} u_i}{\sum_{i=1}^{n} u_i x_i + \sum_{i=1}^{n} u_i}$$

- Discrete distribution with $j = J$

The model parameters of the general occasion are given by,

$$\theta_j = \left(\sum_{i=1}^{n} u_i\right)^{-1} \sum_{i=1}^{n} I(X_i = Y_i = j)u_i \ \ , j = 1, \dots, J$$

$$\theta_0 = 1 - \sum_{j=1}^{J} \theta_j$$

where $I(x)$ indicator function. In the case of the inflation in the up to (1,1) cell we put $J = 1$.

4.3.2. Dixon and Coles model

Dixon and Coles model is another type of inflated model. In contrast with the case of the diagonal inflated bivariate Poisson model which inflates the probability of the draw results, the Dixon and Coles model accounts for the excessive number of particular scores. In other words, there is inflation on the probability of the specific outcomes 0-0, 1-0, 0-1, 1-1 which are frequent football results.

Considering $X \sim Poisson(\lambda_1)$ and $Y \sim Poisson(\lambda_2)$ the Dixon and Coles mass function of X, Y is expressed as,

$$f_{DC}(x, y) = P(X = x, Y = y) = \tau_{\lambda_1 \lambda_2}(x, y) \frac{\lambda_1^x \exp(-\lambda_1)}{x!} \frac{\lambda_2^y \exp(-\lambda_2)}{y!}$$

where λ_1, λ_2 are the scoring rates of the home and the away team respectively and τ is a function that moves the probability of certain scores as follows:

$$\tau_{\lambda_1\lambda_2}(x,y) = \begin{cases} 1-\lambda_1\lambda_2\rho, & if\ x=y=0 \\ 1+\lambda_1\rho, & if\ x=0, y=1 \\ 1+\lambda_2\rho, & if\ x=1, y=0 \\ 1-\rho, & if\ x=y=1 \\ 1, & otherwise \end{cases}$$

where ρ is a dependence parameter which satisfies the constraint:

$$\max\left(-\frac{1}{\lambda_1}, -\frac{1}{\lambda_2}\right) \leq \rho \leq \min\left(\frac{1}{\lambda_1\lambda_2}, 1\right)$$

If $\rho = 0$ then the two random variables X, Y are independent to each other.

The Dixon and Coles marginal distributions of X and Y are still Poisson with parameters λ_1 and λ_2 respectively.

Model Inference

Considering that we have n teams with attack parameters $\{a_1, \ldots, a_n\}$ and defense parameters $\{d_1, \ldots, d_n\}$ as well as a home parameter h, we want to estimate λ_1, λ_2 of the home and the away team. To prevent the model from being over-parameterized we have the following constraints,

$$n^{-1}\sum_{i=1}^{n} a_i = 1 \quad and \quad n^{-1}\sum_{i=1}^{n} d_i = 1$$

The basic tool of inference is the likelihood function. For N matches and score (x_k, y_k) in the kth match , $k = 1, \ldots, N$, the likelihood is expressed as,

$$L(\alpha_i, d_i, \rho, \gamma; i = 1, \ldots, n) \\ = \prod_{k=1}^{N} \tau_{\lambda_{1k}\lambda_{2k}}(x_k, y_k) \frac{\lambda_1^{x_k} \exp(-\lambda_{1k})}{x_k!} \frac{\lambda_2^{y_k} \exp(-\lambda_{2k})}{y_k!}$$

where

$$\lambda_{1k} = a_i(k)d_j(k)h,$$

$$\lambda_{2k} = a_j(k)d_i(k)$$

and $i(k)$ and $j(k)$ denote respectively the indices of the home and the away team playing in the kth match.
Despite the high dimensionality of the model, the maximization of the likelihood can be carried out straightforwardly through direct numerical computations.

4.4. Dynamic Models

All the models that we mentioned in the previous paragraphs are quite easy to use and they assume static team parameters. In other words, a team's performance determined by attack and defense abilities, remains unchanged across time. Although this makes our modeling and estimation easy, it sometimes contradicts the reality. That is a team's performance tends to be dynamic and changes across years, months or even weeks. Many factors may affect this performance such as roster changing, injuries, coaching staff changing, economic situations etc. For example, if an excellent scorer leaves a team, the offensive strength will certainly decrease. In the next paragraphs we will present dynamic extensions of some bivariate models that are already mentioned in the previous paragraphs.

4.4.1. Dixon and Coles dynamic model

We have already presented the Dixon and Coles bivariate model in paragraph **4.3.2**, for which we have the likelihood,

$$L(a_i, d_i, \rho, h; i = 1, \dots, n) = \\ = \prod_{k=1}^{N} \tau_{\lambda_{1k}\lambda_{2k}}(x_k, y_k) \exp(-\lambda_{1k})\, \lambda_{1k}^{x_k} \exp(-\lambda_{2k})\, \lambda_{2k}^{y_k}$$

where N the number of matches, $\lambda_{1k}, \lambda_{2k}$ the scoring rates of the two opponents in the k^{th} match and ρ the dependence parameter. The parameters $\lambda_{1k}, \lambda_{2k}$ depend on $a_i, d_{i,} h$ which are the attack parameters of the i^{th} team, the defense parameters of the i^{th} team and the home effect parameter respectively.

Since the parameters a_i, d_i remain static over time, the model written above can be enhanced by introducing a 'pseudo-likelihood' for each time point t. So it is,

$$L(a_i, d_i, \rho, h; i = 1, \dots, n) = \\ = \prod_{k \in A_t} \{\tau_{\lambda_{1k}\lambda_{2k}}(x_k, y_k) \exp(-\lambda_{1k}) \lambda_{1k}^{x_k} \exp(-\lambda_{2k})\, \lambda_{2k}^{y_k}\}^{\varphi(t-t_k)}$$

where t_k is the time that match k occurs, $A_t = \{k: t_k < t\}$ and φ is a non-increasing function of time. As for $\lambda_{1k}, \lambda_{2k}$ we have (similarly to the non-dynamic model),

$$\lambda_{1k} = a_i(k) d_j(k) h,$$

$$\lambda_{2k} = a_j(k) d_i(k)$$

It is clear that the parameters a_i, d_i, ρ, h are themselves time-dependent. Maximizing the equation above at time t, we estimate the parameters only up to time t and that is how the model reflects on changes in teams' performance.

Weighting function φ

The choice of the function φ depends on the way we want the weight of the historical data to decrease over time. One choice is,

$$\varphi(t) = \begin{cases} 1 & , t \leq t_0 \\ 0 & , t > t_0 \end{cases}$$

where all the results within the last time units since t_0 will be given equal weight in the inference whereas the results before t_0 won't be taken into consideration.

Another choice of the function φ could be,

$$\varphi(t) = \exp(-\xi t),$$

where the effect of all the previous results decreases exponentially over time according to the nonnegative parameter ξ. It is clear that if $\xi = 0$ then we end up with the initial static form. On the other hand, if ξ take large values, then there will be more weight to the most recent results. This last choice is the one that Dixon and Coles dynamic model uses.

Quite often, our basic aim is to predict the winner of a football match and not the exact score. It is remarkable that the probability of a home win, an away win and a draw in the k^{th} match are respectively estimated as,

$$p_k^H = \sum_{i>j} P(X = i, Y = j)$$

$$p_k^A = \sum_{i<j} P(X = i, Y = j)$$

$$p_k^D = \sum_{i=j} P(X = i, Y = j)$$

Now we define,

$$S(\xi) = \sum_{k=1}^{N} (\delta_k^H log\, p_k^H + \delta_k^A log p_k^A + \delta_k^D log p_k^D)$$

where $\delta_k^H, \delta_k^A, \delta_k^D$ take values 0 or 1 depending on the outcome we had in the k^{th} game. For instance, if the home team wins, then $\delta_k^H = 1$, $\delta_k^A = 0$ and $\delta_k^D = 0$. The probabilities p_k^H, p_k^A, p_k^D are the maximum likelihood estimates of $L(a_i, d_i, \rho, h, \xi; i = 1, \dots, n)$ and ξ is a weighting parameter. The parameter ξ plays an important role in the predictive capability of our model. Before defining the function S, the optimal choice of ξ wasn't feasible since the equation of our 'pseudo-likelihood' contained a sequence of dependent likelihoods. Therefore, our aim is to find the value of ξ that maximizes the function S.

4.4.2. Koopman and Lit model

All the statistic models that are used to predict football outcomes can be extended to dynamic. Koopman and Lit model is an extension with dynamic approach of the bivariate Poisson model that we presented in **4.2.2** paragraph. In this model the result of the outcome of the i^{th} football match is taken as the pair (X, Y) with probability density function

$$f_{BP}(x, y; \lambda_1, \lambda_2, \lambda_3) =$$

$$= \exp\{-(\lambda_1 + \lambda_2 + \lambda_3)\} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_{k=0}^{\min(x,y)} \binom{x}{k} \binom{y}{k} k! \left(\frac{\lambda_3}{\lambda_1 \lambda_2}\right)^k$$

with

$$\mathbb{E}[X] = Var[X] = \lambda_1 + \lambda_3,$$

$$\mathbb{E}[Y] = Var[Y] = \lambda_2 + \lambda_3$$

$$COV(X, Y) = \lambda_3$$

Dynamic specification

The scoring rate of the two opponent teams in a football match is determined by $\lambda_1, \lambda_2, \lambda_3$. Each team in a championship has its own scoring rate. In the dynamic case, we consider these rates to change over time since the performance of teams will change over time.

The scoring intensity of the team i when playing against the team j is considered to depend on the attack ability of the team i and the defense ability of team j. The home advantage is also included in our model, so considering that i is the home team and j the away team in week t we have for $i, j = 1, \dots, N$, $i \neq j$,

$$\lambda_{1(i,j)t} = \exp(home + att_{it} + def_{jt})$$

$$\lambda_{2(i,j)t} = \exp(att_{jt} + def_{it})$$

The attack and defense strengths of the teams in a championship change over time since the teams' compositions and performances are not the same over time. As a result, we consider the attack and defense parameters to be auto-regressive processes. We have,

$$att_{i,t} = \mu_{att,i} + \varphi_{att,i} att_{i,t-1} + \eta_{att,it}$$

$$def_{i,t} = \mu_{def,i} + \varphi_{def,i} def_{i,t-1} + \eta_{def,it}$$

where $\mu_{att,i}$ and $\mu_{def,i}$ are unknown constants, $\varphi_{att,i}$ and $\varphi_{def,i}$ are auto-regressive coefficients and $\eta_{att,it}$ and $\eta_{def,it}$ are normally distributed error terms which are independent of each other for all $i = 1, \dots, N$ and $t = 1, \dots, n$.

The dynamic processes are considered to be stationary, so $|\varphi_{att,i}| < 1$ and $|\varphi_{def,i}| < 1$ for $i = 1, \dots, N$. We also have that,

$$\eta_{att,it} \sim NID(0, \sigma^2_{att,i})$$

$$\eta_{def,it} \sim NID(0, \sigma^2_{def,i})$$

where $NID(a, b)$ is normal independent distribution with mean a and variance b.

The initial conditions for the auto-regressive processes $att_{i,t}, def_{i,t}$ are based on means and variances of their unconditional distributions which are given by,

$$\mathbb{E}[att_{i,t}] = \frac{\mu_{att,i}}{1 - \varphi_{att,i}} \quad , Var[att_{i,t}] = \frac{\sigma^2_{att,i}}{\left(1 - \varphi_{att,i}\right)^2}$$

and

$$\mathbb{E}[def_{i,t}] = \frac{\mu_{def,i}}{1 - \varphi_{def,i}} \quad , Var[def_{i,t}] = \frac{\sigma^2_{def,i}}{\left(1 - \varphi_{def,i}\right)^2}$$

Estimation

Considering J teams, we have $J/2$ match results for each week t. A specific match result is denoted by (X_{it}, Y_{jt}) with $i \neq j$ and $i, j \in \{1, \dots, J\}$. The numbers of goals scored by all teams in week t are collected in the $J \times 1$ observation vector y_t. We also assume the state vector z_t which contains the strengths of attack and defense of all J teams at time t, $\left(att_{1t}, \dots, att_{Jt}, def_{1t}, \dots, def_{Jt}\right)^{\mathrm{T}}$ with,

$$z_t = \mu + \Phi z_{t-1} + \eta_t$$

where μ is a constant $2J \times 1$ vector, Φ is the auto-regressive $2J \times 2J$ coefficient matrix and $\eta_t \sim N(0, H)$ is the $2J \times 1$ disturbance vector. Let $\varphi = diag\Phi$ and $h = diagH$. The observation density of y_t for a given realization of z_t is given by

$$p(y_t | z_t; \varphi, h, home, \lambda_3) = \prod_{k=1}^{\frac{J}{2}} f_{BP}(\lambda_{1,i,j,t}, \lambda_{2,i,j,t}, \lambda_3)$$

where f_{BP} the density of the bivariate Poisson distribution, index k represents the kth match between home team i and visiting team j and $\lambda_{1,i,j,t} = \exp\{home + w_{ij} z_t\}$, $\lambda_{2,i,j,t} = \exp\{w_{ji} z_t\}, i \neq j$. The vector w_{ij} selects the appropriate a_{it}, β_{jt} elements from z_t.

The joint density (y, z) is expressed as,

$$p(y, z; \varphi, h, home, \lambda_3) = p(y | z; \varphi, h, home, \lambda_3) \cdot p(z; \varphi, h, home, \lambda_3)$$

where

$$p(z; \varphi, h, home, \lambda_3) = p(z_1; \varphi, h, home, \lambda_3) \prod_{t=2}^{n} p(z_t | z_1, \dots, z_{t-1}; \varphi, h, home, \lambda_3)$$

Therefore the likelihood function of y is,

$$l(\psi) = p(y; \psi) = \int p(y, z; \psi) dz = \int p(y | z; \psi) p(z; \psi) dz$$

with $\psi = (\varphi, h, home, \lambda_3)$

An analytical solution to evaluate this integral is not feasible, so the maximum likelihood estimation is carried out through numerical evaluation methods.

Chapter 5

Application

In this chapter, four models will be used in terms of an application over football analysis and prediction. Initially, the aim of the application will be presented along with the data. Subsequently, the models' fitting will take place along with comparison of the models. At the end, prediction on a playoff match will be carried out. The R-code of the procedure as well as the whole dataset will be given in the Appendix.

5.1. Analyzing the Greek Superleague

The application that follows, concerns the Greek Superleague. Our basic aim is to analyze the teams' performance by estimating the "expected goals" for each team in every match of the season 2019-2020 and the regular season 2020-2021. The analysis will take place through four models: the bivariate Poisson model, the bivariate Poisson model with geometric diagonal inflation, the bivariate Poisson model with inflation at scores $0-0$ and $1-1$, and the diagonal inflated Double Poisson model.

5.1.1. Model specification

The basic aim of a statistician when using a model, is the estimation of the parameters of the model.

The bivariate Poisson models that were presented in **Chapter 4**, are said to use the number of goals that a team succeeds or concedes as covariates for the estimation of the model parameters. However, as it is pointed out by Wheatcroft (2020), the match statistics such as shots and corner kicks might be more informative than goals in terms of making match predictions.

Covariates for scoring rates $\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}$

In our application, the predictors that will be used for the scoring rates of the two opponents are:

1) **Overall Rating:** The overall team rating is a reasonable choice-predictor for the model as it depicts completely the quality of a team's performance in a football game (Hongyou Liu, 2015). The football performance analysts evaluate the performance of each player in a single match every 5 minutes. If a player makes a successful pass or cross or a good penetration in the opponent's area then the player will gain points. On the other hand, if a player makes a mistake then he will lose points. As a result, every 5-minutes, a total rating for each player is computed, which is positive or negative depending on whether the good actions are more than the bad ones or not. ***Table 1*** below shows the evaluation points for the match Asteras Tripolis vs Panathinaikos in the season 2020-2021. At the end of the match, each player has his total evaluation points and by calculating the sum of all players' points the team total evaluation points are obtained.

ΑΤΟΜΙΚΟΣ ΔΕΙΚΤΗΣ ΑΞΙΟΛΟΓΗΣΗΣ ανα 5-ΛΕΠΤΟ / PLAYERS PERFORMANCE per 5-MINUTE INTERVALS ΑΣΤΕΡΑΣ ΤΡΙΠ. 1 - 0 ΠΑΝΑΘΗΝΑΪΚΟΣ

		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	90+	points	mins	index	1st half	2nd half
1	Ν.ΠΑΠΑΔΟΠΟΥΛΟΣ	3.0			8.0	1.0	2.0				2.0							3.0			19.0	97'	0.196		
27	Γ.ΚΩΤΣΙΡΑΣ	3.0		1.0	1.0			1.0	-1.5	-0.5	2.0	1.0		2.0	2.0	-1.0	-1.0	-3.0	1.0	1.0	8.0	97'	0.082		
29	Φ.ΑΛΒΑΡΕΖ		0.5	1.0		2.0		-1.5					-1.0								1.0	71'	0.014		
4	Τ.ΠΑΣΑΛΙΔΗΣ		1.0		1.0		3.0		2.0	1.0	1.0	1.0	-1.0			1.0	1.0	1.0	1.0		13.0	97'	0.134		
15	Ν.ΣΟΥΑΡΕΘ		-0.5			1.5			4.0	3.0	1.0			1.0				1.0	1.0		13.0	97'	0.134		
12	Χ.ΜΟΥΝΑΦΟ	1.5	1.0	1.5	1.5	1.0	2.0	1.5	-0.5	1.0		3.0									13.5	72'	0.188		
14	Φ.ΜΠΕΛΟΚ					-3.0	0.5	2.0	1.0	1.0											1.5	56'	0.027		
23	Α.ΡΙΕΡΑ		3.0	-1.5				3.0	-1.0	1.0			-2.0			15.0	0.5	-0.5	1.0	1.0	19.5	97'	0.201		
97	Σ.ΑΛΙ ΜΠΑΜΠΑ	-1.0		1.0		-0.5	1.0					-1.0									-0.5	56'	-0.009		
11	Φ.ΡΙΤΖΙΣ ΚΡΕΣΠΙ	9.0	-0.5	-0.5	1.0				2.0	-1.0	-1.0	3.0				1.0	1.0		0.5	3.0	17.5	93'	0.188		
9	Χ.ΜΠΑΡΑΛΕΣ	1.0	1.5	-1.0	-2.0		2.5	3.0	1.0	-2.5	-0.5	1.0		1.0					2.0	3.0	10.0	97'	0.103		
99	Α.ΤΣΙΦΤΣΗΣ																				0.0	0'	0.000		
6	Χ.ΒΑΛΙΕΝΤΕ													2.0		1.0		1.0		1.5	5.5	41'	0.134		
3	Χ.ΤΑΣΟΥΛΗΣ																1.5	2.0			3.5	26'	0.135		
8	Μ.ΦΕΡΝΑΝΤΕΖ																	1.0		2.0	3.0	25'	0.120		
2	Ρ.ΓΚΑΡΣΙΑ																				0.0	4'	0.000		
18	Μ.ΟΡΟΣΚΟ																				0.0	0'	0.000		
7	Λ.ΤΙΛΙΚΑ													-2.0	0.5	12.0	-0.5	-1.0		2.0	4.0	41'	0.098		
	ΑΣΤΕΡΑΣ ΤΡΙΠ.	19	8	3	11	2	11	9	7	3	5	8	-4	4	3	31	3	5	7	16	131.5	1067'	0.138	c: Μ.ΡΑΣΤΑΒΑΤΣ	
	ευστοχες/τελικες αξιολογ.	2 / 2		29	2 / 2		53	2 / 3		72	2 / 4		80	3 / 6		118	3 / 8		147.5		on target/total shots			evaluation points	

		5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	90+	points	mins	index	1st half	2nd half
1	Σ.ΔΙΟΥΔΗΣ		5.0	3.0											3.0				3.0		14.0	97'	0.144		
14	Φ.ΣΑΝΤΣΕΕΣ	-1.5					-2.5	1.0	0.5	1.0	0.5	-1.0	-1.0	-0.5	-1.0					0.5	-4.0	97'	-0.041		
24	Θ.ΜΑΥΡΟΜΜΑΤΗΣ	-1.0		-1.0	0.5	1.0		1.0	2.0	-0.5		2.0					1.0				5.0	86'	0.058		
4	Φ.ΒΕΛΕΘ	3.0	1.0	3.0	2.0	1.0			-1.0	4.5	-1.0			1.0			1.0	1.0	-1.0		13.5	97'	0.139		
44	Α.ΠΟΥΓΓΟΥΡΑΣ	1.0	1.0	4.0	3.0		0.5			1.0	1.0	1.0		0.5	1.0	-1.0	1.0	1.0	-2.5		12.5	97'	0.129		
21	Δ.ΚΟΥΡΜΠΕΛΗΣ	1.0		3.0	2.0		2.0	0.5	1.0		3.0	0.5	-1.0								12.0	70'	0.171		
20	Δ.ΣΕΡΠΕΖΗΣ	1.0			1.5	4.0		-1.0	1.0		-0.5										6.0	60'	0.100		
8	Γ.ΑΓΙΟΥΜΠ	1.5	-1.0	-0.5	2.0	-0.5	2.5	-0.5	-1.0	-3.5	4.0	1.0	1.0	1.0		-0.5	1.0	1.0			8.0	97'	0.082		
7	Α.ΣΑΒΙΕΡ	0.5		-2.0	2.0	-1.5	4.0		1.0		-1.0			2.5	-1.0						4.5	71'	0.063		
9	Φ.ΜΑΚΕΝΤΑ	2.0	-1.0	1.0	2.0	1.0		1.0	0.5								-1.0		0.5		6.0	97'	0.062		
10	.ΚΑΡΛΙΤΟΣ	-1.0	1.0	2.0	3.0		1.0	1.0		-1.0	2.0	-1.0		-3.0	3.0		1.0			1.0	9.0	97'	0.093		
15	Β.ΞΕΝΟΠΟΥΛΟΣ																				0.0	0'	0.000		
11	Α.ΧΑΤΖΗΓΙΟΒΑΝΗΣ																2.0	-0.5			1.5	26'	0.058		
18	Γ.ΜΠΟΥΖΟΥΚΗΣ												1.0	6.0	1.0		1.0	1.0		1.0	11.0	37'	0.297		
22	.ΑΪΤΟΡ																				0.0	0'	0.000		
37	Α.ΑΘΑΝΑΣΑΚΟΠΟΥΛΟΣ																				0.0	0'	0.000		
55	Σ.ΑΛΕΞΑΝΔΡΟΠΟΥΛΟΣ																-1.0			-1.0	-2.0	27'	-0.074		
47	Β.ΖΑΓΑΡΙΤΗΣ																			-1.0	-1.0	11'	-0.091		
	ΠΑΝΑΘΗΝΑΪΚΟΣ	7	6	13	23	6	15	3	4	2	10	3	-1	8	8	-2	6	6	0	1	96.0	1067'	0.111	c: Ν.ΠΟΓΙΑΤΟΣ	
	ευστοχες/τελικες αξιολογ.	1 / 1		25	4 / 6		69	5 / 7		77	5 / 7		89	5 / 7		103	5 / 7		118.0		on target/total shots			evaluation points	

Table 1: Evaluation Index from the analyst of Asteras Tripolis for the match: Asteras Tripolis vs Panathinaikos (Greek Superleague 2020-2021)

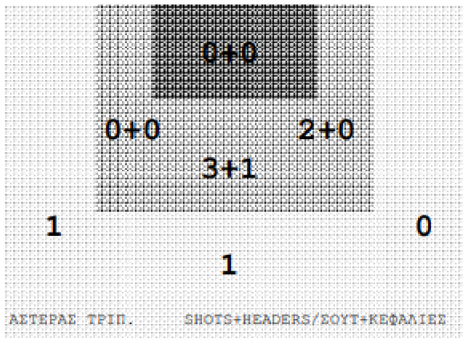
2) **Shots in the penalty and the goal box area:** The number shots made by a team play a crucial role in the scoring rate, especially when they are attempted at close range from the rival goalpost. These shots consist of the shots inside the penalty area and the shots inside the goal-box area. ***Table 2*** below presents these attempts from the same match.

ΑΣΤΕΡΑΣ ΤΡΙΠ. 1 - 0 ΠΑΝΑΘΗΝΑΪΚΟΣ

```
Referee: Α. ΣΙΔΗΡΟΠΟΥΛΟΣ        (ΔΩΔΕΚΑΝΗΣΟΥ)
Assist.: Π. ΚΩΣΤΑΡΑΣ            (ΑΙΤΩΛ/ΝΙΑΣ)
         Λ. ΔΗΜΗΤΡΙΑΔΗΣ         (ΜΑΚΕΔΟΝΙΑΣ)
```

5	ΣΟΥΤ ΕΝΤΟΣ ΠΕΡ / SHOTS INSIDE	1
2	ΣΟΥΤ ΕΚΤΟΣ ΠΕΡ / SHOTS OUTSIDE	4
1	ΚΕΦΑΛΙΕΣ / HEADERS	2
5	ΑΠΟΚΡΟΥΣΕΙΣ / SAVES	2
14	ΕΠΕΜΒΑΣΕΙΣ / CLEARANCES	16
16	ΚΛΕΨΙΜΑΤΑ / STEALS	16
14	ΦΑΟΥΛ / FOULS COM.	18
0	ΟΦ ΣΑΙΝΤ / OFFSIDE	1
1	ΚΟΡΝΕΡ / CORNERS	3
40	ΛΑΘΗ / ERRORS	50
6/23	ΓΕΜΙΣΜΑΤΑ / CROSSES	4/16
2	ΔΙΕΙΣΔΥΣΕΙΣ / PENETRATIONS	3

```
Σ.ΑΛΙ ΜΠΑΜΠΑ                ( 03') χαμενη ευκαιρια
Φ.ΜΠΕΛΟΚ                    ( 21') κιτρινη καρτα
        δοκαρι              ( 26')  .ΚΑΡΛΙΤΟΣ
        κιτρινη καρτα       ( 27') Φ.ΣΑΝΤΣΕΣ
        χαμενη ευκαιρια     ( 29')  .ΚΑΡΛΙΤΟΣ
        κιτρινη καρτα       ( 42') Γ.ΑΓΙΟΥΜΠ
Χ.ΜΠΑΡΑΛΕΣ                  ( 44') κιτρινη καρτα
Χ.ΒΑΛΙΕΝΤΕ / Φ.ΜΠΕΛΟΚ       ( 56') in / out αλλαγη
Λ.ΤΙΛΙΚΑ   / Σ.ΑΛΙ ΜΠΑΜ     ( 56') in / out αλλαγη
        αλλαγη in / out     ( 60') Γ.ΜΠΟΥΖΟΥΚ / Δ.ΣΕΡΠΕΖΗΣ
        αλλαγη in / out     ( 70') Σ.ΑΛΕΞΑΝΔΡ / Δ.ΚΟΥΡΜΠΕΛΗΣ
        αλλαγη in / out     ( 71') Α.ΧΑΤΖΗΓΙΟ / Α.ΣΑΒΙΕΡ
Χ.ΤΑΣΟΥΛΗΣ / Φ.ΑΛΒΑΡΕΖ      ( 71') in / out αλλαγη
Μ.ΦΕΡΝΑΝΤΕ / Χ.ΜΟΥΝΑΦΟ      ( 72') in / out αλλαγη
Α.ΡΙΕΡΑ                     ( 73') γκολ
        κιτρινη καρτα       ( 77') Φ.ΜΑΚΕΝΤΑ
Χ.ΤΑΣΟΥΛΗΣ                  ( 83') δοκαρι
Γ.ΚΩΤΣΙΡΑΣ                  ( 85') κιτρινη καρτα
        αλλαγη in / out     ( 86') Β.ΖΑΓΑΡΙΤΗ / Θ.ΜΑΥΡΟΜΜΑΤΗΣ
        κιτρινη καρτα       ( 88') Α.ΠΟΥΓΓΟΥΡΑΣ
Ρ.ΓΚΑΡΣΙΑ  / Φ.ΡΙΤΖΙΣ Κ     (90+3  in / out αλλαγη
```



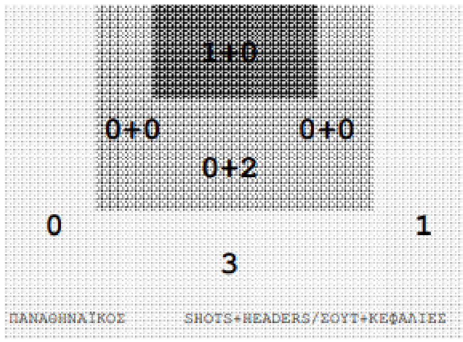


Table 2: Attemts made from the two opponents for the match: Asteras Tripolis vs Panathinaikos (Greek Superleague 2020-2021)

3) **Corner Kicks:** The number of the corner kicks gained during game shows a lot about the offensive strategy of the team. For instance, if a team usually attacks from the sides, then it will gain more corner kicks than a team which attacks through the central axis of the field. It is also worth mentioning that the number of the corner kicks describe in a way the dominance of a team against the opponent as it shows in a way how much time a team spends in the opponent's area.

Covariates for the dependence parameter λ_3

As it is mentioned, the parameter λ_3 concerns the level of interaction of the two opponents in a football game. The two teams interact with each other during the match which means that the scoring rate of the teams is affected a lot by the game conditions, such as the speed of the game. Using covariates on λ_3 helps us to have more insight regarding the type of influence. In the following application the dependence parameter λ_3 will be considered to be constant, which is the simplest approach.

5.1.2. Data

The data of the following application were provided by the sports analyst of Asteras Tripolis, Thodoris Tsilimigras. In every match, the final score $(g1, g2)$, the overall ratings of the two opponents $(rat1, rat2)$, the shots from the penalty area $(penbox1, penbox2)$, the shots from the goal box $(goalbox1, goalbox2)$ as well as the corner kicks $(corner1, corner2)$ constitute the dataset.

1	team1	team2	g1	g2	rat1	rat2	penbox1	penbox2	goalbox1	goalbox2	corner1	corner2
2	Aek	Xanthi	1	2	82	50	1	3	1	0	3	3
3	Aris	Ofi	1	1	89	70	8	2	1	0	7	1
4	Atromitos	Larisa	1	1	144	155	4	4	1	2	7	2
5	Olympiakc	Tripoli	1	0	167	92	8	3	1	0	4	1
6	Panionios	Volos	1	2	133	131	3	7	0	0	7	2
7	Paok	Panetolikc	2	1	173	92	3	3	1	0	6	2
8	Lamia	Panathinai	1	1	92	142	1	1	1	0	6	5
9	Larisa	Olympiakc	0	1	60	89	0	2	1	1	1	9
10	Tripoli	Aek	2	3	99	226	3	9	0	1	3	7
11	Volos	Aris	1	0	89	110	4	2	0	3	1	8
12	Panathinai	Ofi	1	3	84	114	3	8	0	0	1	1
13	Panetolikc	Xanthi	1	2	118	111	2	4	0	0	5	0
14	Paok	Panionios	2	1	123	55	4	1	1	0	9	2
15	Lamia	Atromitos	2	2	136	178	3	4	0	0	5	6
16	Aek	Lamia	2	0	142	51	6	1	0	0	10	1
17	Xanthi	Tripoli	2	1	119	92	2	4	1	0	3	4
18	Aris	Panathinai	4	0	202	119	5	2	5	1	3	3
19	Atromitos	Paok	2	3	133	160	2	2	1	1	2	5
20	Olympiakc	Volos	5	0	253	34	9	2	2	0	5	3
21	Ofi	Panetolikc	3	1	241	134	8	1	0	0	2	4
22	Panionios	Larisa	1	0	133	121	4	2	0	0	3	6
23	Larisa	Xanthi	3	0	201	87	2	1	0	0	7	0
24	Tripoli	Atromitos	2	1	217	184	8	1	0	0	6	6
25	Volos	Ofi	1	0	135	156	7	4	0	1	3	10
26	Panathinai	Olympiakc	1	1	69	124	3	2	1	0	2	5
27	Lamia	Panionios	1	1	155	117	4	1	0	0	4	1
28	Panetolikc	Aek	0	1	88	165	0	7	0	1	2	3
29	Paok	Aris	2	2	135	102	4	2	0	0	8	3
30	Aek	Paok	2	2	136	137	2	6	1	2	3	4
31	Xanti	Volos	3	1	260	150	8	4	0	0	4	3

Table 3: Part of the data set: Scores and match statistics for the games of Greek Superleague 2019-20 regular season

The teams that take part in this application are:

```
> sl=read.csv("data/sl.csv",stringsAsFactors=T)
> levels(sl[,2])
```

```
 [1] "Aek"          "Apollon"      "Aris"         "Atromitos"
 [5] "Giannena"     "Lamia"        "Larisa"       "Ofi"
 [9] "Olympiakos"   "Panathinaikos" "Panetolikos" "Panionios"
[13] "Paok"         "Tripoli"      "Volos"        "Xanthi"
```

Table 4: The teams-factors of the data in an alphabetical order

The quality of the selected predictors that are used in the application are evaluated through the R-output below:

```
> sign=glm(g2~rat2+penbox2+goalbox2+corner2,family="poisson",
      data=sl) ; summary(sign)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2023  -1.0456  -0.1455   0.4814   2.2553

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.520710   0.179090  -8.491  < 2e-16 ***
rat2         0.008484   0.001442   5.882 4.06e-09 ***
penbox2      0.086295   0.024767   3.484 0.000493 ***
goalbox2     0.192869   0.062014   3.110 0.001870 **
corner2     -0.047022   0.020301  -2.316 0.020546 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 524.42  on 421  degrees of freedom
Residual deviance: 365.05  on 417  degrees of freedom
```

Table 5: Summary of glm

The output presents the level of significance of the predictors in relation to the response variable $g2$ which denotes the goals achieved by a team during match. It is clear that the intercept as well as the overall rating and the attempts from the penalty area are highly significant. The lowest significance is obtained by the corner kicks that a team gains in a match. It is also worth mentioning that the corner kicks are negatively correlated with the goals scored by a team. This may lead to the conclusion that in the Greek Superleague, the attacking strategy shouldn't be based on gaining corner kicks.
In order to check the dependence between the selected covariates, a correlation matrix is obtained:

	Rating	PenaltyBox	GoalBox	Corner
Rating	1.0000000	0.5999163	0.3391532	0.3793011
PenaltyBox	0.5999163	1.0000000	0.1121999	0.2918752
GoalBox	0.3391532	0.1121999	1.0000000	0.2288395
Corner	0.3793011	0.2918752	0.2288395	1.0000000

Table 6: Correlation matrix

The level of correlation between any pair of the explanatory variables above is quite small in general terms, which implies that each of the variables can independently predict the value of the dependent variable.

5.1.3. Fitting the models

The analysis of the Greek Superleague 2019-20 and 2020-21 will take place through functions in R. The package that contains these functions is made by Karlis and Ntzoufras and it is available at http://www.stat-athens.aueb.gr/~jbn/papers/paper14.htm. It contains the EM algorithm for fitting the bivariate Poisson model and the diagonal inflated bivariate Poisson model, as well as some extra functions that the algorithm uses. The R-code is given in the Appendix.

- ***Fitting the bivariate Poisson model***

The function **lm.bp** applies the EM algorithm for fitting the bivariate Poisson model of the form $(x_i, y_i) \sim BP(\lambda_{1i}, \lambda_{2i}, \lambda_{3i})$ for $i = 1, \dots, n$ with $l_k = w_k \beta_k$, $k = 1,2,3$ where $l_k = log\lambda_k$. Its syntax is:

$$\boldsymbol{lm.bp(l1, l2, l1l2 = NULL, l3 = \sim 1, data, common.intercept = FALSE, zeroL3 = FALSE, maxit = 300, pres = 1e - 8)}$$

The input components $\boldsymbol{l1}$, $\boldsymbol{l2}$ and $\boldsymbol{l3}$ are of the form "$x \sim x_1 + \cdots + x_k$", "$y \sim y_1 + \cdots + y_k$" and "$z \sim z_1 + \cdots + z_p$" respectively, concerning the parameters of $log\lambda_1$, $log\lambda_2$ and $log\lambda_3$. The component $\boldsymbol{l1l2}$ concerns the common parameters of $log\lambda_1$ and $log\lambda_2$ (whether they exist) and the component **data** is the data frame which contains the variables. There are also two logical arguments: **common.intercept** and **zeroL3.** The first one refers to whether a common intercept on $log\lambda_1$ and $log\lambda_2$ is used and the second one refers to whether λ_3 is set equal to zero. Finally, the component **maxit** is associated with the maximum number of the EM steps that will take place and the argument **pres** is the precision that is used to terminate the EM algorithm. If the relative log-likelihood difference is lower than the value of the precision then the EM algorithm will terminate.

```
> biv=lm.bp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+
           penbox2+goalbox2+corner2,l1l2=NULL,data=sl)
> biv$coefficients
```

```
(l1):(Intercept)      (l1):corner1    (l1):goalbox1     (l1):penbox1
    -1.281071082      -0.030432260      0.100234116      0.024295455
      (l1):rat1  (l2):(Intercept)     (l2):corner2    (l2):goalbox2
     0.008715946      -1.705548672     -0.048838609      0.197741080
    (l2):penbox2         (l2):rat2 (l3):(Intercept)
     0.087018910       0.009189555     -2.709568924
```

```
> biv$parameters
  [1] 11

> biv$iterations
  [1] 56
```

```
> biv$lambda1
```

```
        1         2         3         4         5         6         7
0.5867651 0.6545679 0.9592680 1.4153269 0.7695189 1.2427348 0.5843424
        8         9        10        11        12        13        14
0.5024232 0.6462280 0.6449372 0.6026145 0.7003606 0.7516502 0.8394760
       15        16        17        18        19        20        21
0.8171303 0.8299869 2.7480759 0.9666780 3.2904659 2.5934030 0.8905061
       22        23        24        25        26        27        28
1.3585243 1.8627598 0.9746791 0.5669893 1.0463974 0.5627429 0.7782599
```

```
> biv$lambda2
```

```
      423       424       425       426       427       428       429
0.3225394 0.3917699 1.4401005 0.5231477 1.0097857 0.4982117 0.5724637
      430       431       432       433       434       435       436
0.3846402 2.7465120 0.7274970 0.9894295 0.7136002 0.2979643 0.9853445
      437       438       439       440       441       442       443
0.3015857 0.4929298 0.6792754 0.8979561 0.2552321 0.5585094 0.4903645
      444       445       446       447       448       449       450
```

After estimating the parameters λ_1, λ_2 and λ_3, the fitted values for the two responses x and y (which denote the goals achieved by the two teams) are obtained. The fitted values can be estimated as,

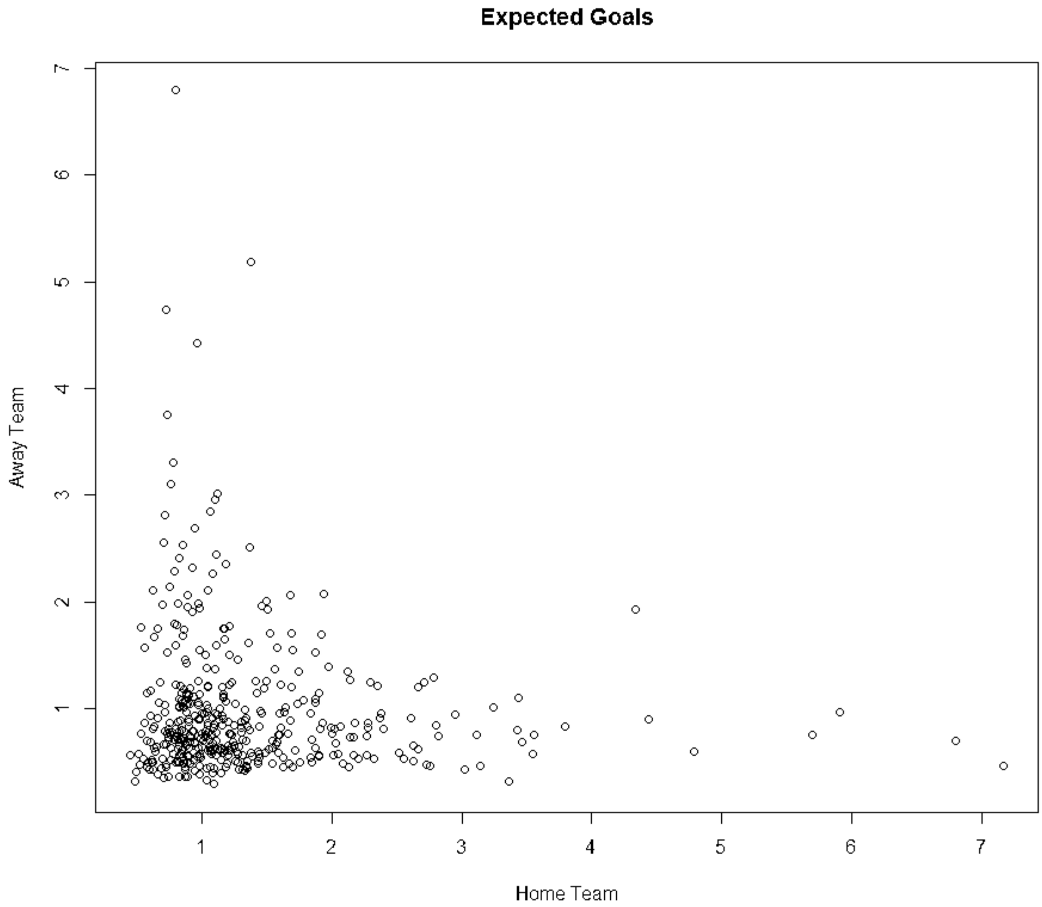
$$\hat{x} = \lambda_1 + \lambda_3$$
$$\hat{y} = \lambda_2 + \lambda_3$$

The fitted values $\widehat{x_i}, \widehat{y_i}$ denote the number of goals that each of the two teams deserved to have achieved in the i-th match (expected goals). These values arise by taking into account their performance in the i-th match (***Table 6***).

	x	y
1	0.6533306	0.3891049
2	0.7211334	0.4583354
3	1.0258335	1.5066660
4	1.4818924	0.5897132
5	0.8360844	1.0763512
6	1.3093003	0.5647772
7	0.6509079	0.6390292
8	0.5689887	0.4512057
9	0.7127935	2.8130774
10	0.7115027	0.7940625
11	0.6691800	1.0559950
12	0.7669261	0.7801657
13	0.8182157	0.3645298
14	0.9060415	1.0519100
15	0.8836958	0.3681512
16	0.8965524	0.5594953
17	2.8146414	0.7458409
18	1.0332435	0.9645216
19	3.3570314	0.3217976
20	2.6599685	0.6250749

Table 6: Expected goals obtained by the bivariate Poisson model

In many matches, a deviation is observed between the goals that a team achieved and the goals that should have succeeded. For instance, in the 16-th match of the regular season 2019-20 (Xanthi vs Asteras Tripolis) where the final score was $2-1$, the expected goals of Xanthi based on the match performance were 0.8965524. This leads to the remark that Xanthi was either lucky or too effective due to the fact that it took only few attempts to achieve goal. However, the final result was victory of the home team which is in accordance with the expectation $\hat{x}_{16} > \hat{y}_{16}$.



Graph: Plot of the home and away expected goals

The plot above depicts the relationship between the home expected goals and the away expected goals in the Greek Superleague 2019-20 and the regular season 2020-21. As it appears, in most games the performance of the two opponents is interwoven with about 1 goal for each team. After all, it is observed that in most matches, the levels of performance of the two opponents are similar. This may imply the existence of high competitiveness in the Greek Superleague.

- ***Fitting the diagonal inflated bivariate Poisson models***

The function **lm.dibp** contains the EM algorithm for fitting the diagonal inflated bivariate Poisson model of the form:

$$(x_i, y_i) \sim DIBP\left(\lambda_{1i}, \lambda_{2i}, \lambda_{3i,}, p, D(\theta)\right) \text{ for } i = 1, \dots, n$$

with $l_k = w_k \beta_k$, $k = 1,2,3$ where $l_k = log\lambda_k$.

Its syntax in R is:

$$lm.dibp(l1, l2, l1l2 = NULL, l3 = \sim 1, data, common.intercept = FALSE, zeroL3 = FALSE, distribution = \text{"discrete",jmax=2}, maxit = 300, pres = 1e-8)$$

The syntax of the diagonal inflated model above contains an extra input component compared with the bivariate Poisson model. That is the component **distribution** which refers to the discrete distribution that provokes inflation. The choices could be “poisson”, “geometric” or “discrete”. In the case of the last choice, the argument **jmax** is required, which shows up to which draw outcome there will be probability inflation.

A diagonal inflated model with geometric inflation and an inflated model with inflation in the outcomes $0-0$ and $1-1$ will be used for our application. After these attempts, the occasion where $\lambda_3 = 0$ will also be shown which lead to an inflated double Poisson model.

- For the model with **geometric inflation**:

```
> infg=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+
               goalbox2+corner2,l1l2=NULL,data=sl,distribution=
               “geometric”)
```

```
> infg$coefficients
```

```
(l1):(Intercept)      (l1):corner1     (l1):goalbox1      (l1):penbox1
   -1.279176e+00     -3.040164e-02      1.002587e-01      2.433551e-02
      (l1):rat1  (l2):(Intercept)      (l2):corner2     (l2):goalbox2
    8.707181e-03     -1.703233e+00     -4.882711e-02      1.976890e-01
    (l2):penbox2        (l2):rat2  (l3):(Intercept)                 p
    8.701068e-02      9.181056e-03     -2.721915e+00      1.680058e-07
           theta
    5.588838e-01
```

The fitted values of the responses x, y are expressed as,

$$\hat{x} = (1-p)(\lambda_1 + \lambda_3) \quad and\ \hat{y} = (1-p)(\lambda_2 + \lambda_3) \qquad , x \neq y$$
$$\hat{x} = (1-p)(\lambda_1 + \lambda_3) + p\mathbb{E}_D[x]\ and\ \hat{y} = (1-p)(\lambda_2 + \lambda_3) + p\mathbb{E}_D[x] \quad , x = y$$

	x	y
1	0.6532962	0.3889015
2	0.7214134	0.4581913
3	1.0260070	1.5071227
4	1.4823479	0.5896923
5	0.8360864	1.0767139
6	1.3093623	0.5647241
7	0.6508729	0.6388756
8	0.5688879	0.4510025
9	0.7127779	2.8132190
10	0.7115284	0.7941918
11	0.6691527	1.0564574
12	0.7668757	0.7803050
13	0.8183594	0.3642685
14	0.9060445	1.0519200
15	0.8838574	0.3679034
16	0.8966062	0.5594414

Table 7: Expected goals obtained by the bivariate Poisson with geometric inflation

- For the model with the **discrete inflation with $j = 1$** :

```
> inf1=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+
               penbox2+goalbox2+corner2,l1l2=NULL,data=
               sl,jmax=1)
```

```
> inf1$coefficients
```

```
(l1):(Intercept)      (l1):corner1     (l1):goalbox1      (l1):penbox1
   -1.2796688377     -0.0304057762      0.1002380657      0.0243202935
      (l1):rat1  (l2):(Intercept)      (l2):corner2     (l2):goalbox2
    0.0087097908     -1.7038724992     -0.0488136144      0.1977210394
    (l2):penbox2         (l2):rat2  (l3):(Intercept)                 p
    0.0870290428      0.0091829090     -2.7192886316      0.0001305199
          theta1
    1.0000000000
```

- Finally, the **inflated double Poisson** model will be obtained by putting $\lambda_3 = 0$ in the last model. After all, as it is mentioned, the dependence between the two opponents can be expressed by the inflated model even if $\lambda_3 = 0$.

```
> infdp=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+
                penbox2+goalbox2+corner2,l1l2=NULL,data=
                zeroL3=TRUE,jmax=1)
```

The fitted values $\hat{x}$ and $\hat{y}$ in the inflated double-Poisson occasion are:

$$\hat{x} = (1-p)\lambda_1 \quad and\ \hat{y} = (1-p)\lambda_2 \quad ,x \neq y$$
$$\hat{x} = (1-p)\lambda_1 + p\mathbb{E}_D[x]\ and\ \hat{y} = (1-p)\lambda_2 + p\mathbb{E}_D[x] \quad ,x = y$$

5.1.4. Model comparison

Four bivariate models were used for analyzing the Greek Superleague 2020-19 and 2020-21. The following matrix depicts a summary of this analysis.

	Parameters	AIC	BIC	Mix.Prop(p)
Bivariate Poisson	11	2081.151	2133.271	0.0000000
Inflated with Discrete(1)	13	2085.153	2146.749	0.0130500
Inflated with Geometric	13	2085.151	2146.747	0.0000168
Inflated Double-Poisson	12	2084.952	2141.810	0.0460200

Table 8: Comparison of the fitted-models

A considerable remark is that the bivariate Poisson model seems to be a preferable option due to the fact that the AIC and BIC values of this model are smaller than the others. Although the inflated bivariate Poisson models are generally considered to be better options when analyzing football matches, in the case of Greek Superleague there was no excess in draw outcomes. This makes the simple bivariate Poisson model a better fit to our data.

Finally, let us compare the bivariate Poisson model above (which uses match statistics as covariates) with the bivariate Poisson model whose explanatory variables are the goals that teams have succeeded and conceded so far. After all, many authors suggest the latter.

	Loglikelihood	AIC	BIC
Bivariate Poisson	-1029.576	2081.151	2133.271
Bivariate Poisson (goals as cov)	-1098.000	2269.030	2444.342

Table 9: Comparison of the fitted-bivariate Poisson model and the bivariate Poisson model that uses goals as covariates

It is clear that the model that uses the game ratings and statistics as covariates is proved to be a better option according to the table above. As a result, the model that will be used for the prediction that follows is the bivariate Poisson model which uses the match ratings and statistics as covariates.

5.2. Prediction

5.2.1. Predicting a playoff match

In a football game, the scoring rates of the two opponents λ_1, λ_2 are estimated through their game statistics and ratings. However, the match statistics are unknown before a match starts. As a result, in order to predict the outcome of an upcoming football match, the statistics of this match must be firstly estimated (Edward Wheatcroft 2020).
After analyzing the seasons 2019-20 and 2020-21 of the Greek Championship we will make a prediction for the first playoff match of the season 2020-21. The prediction will take place through the function **bivpois.table** (Karlis and Ntzoufras). Its syntax in R is:

$$bivpois.table(x, y, lambda = c(1,1,1))$$

This function returns a probability matrix (with $(x+1) \times (y+1)$ dimension) of the bivariate Poisson distribution using recursive relations. The components $\boldsymbol{x}$ and $\boldsymbol{y}$ show the values that will be evaluated. The cell ij in the matrix contains the probability $P(X = i-1, Y = j-1)$. It is reasonable that x and y must be at least 1. The component $\boldsymbol{lambda}$ is a vector that contains the values of the parameters $\lambda_1, \lambda_2, \lambda_3$.
The first match of the playoff period of the season 2020-21 was Asteras Tripolis vs Panathinaikos. By calculating the expected statistics of the two teams before the match, the scoring rates λ_1 and λ_2 can be obtained. The dependence parameter λ_3 is constant and equal to 0.00665655.

```
> l1=exp(-1.281071082+0.008715946*ratA+0.024295455*penboxA+
        0.100234116*goalboxA-0.030432260*cornerA);l1

[1] 1.163891

> l2=exp(-1.705548672+0.009189555*ratP+0.087018910*penboxP+
        0.197741080*goalboxP-0.048838609*97orner);l2
```

```
[1] 0.8765884

> l3=0.00665655;l3
```

By calling the function $bivpois.table(8{,}8, lambda(l1, l2, l3)$ the probabilities of all the outcomes up to $8-8$ are obtained.

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1.215967e-01	1.065903e-01	4.671790e-02	1.365079e-02	2.991531e-03
[2,]	1.415254e-01	1.321536e-01	6.146979e-02	1.899784e-02	4.390488e-03
[3,]	8.236008e-02	8.161660e-02	4.017052e-02	1.310159e-02	3.187326e-03
[4,]	3.195273e-02	3.349173e-02	1.739566e-02	5.974267e-03	1.527272e-03
[5,]	9.297378e-03	1.027692e-02	5.619013e-03	2.027837e-03	5.438147e-04
[6,]	2.164228e-03	2.516021e-03	1.444802e-03	5.468430e-04	1.535850e-04
[7,]	4.198210e-04	5.120731e-04	3.081788e-04	1.221066e-04	3.585953e-05
[8,]	6.980372e-05	8.913472e-05	5.611043e-05	2.323328e-05	7.123527e-06
[9,]	1.015549e-05	1.354871e-05	8.904969e-06	3.847004e-06	1.229693e-06

	[,6]	[,7]	[,8]	[,9]
[1,]	5.244682e-04	7.662379e-05	9.595361e-06	1.051398e-06
[2,]	8.095568e-04	1.240933e-04	1.626846e-05	1.862433e-06
[3,]	6.172456e-04	9.915982e-05	1.359753e-05	1.625294e-06
[4,]	3.101910e-04	5.216618e-05	7.475555e-06	9.322639e-07
[5,]	1.156731e-04	2.034094e-05	3.043301e-06	3.956670e-07
[6,]	3.416603e-05	6.274897e-06	9.792153e-07	1.326184e-07
[7,]	8.331503e-06	1.596263e-06	2.595653e-07	3.658924e-08
[8,]	1.726282e-06	3.446382e-07	5.833741e-08	8.551999e-09
[9,]	3.104232e-07	6.450403e-08	1.135493e-08	1.729607e-09

Table 10: Probability matrix for the scores of the playoff match: Asteras Tripolis vs Panathinaikos

By taking the sum of the elements of the matrix diagonal as well as the sum of the elements above and below the diagonal, the following probabilities are obtained:

$$Asteras\ Tripolis\ win\text{:}\ \ 42{,}4\%$$
$$Draw\text{:}\ 30\%$$
$$Panathinaikos\ win\text{:}\ 27{,}6\%$$

The actual final result in this match was $2-2$.

5.2.2. Betting odds

In betting companies, the bookmakers use betting odds to describe an upcoming match. By inversing the win-draw-lose probabilities of the match Asteras Tripolis vs Panathinaikos above, the following betting values-odds arise:

$$Asteras\ Tripolis\ win: 2{,}35$$
$$Draw: 3{,}33$$
$$Panathinaikos\ win: 3{,}62$$

Certainly, the betting odds of many other characteristics of the game (such as how many goals are going to be achieved in general) can also be obtained by inversing of the respective probabilities from the matrix above.
These betting odds above are usually reduced by bookmakers so that there is a gain for the companies. Actually, the relation between the betting odds o_i and the probabilities p_i of an event i is expressed as ,

$$p_i = \frac{1}{o_i + g}$$

where g is the gain of the bookmaker.
As a result, it easy to notice that the odds in practice also contain the market value information.

Conclusion

Sports analytics constitute a sector of statistics which is continually evolving while making the predictions of many sport events more and more effective. In the case of football, there have been many predictive models so far, each of which has its own specifications and properties. It is worth mentioning that sometimes, considering models with simpler structure than others may be preferable. Concerning the information which predictive models use, the in-game statistics and ratings are more informative than the goals that teams have been succeeded so far. These facts could be of great interest, as the companies associated with football, such as betting companies, can improve their approach on modeling and prediction, which will lead to increase of profits. More importantly, the teams themselves could assess various characteristics and make decisions in order to increase their chances for a successful outcome.

References

Kocherlakota S, Kocherlakota K (1992). “Bivariate Discrete distributions”. New York: Marcel Dekker

stat.auckland.ac.nz

Joel Liden (2016). “Bivariate models to predict football results”. U.U.D.M. Project Report

Martin Haugh (2016). “An introduction to copulas”. Quantitative risk management

Chandra R. Bhat, Naveen Eluru (2009). “A copula-based approach to accommodate residential self-selection effects in travel behavior modeling”. University of Texas at Austin

Roemer J. Janse, Tiny Hoekstra, Kitty J. Jager, Carmine Zoccali, Giovanni Tripepi, Friedo W. Dekker, Merel van Diepen (2021). “Conducting correlation analysis”. Clinical Kidney Journal

David Nettleton (2014). “Pearson Correlation”. Commercial Data Mining.

Theodosis Dimitrakos (2012). “Kendall Notes” . Notes, Mathematics Department of Samos

P. McCullagh, J.A. Nedler (1982). “Generalized Linear Models”

http://biometry.github.io/APES//LectureNotes/2016-JAGS/Overdispersion/OverdispersionJAGS.html

Joseph M. Hilbe (2014). “Modeling Count Data”. Cambridge University Press

Farid Kianifard, Paul P. Gallo (2007). “Poisson regression analysis in clinical research”. Journal of Biopharmaceutical Statistics

Eugene D. Hahn, Refik Soyer (2005). “Probit and Logit Models:Differences in the Multivariate Realm”

Alan Agresti (2010). “Analysis of Ordinal Categorical Data”. A John Wiley and Sons, Inc, Publication

Anne R. Daykin, Peter G. Moffatt (2010). “Analyzing ordered responses: A review of the ordered probit model. Understanding statistics

Daniel B. Hall (2000). “Zero-inflated Poisson and Binomial Regression with random effects: A case study”. Department of Statistics, University of Georgia

Benjamin Ghojogh, Aydin Ghojogh, Mark Crowley, Fakhri Karray (2020). “Fitting a mixture distribution to Data: Tutorial”. Stat.OT

Sujit K. Ghosh, Pabak Mukhopadhyay, Jye-Chyi Lu (2006). “Bayesian Analysis of zero-inflated regression models”. Journal of statistical planning and inference

Nianci Gan (2000). “General zero-inflated models and their applications”. North Carolina State University Project

Sun Y. Jeon (2013). “Zero-inflated Poisson regression”

Kevin E. Staub, Rainer Winkelmann (2012). “Consistent estimation od zero-inflated count models”. Wiley Online Library

Farid Kianifard, Paul P. Gallo (2007). “Poisson regression analysis in clinical research”. Journal of Biopharmaceutical Statistics

Gary Napier (2020). “Time Series”. Course

Shu Kay Ng, Thriyambakam Krishnan Goeffrey J. McLachlan (2012). “The EM algorithm”. Handbook of Computational Statistics

Nan Laird (1993). “The EM Algorithm”. Handbook of Statistics

Yuzhen Ye (2018). “Expectation-Maximization algorithm (EM)”. Machine Learning in Bioinformatics

Maya R. Gupta, Yihua Chen (2010). “Theory and Use of the EM Algorithm”.

Samis Trevezas (2021). “Statistics for stochastic processes”. Notes, Mathematics Department, Athens

(2016). “The Bradley-Terry model”. Introduction to Statistical Inference, Lecture 24

Roger R. Davidson (1970). “On extending the Bradley-Terry model to accommodate ties in paired comparison experiments”. Journal of the American Statistical Association

Gunther Schauberger, Andreas Groll, Gerhard Tutz (2017). “Analysis of the importance of on-field covariates in the German Bundesliga”. Journal of Applied Statistics

Simon Jackman (2000). “Models for ordered outcomes”. Political Science 200C

Dimitris Karlis, Ioannis Ntzoufras (2003). “Analysis of sports data by using bivariate Poisson models”. The Statistician

Rasmus Ekman (2020). “Bivariate copula-based regression for modeling results of football matches”.

Kocherlakota S., Kocherlakota K. (2001). “Regression in the bivariate Poisson distribution”. Communication in Statistics

Dimitris Karlis, Ioannis Ntzoufras (2020). “Intro and current issues of football analytics”. Short course on football analytics, AUEB

Dimitris Karlis, Ioannis Ntzoufras (2020). “The simple Double Poisson model”. Short course on football analytics, AUEB

Dimitris Karlis, Ioannis Ntzoufras (2005). “Bivariate Poisson and diagonal inflated bivariate Poisson regression models in R”. Journal of Statistical Software

Kimberly F. Sellers, Darcy Steeg Morris, Narayanaswamy Balakrishnan (2016). “Bivariate Conway-Maxwell-Poisson distribution: Formulation, properties and inference”. Journal of Multivariate Analysis

Rufin Bidounga, Evgrand Giles Brunel Mandangui Maloumbi, Reolie Foxie Mizele Kitoti, Dominique Mizere (2020). “The new bivariate Conway-Maxwell-Poisson distribution obtained by the crossing method”. International Journal of Statistics and Probability

pena.lt/y/2015/12/12/frequency-of-draws-in-football/

Mark J. Dixon, Stuart G. Coles (1997). "Modelling association football scores and inefficiencies in the football betting market". Appl. Statist.

Siem Jan Koopman, Rutger Lit (2014). " A dynamic bivariate Poisson model for analyzing and forecasting match results in the English Premier League". Journal of the Royal Statistical Society

Andreas Groll, Thomas Kneib, Andreas Mayr, Gunther Schauberger (2018). "On the dependency of soccer scores - a sparse bivariate Poisson model for the UEFA European football championship 2016". Journal of Sports Analytics

Edward Wheatcroft (2020). "Forecasting football matches by predicting match statistics". London School of Economics and Political Science

Hongyou Liu (2015). "Evaluation on match performances of professional football players and teams under different situational conditions".INEF

Dimitris Karlis, Ioannis Ntzoufras (2005). "Bivariate Poisson models using the EM algorithm". The bivpois Package

Jasmine Siwei Xu (2011). "Online Sports Gambling: A look into the efficiency of bookmakers' odds as forecasts in the case of English Premier League". Undergraduate Economics Honor Thesis, California)

APPENDIX

A1. Data Set

1	team1	team2	g1	g2	rat1	rat2	penbox1	penbox2	goalbox1	goalbox2	corner1	corner2
2	Aek	Xanthi	1	2	82	50	1	3	1	0	3	3
3	Aris	Ofi	1	1	89	70	8	2	1	0	7	1
4	Atromitos	Larisa	1	1	144	155	4	4	1	2	7	2
5	Olympiakc	Tripoli	1	0	167	92	8	3	1	0	4	1
6	Panionios	Volos	1	2	133	131	3	7	0	0	7	2
7	Paok	Panetolikc	2	1	173	92	3	3	1	0	6	2
8	Lamia	Panathinai	1	1	92	142	1	1	1	0	6	5
9	Larisa	Olympiakc	0	1	60	89	0	2	1	1	1	9
10	Tripoli	Aek	2	3	99	226	3	9	0	1	3	7
11	Volos	Aris	1	0	89	110	4	2	0	3	1	8
12	Panathinai	Ofi	1	3	84	114	3	8	0	0	1	1
13	Panetolikc	Xanthi	1	2	118	111	2	4	0	0	5	0
14	Paok	Panionios	2	1	123	55	4	1	1	0	9	2
15	Lamia	Atromitos	2	2	136	178	3	4	0	0	5	6
16	Aek	Lamia	2	0	142	51	6	1	0	0	10	1
17	Xanthi	Tripoli	2	1	119	92	2	4	1	0	3	4
18	Aris	Panathinai	4	0	202	119	5	2	5	1	3	3
19	Atromitos	Paok	2	3	133	160	2	2	1	1	2	5
20	Olympiakc	Volos	5	0	253	34	9	2	2	0	5	3
21	Ofi	Panetolikc	3	1	241	134	8	1	0	0	2	4
22	Panionios	Larisa	1	0	133	121	4	2	0	0	3	6
23	Larisa	Xanthi	3	0	201	87	2	1	0	0	7	0
24	Tripoli	Atromitos	2	1	217	184	8	1	0	0	6	6
25	Volos	Ofi	1	0	135	156	7	4	0	1	3	10
26	Panathinai	Olympiakc	1	1	69	124	3	2	1	0	2	5
27	Lamia	Panionios	1	1	155	117	4	1	0	0	4	1
28	Panetolikc	Aek	0	1	88	165	0	7	0	1	2	3
29	Paok	Aris	2	2	135	102	4	2	0	0	8	3
30	Aek	Paok	2	2	136	137	2	6	1	2	3	4
31	Xanti	Volos	3	1	260	150	8	4	0	0	4	3

32	Aris	Larisa	2	3	204	176	10	3	2	1	15	5
33	Atromitos	Panetolikc	2	0	137	93	4	1	0	0	6	2
34	Olympiakc	Lamia	2	0	249	71	7	2	1	0	6	0
35	Ofi	Tripoli	3	1	264	122	13	3	1	0	7	3
36	Panionios	Panathinai	0	1	78	185	0	7	0	0	4	4
37	Larisa	Aek	0	0	66	72	1	2	0	0	6	4
38	Aris	Olympiakc	1	2	120	166	2	1	0	1	3	3
39	Tripoli	Paok	1	2	180	154	3	3	0	0	5	5
40	Volos	Atromitos	2	3	61	106	2	4	0	0	4	6
41	Ofi	Panionios	4	1	317	148	11	3	0	0	9	2
42	Panathinai	Xanthi	0	1	128	105	1	1	1	2	7	1
43	Lamia	Panetolikc	0	0	118	52	15	1	0	0	3	7
44	Aek	Volos	3	2	198	99	7	1	0	0	10	3
45	Xanthi	Aris	0	1	64	86	3	4	0	0	3	4
46	Atromitos	Panathinai	0	1	102	170	2	3	0	0	3	3
47	Olympiakc	Ofi	2	1	194	109	9	3	1	0	5	0
48	Panetolikc	Larisa	2	2	122	157	2	1	2	1	4	4
49	Panionios	Tripoli	0	1	94	107	3	1	0	0	9	4
50	Paok	Lamia	3	0	225	85	4	1	1	0	5	2
51	Aris	Panetolikc	2	0	106	68	8	0	0	0	2	3
52	Tripoli	Lamia	4	1	193	78	1	2	2	0	5	1
53	Atromitos	Panionios	4	0	200	122	3	1	0	1	3	9
54	Volos	Paok	0	2	132	183	2	3	0	1	2	8
55	Olympiakc	Aek	2	0	173	127	2	6	2	0	7	6
56	Ofi	Xanthi	2	0	157	99	6	2	0	0	2	4
57	Panathinai	Larisa	1	2	151	101	7	2	1	0	6	0
58	Aek	Atromitos	3	2	247	138	8	5	3	1	7	5
59	Larisa	Ofi	3	2	132	140	4	1	0	0	5	5
60	Xanthi	Olympiakc	0	0	124	215	2	5	0	0	2	13
61	Panetolikc	Tripoli	1	1	109	86	3	2	1	0	7	1
62	Panionios	Aris	1	1	98	199	2	6	0	1	3	4
63	Paok	Panathinai	2	2	238	169	7	2	1	1	3	1
64	Lamia	Volos	1	0	181	101	5	3	1	0	11	3
65	Larisa	Lamia	0	3	214	198	3	4	0	0	8	1
66	Xanthi	Panionios	1	2	173	191	2	2	0	2	6	2
67	Aris	Tripoli	2	1	205	183	3	3	1	0	3	5
68	Volos	Panetolikc	3	2	141	168	1	6	3	0	6	6
69	Olympiakc	Atromitos	2	0	231	115	11	0	0	0	11	1
70	Ofi	Paok	0	1	89	128	0	4	0	1	7	5
71	Panathinai	Aek	3	2	172	138	4	11	0	0	9	2
72	Aek	Aris	1	1	117	91	4	2	1	1	6	2
73	Tripoli	Volos	0	0	165	98	3	3	0	0	7	1
74	Atromitos	Ofi	2	1	223	197	9	3	1	0	6	4
75	Panetolikc	Panathinai	0	0	107	118	2	3	0	1	3	5
76	Lamia	Xanthi	1	0	155	84	8	1	0	0	9	2
77	Panionios	Olympiakc	1	1	109	252	1	10	0	0	1	14
78	Paok	Larisa	1	0	218	122	4	1	1	0	7	4
79	Larisa	Volos	2	1	213	185	8	4	1	0	5	2
80	Xanthi	Atromitos	1	0	129	192	2	3	0	0	3	7
81	Aris	Lamia	1	1	263	169	8	1	0	0	9	2
82	Olympiakc	Paok	1	1	205	125	4	2	1	1	9	2
83	Ofi	Aek	1	0	153	108	5	1	0	0	2	5
84	Panathinai	Tripoli	1	0	167	135	4	2	0	0	2	1
85	Panionios	Panetolikc	3	0	159	129	4	1	2	0	3	4
86	Aek	Panionios	5	0	208	86	6	3	2	0	10	4
87	Tripoli	Larisa	1	1	209	124	7	3	1	0	5	1
88	Atromitos	Aris	2	2	163	164	7	10	0	0	4	7
89	Volos	Panathinai	1	1	134	160	6	7	0	0	2	5
90	Panetolikc	Olympiakc	0	3	145	223	2	6	1	1	3	4
91	Paok	Xanthi	2	0	238	102	6	1	2	0	15	0
92	Lamia	Ofi	2	1	143	132	4	6	0	0	3	2
93	Larisa	Atromitos	1	2	207	137	8	4	0	0	8	2

94	Xanthi	Aek	0	1	63	86	0	3	0	0	2	5
95	Tripoli	Olympiakc	0	5	118	286	0	9	0	2	2	4
96	Volos	Panionios	2	1	205	133	4	2	1	0	6	11
97	Ofi	Aris	3	1	174	126	7	6	1	0	4	11
98	Panathinai	Lamia	2	0	166	103	5	3	1	0	10	2
99	Panetolikc	Paok	0	3	81	156	2	2	0	1	2	8
100	Aek	Tripoli	2	1	126	151	4	2	1	1	5	4
101	Xanthi	Panetolikc	0	0	150	87	4	1	0	0	7	2
102	Aris	Volos	4	0	245	157	7	2	1	0	11	6
103	Atromitos	Lamia	1	1	111	120	2	3	1	0	3	2
104	Olympiakc	Larisa	4	1	243	119	8	2	0	0	3	2
105	Ofi	Panathinai	1	1	170	134	3	5	0	0	6	3
106	Panionios	Paok	0	2	120	249	2	7	0	1	3	11
107	Larisa	Panionios	2	0	232	122	3	1	0	0	8	5
108	Tripoli	Xanthi	5	0	257	61	6	2	1	0	4	1
109	Volos	Olympiakc	0	0	86	193	1	6	0	0	4	10
110	Panathinai	Aris	0	0	117	121	4	2	0	0	10	7
111	Panetolikc	Ofi	2	0	148	93	4	1	0	0	3	7
112	Paok	Atromitos	5	1	251	136	3	3	3	0	9	1
113	Lamia	Aek	0	0	124	145	0	4	2	0	5	6
114	Aek	Panetolikc	3	1	169	112	4	4	1	0	8	1
115	Xanthi	Larisa	2	1	114	83	4	2	0	0	7	9
116	Aris	Paok	4	2	154	149	2	4	0	0	3	5
117	Atromitos	Tripoli	2	1	96	114	3	3	1	0	4	1
118	Olympiakc	Panathinai	1	0	195	124	5	3	0	0	5	3
119	Ofi	Volos	1	2	215	130	9	0	1	1	6	2
120	Panionios	Lamia	0	1	94	100	4	1	0	0	8	2
121	Larisa	Aris	0	0	126	121	4	2	0	1	6	4
122	Tripoli	Ofi	2	0	183	122	4	1	1	0	8	3
123	Volos	Xanthi	1	3	80	158	3	1	0	0	2	2
124	Panathinai	Panionios	3	0	166	54	6	3	1	0	6	2
125	Panetolikc	Atromitos	0	1	128	118	1	2	0	1	4	4
126	Paok	Aek	1	0	119	111	7	3	0	0	9	5
127	Lamia	Olympiakc	0	4	80	241	2	8	0	0	2	10
128	Aek	Larisa	3	0	162	57	3	0	1	0	11	3
129	Xanthi	Panathinai	0	1	68	109	2	3	1	0	2	8
130	Atromitos	Volos	0	0	117	69	2	0	1	0	11	4
131	Olympiakc	Aris	4	2	188	148	4	2	3	2	5	3
132	Panetolikc	Lamia	1	1	164	132	4	1	1	0	10	1
133	Panionios	Ofi	1	2	108	241	3	8	0	0	6	6
134	Paok	Tripoli	3	1	196	131	6	2	4	1	6	3
135	Larisa	Panetolikc	2	2	127	133	5	1	0	1	8	5
136	Aris	Xanthi	1	0	135	75	4	1	0	0	4	2
137	Tripoli	Panionios	2	0	157	107	4	4	0	0	7	5
138	Volos	Aek	1	3	62	194	2	8	0	0	3	5
139	Ofi	Olympiakc	0	1	134	144	1	2	0	2	5	4
140	Panathinai	Atromitos	3	0	267	118	10	1	2	0	8	3
141	Lamia	Paok	0	1	66	100	1	1	0	2	3	1
142	Aek	Olympiakc	0	0	81	89	3	3	0	0	3	8
143	Larisa	Panathinai	0	2	84	113	0	4	0	0	1	5
144	Xanthi	Ofi	2	2	145	136	0	4	2	0	2	2
145	Panetolikc	Aris	2	0	164	162	3	4	0	0	7	6
146	Panionios	Atromitos	1	0	215	138	7	2	2	0	9	0
147	Paok	Volos	1	0	161	102	5	2	1	0	9	4
148	Lamia	Tripoli	1	1	60	47	5	3	0	1	0	4
149	Aris	Panionios	2	0	258	78	8	2	1	0	9	2
150	Tripoli	Panetolikc	2	1	222	133	10	3	0	1	5	4
151	Atromitos	Aek	0	1	85	106	0	3	0	1	3	4
152	Volos	Lamia	1	0	159	123	2	2	1	2	3	5
153	Olympiakc	Xanthi	3	1	172	162	4	2	1	0	3	1
154	Ofi	Larisa	0	0	168	119	6	1	1	1	9	3
155	Panathinai	Paok	2	0	164	152	3	3	1	0	3	3

156	Aek	Panathinai	1	0	141	115	1	1	0	0	4	2
157	Tripoli	Aris	1	1	117	127	4	0	0	1	5	10
158	Atromitos	Olympiakc	0	1	126	193	1	5	0	0	2	10
159	Panetolikc	Volos	1	1	100	114	6	0	1	2	8	2
160	Panionios	Xanthi	0	0	120	188	2	2	0	0	2	5
161	Paok	Ofi	4	0	197	111	5	2	0	0	5	0
162	Lamia	Larisa	0	0	118	111	2	6	0	0	3	1
163	Larisa	Paok	1	2	143	159	2	1	0	1	4	1
164	Xanthi	Lamia	0	0	160	115	6	2	0	1	10	1
165	Aris	Aek	0	1	119	137	1	2	0	0	5	2
166	Volos	Tripoli	0	1	72	158	0	6	0	0	1	4
167	Olympiakc	Panionios	4	0	330	115	8	5	3	0	6	5
168	Ofi	Atromitos	1	0	113	69	3	5	1	0	5	1
169	Panathinai	Panetolikc	3	1	198	171	4	4	1	1	8	3
170	Aek	Ofi	3	0	217	122	8	4	0	0	5	1
171	Tripoli	Panathinai	1	1	125	110	2	3	1	0	7	0
172	Atromitos	Xanthi	1	0	167	169	3	0	1	0	7	4
173	Volos	Larisa	0	0	112	90	2	1	1	0	9	2
174	Panetolikc	Panionios	1	0	129	83	2	0	1	0	7	0
175	Paok	Olympiakc	0	1	111	124	0	3	0	0	12	3
176	Lamia	Aris	2	2	152	161	7	2	1	1	6	0
177	Larisa	Tripoli	3	0	150	132	3	1	1	0	2	4
178	Xanthi	Paok	1	1	119	135	0	3	0	1	5	6
179	Aris	Atromitos	1	2	176	142	6	1	0	1	5	3
180	Olympiakc	Panetolikc	2	0	211	84	6	0	0	0	7	0
181	Ofi	Lamia	3	0	164	101	6	3	1	0	4	4
182	Panathinai	Volos	4	1	179	137	5	2	1	1	6	4
183	Panionios	Aek	1	1	146	215	4	9	1	1	7	11
184	Aek	Panathinai	1	1	116	115	1	2	0	0	1	1
185	Aris	Ofi	3	1	169	177	3	4	0	0	3	6
186	Paok	Olympiakc	0	1	126	112	2	3	0	0	4	4
187	Xanthi	Atromitos	1	0	120	104	1	1	1	0	4	4
188	Larisa	Tripoli	1	2	123	133	4	4	1	0	4	7
189	Panionios	Volos	1	0	118	125	3	2	1	0	4	7
190	Lamia	Panetolikc	2	0	124	122	4	3	0	1	3	7
191	Olympiakc	Aris	3	1	217	161	7	3	1	1	5	1
192	Ofi	Aek	0	2	91	160	2	4	0	1	2	3
193	Panathinai	Paok	0	0	123	145	1	8	0	0	1	5
194	Tripoli	Panionios	0	0	154	128	2	5	0	0	3	3
195	Atromitos	Lamia	1	1	143	99	0	0	1	1	5	3
196	Volos	Xanthi	1	0	139	106	4	0	0	0	2	5
197	Panetolikc	Larisa	3	0	169	62	5	1	0	1	4	3
198	Aek	Aris	2	2	168	128	5	1	0	0	4	4
199	Olympiakc	Panathinai	3	0	196	135	2	0	0	0	2	4
200	Paok	Ofi	3	1	265	136	8	0	2	1	7	1
201	Xanthi	Panetolikc	1	1	142	123	10	2	0	0	9	3
202	Larisa	Volos	3	1	149	95	6	5	0	1	4	3
203	Tripoli	Atromitos	1	1	159	127	3	3	0	1	4	2
204	Lamia	Panionios	0	1	54	101	0	2	0	0	5	2
205	Aek	Olympiakc	1	2	128	225	1	4	2	2	7	6
206	Aris	Paok	0	2	138	178	2	7	0	1	9	4
207	Ofi	Panathinai	0	0	170	132	4	1	0	0	2	1
208	Atromitos	Larisa	3	0	212	104	9	1	0	0	6	1
209	Volos	Lamia	0	0	117	102	5	1	0	0	4	0
210	Panetolikc	Tripoli	1	1	153	124	5	5	0	0	5	3
211	Panionios	Xanthi	2	1	124	187	2	4	3	1	4	6
212	Panathinai	Aris	2	0	149	87	2	0	1	0	3	4
213	Olympiakc	Ofi	2	1	204	151	5	2	2	0	5	3
214	Paok	Aek	0	2	131	185	3	3	0	1	4	8

215	Aris	Aek	1	4	138	244	1	9	0	2	2	5
216	Ofi	Paok	2	2	113	127	3	5	0	2	3	3
217	Panathinai	Olympiakc	0	0	121	135	3	2	0	0	2	5
218	Larisa	Panionios	0	0	165	73	1	1	0	0	9	4
219	Tripoli	Volos	4	0	198	140	3	4	0	0	3	1
220	Atromitos	Panetolikc	2	2	174	158	5	2	0	1	4	4
221	Lamia	Xanthi	0	0	46	69	0	1	0	0	0	8
222	Aek	Ofi	2	0	134	91	8	0	0	0	2	2
223	Aris	Olympiakc	2	4	210	185	5	8	0	0	6	6
224	Paok	Panathinai	0	0	183	159	2	2	0	0	4	2
225	Olympiakc	Paok	0	1	190	158	2	6	0	1	4	8
226	Ofi	Aris	0	1	110	175	6	8	0	1	4	3
227	Panathinai	Aek	1	3	162	158	6	5	0	0	6	3
228	Xanthi	Tripoli	1	2	143	118	4	7	0	1	11	1
229	Volos	Atromitos	2	3	188	210	5	5	0	1	4	4
230	Panionios	Panetolikc	0	2	136	221	2	2	1	2	5	5
231	Lamia	Larisa	0	0	145	115	0	2	1	0	6	2
232	Aek	Paok	0	0	156	138	4	1	0	0	6	4
233	Aris	Panathinai	0	1	135	206	8	9	0	1	3	3
234	Ofi	Olympiakc	1	3	113	169	3	8	1	1	7	4
235	Olympiakc	Aek	3	0	228	136	5	3	2	0	6	2
236	Panathinai	Ofi	3	2	183	181	5	9	0	0	2	5
237	Paok	Aris	0	0	140	126	7	1	0	1	5	3
238	Larisa	Xanthi	0	0	115	110	2	1	0	0	2	4
239	Tripoli	Lamia	1	1	171	119	4	1	1	0	3	3
240	Atromitos	Panionios	0	0	198	133	2	1	1	0	13	3
241	Panetolikc	Volos	1	0	144	119	5	1	1	1	1	4
242	Aek	Olympiakc	1	1	122	143	2	3	3	0	9	4
243	Apollon	Giannena	1	2	133	176	5	5	0	1	3	3
244	Aris	Lamia	3	1	177	110	3	2	0	0	7	0
245	Tripoli	Panathinai	1	0	141	114	3	0	0	1	1	3
246	Atromitos	Volos	0	2	121	140	2	3	0	1	2	2
247	Ofi	Panetolikc	1	1	112	138	4	1	0	0	1	2
248	Paok	Larisa	1	0	192	105	7	6	1	0	8	3
249	Giannena	Larisa	1	2	103	131	2	3	2	0	0	5
250	Volos	Aris	0	1	114	148	1	4	0	0	2	6
251	Olympiakc	Tripoli	3	0	255	153	8	2	1	0	4	5
252	Panathinai	Apollon	1	0	104	83	2	2	0	0	5	4
253	Panetolikc	Aek	0	2	76	179	1	3	0	0	1	7
254	Paok	Atromitos	1	1	203	123	6	4	0	0	7	3
255	Lamia	Ofi	1	2	147	140	5	0	1	0	4	4
256	Aek	Lamia	3	0	314	172	5	3	4	0	7	5
257	Larisa	Panathinai	1	1	101	127	1	2	1	0	3	4
258	Aris	Giannena	2	2	162	141	4	0	0	1	3	3
259	Tripoli	Apollon	0	0	169	128	1	1	0	1	3	9
260	Volos	Paok	0	0	128	185	1	4	0	3	2	5
261	Olympiakc	Panetolikc	2	0	231	99	7	1	1	1	8	5
262	Ofi	Atromitos	2	2	204	139	8	3	0	1	4	3
263	Giannena	Olympiakc	1	1	167	180	2	3	0	3	4	7
264	Apollon	Larisa	1	0	116	107	3	5	0	0	3	6
265	Atromitos	Aek	1	0	199	100	3	5	0	0	2	0
266	Panathinai	Aris	0	1	105	130	5	3	1	0	8	2
267	Panetolikc	Tripoli	1	1	167	248	2	6	0	0	3	15
268	Paok	Ofi	3	0	243	154	4	3	1	0	8	4
269	Lamia	Volos	1	2	151	183	5	3	1	2	0	4
270	Aek	Paok	1	1	107	183	1	3	0	3	2	2
271	Larisa	Tripoli	1	3	151	212	5	5	1	2	5	5
272	Aris	Apollon	1	0	158	86	6	2	0	0	0	4
273	Volos	Giannena	2	1	166	117	2	2	0	0	4	4
274	Olympiakc	Atromitos	4	0	233	94	3	1	3	0	6	1
275	Ofi	Panathinai	1	1	199	172	3	4	1	1	9	5
276	Lamia	Panetolikc	0	0	99	119	3	0	0	0	4	3

277	Giannena	Aek	0	1	120	119	1	4	0	0	3	3
278	Larisa	Aris	0	3	129	194	2	4	1	1	7	6
279	Apollon	Lamia	0	1	111	129	1	4	0	0	8	0
280	Tripoli	Ofi	1	0	146	119	3	2	0	0	5	5
281	Atromitos	Panetolikc	2	0	195	109	6	1	0	0	4	7
282	Panathinai	Volos	1	1	159	145	4	3	0	0	9	5
283	Paok	Olympiakc	1	1	225	153	3	2	2	1	4	5
284	Aek	Ofi	2	1	152	140	7	4	0	0	5	0
285	Aris	Tripoli	1	0	213	148	9	0	0	1	6	3
286	Atromitos	Giannena	0	2	112	215	0	7	1	0	0	5
287	Volos	Larisa	1	1	153	141	5	5	0	1	3	2
288	Olympiakc	Apollon	2	0	236	163	4	5	3	0	11	7
289	Panetolikc	Paok	1	3	133	190	1	5	0	3	0	9
290	Lamia	Panathinai	0	2	113	181	2	9	9	2	1	9
291	Giannena	Panetolikc	0	0	114	96	5	1	1	0	7	2
292	Larisa	Lamia	0	1	146	221	3	3	1	2	5	4
293	Apollon	Paok	1	3	131	163	3	6	1	0	7	4
294	Aris	Aek	0	1	118	143	2	0	0	0	4	1
295	Tripoli	Volos	1	1	123	139	4	1	0	1	5	1
296	Ofi	Olympiakc	0	2	88	201	3	6	0	1	3	7
297	Panathinai	Atromitos	0	1	100	156	3	0	0	0	3	5
298	Aek	Larisa	4	1	167	226	8	3	1	0	11	2
299	Atromitos	Apollon	2	2	152	140	6	1	1	2	13	2
300	Volos	Ofi	1	4	133	151	6	5	1	1	7	5
301	Olympiakc	Panathinai	1	0	234	84	5	1	0	2	3	5
302	Panetolikc	Aris	0	1	158	129	3	4	0	0	7	4
303	Paok	Giannena	2	1	298	144	8	2	2	0	13	1
304	Lamia	Tripoli	2	2	117	129	5	5	0	0	3	5
305	Larisa	Atromitos	0	0	91	91	1	2	0	0	5	3
306	Apollon	Volos	3	3	228	182	11	3	3	1	2	8
307	Aris	Olympiakc	1	2	134	246	4	3	1	1	7	1
308	Tripoli	Aek	1	2	162	126	1	8	0	0	2	2
309	Ofi	Giannena	2	1	111	115	4	2	0	0	3	7
310	Panathinai	Panetolikc	2	1	130	100	3	1	0	0	4	3
311	Lamia	Paok	0	2	115	163	3	3	0	1	1	3
312	Aek	Panathinai	1	2	150	153	2	2	1	2	9	2
313	Giannena	Lamia	2	0	181	86	4	3	1	0	10	4
314	Atromitos	Aris	2	2	123	129	3	3	0	0	1	3
315	olympiako	Volos	4	1	266	137	8	2	2	0	6	2
316	Ofi	Apollon	0	2	198	160	5	2	1	1	7	4
317	Panetolikc	Larisa	2	1	144	141	2	6	0	1	2	7
318	Paok	Tripoli	2	0	172	107	2	2	2	0	5	4
319	Larisa	Ofi	0	1	122	178	0	3	0	0	2	5
320	Apollon	Aek	3	4	200	205	2	3	0	2	1	7
321	Aris	Paok	1	0	151	147	2	4	0	0	0	5
322	Tripoli	Atromitos	2	0	237	118	6	0	1	0	4	3
323	Volos	Panetolikc	0	0	143	97	5	0	0	0	10	4
324	Lamia	Olympiakc	0	6	104	252	4	4	0	3	3	9
325	Panathinai	Giannena	2	0	137	112	2	4	1	1	5	1
326	Aek	Volos	2	2	204	150	5	3	3	3	9	5
327	Giannena	Tripoli	2	2	140	202	3	5	1	3	2	2
328	Atromitos	Lamia	2	1	164	150	4	3	0	0	7	1
329	Olympiakc	Larisa	5	1	299	120	6	2	5	1	8	3
330	Ofi	Aris	0	3	123	139	6	4	0	1	5	3
331	Panetolikc	Apollon	0	1	100	121	2	2	0	0	3	2

332	Paok	Panathinai	2	1	217	144	4	2	3	2	6	1
333	Giannena	Apollon	1	3	128	144	3	2	0	1	4	2
334	Larisa	Paok	1	1	123	210	1	10	0	2	2	12
335	Volos	Atromitos	1	0	164	141	3	4	2	0	7	3
336	Olympiakc	Aek	3	0	145	109	5	3	0	0	3	1
337	Panathinai	Tripoli	0	0	111	111	1	4	1	0	1	3
338	Panetolikc	Ofi	2	1	122	151	2	4	1	1	6	3
339	Lamia	Aris	2	0	142	113	3	0	0	1	4	5
340	Aek	Panetolikc	1	0	160	83	2	1	1	0	5	2
341	Larisa	Giannena	0	0	155	117	4	3	0	0	3	4
342	Apollon	Panathinai	0	1	164	121	3	4	1	1	3	3
343	Aris	Volos	2	0	223	119	10	0	0	0	3	3
344	Tripoli	Olympiakc	0	4	98	260	2	6	0	1	1	2
345	Atromitos	Paok	3	2	188	170	2	3	0	1	2	4
346	Ofi	Lamia	2	0	175	94	5	1	0	0	4	1
347	Giannena	Aris	0	0	111	131	5	6	0	0	3	7
348	Apollon	Tripoli	0	1	152	110	3	2	1	0	7	3
349	Atromitos	Ofi	0	0	184	162	8	7	0	1	0	7
350	Panathinai	Larisa	2	0	179	164	4	1	1	1	1	4
351	Panetolikc	Olympiakc	1	2	121	190	2	6	0	1	2	5
352	Paok	Volos	3	1	228	155	10	5	2	1	5	3
353	Lamia	Aek	0	1	104	178	0	4	0	1	1	2
354	Aek	Atromitos	2	1	166	78	3	1	1	0	4	2
355	Larisa	Apollon	0	1	132	139	4	3	0	0	1	6
356	Aris	Panathinai	0	1	306	194	14	4	0	1	9	0
357	Tripoli	Panetolikc	2	0	234	111	4	1	1	0	2	1
358	Volos	Lamia	1	1	193	191	6	6	0	1	6	3
359	Olympiakc	Giannena	1	0	209	115	8	2	1	0	8	2
360	Ofi	Paok	0	3	106	257	0	9	0	2	2	6
361	Giannena	Volos	0	1	138	97	3	2	2	0	5	1
362	Apollon	Aris	0	1	111	190	1	1	0	3	1	6
363	Tripoli	Larisa	1	0	169	87	3	1	1	0	4	3
364	Atromitos	Olympiakc	0	1	133	198	0	5	0	0	3	5
365	Panathinai	Ofi	2	0	172	173	3	3	0	0	6	3
366	Panetolikc	Lamia	0	0	131	142	1	4	0	0	2	7
367	Paok	Aek	2	2	175	193	4	2	2	3	7	4
368	Aek	Giannena	0	2	137	120	2	5	0	0	5	2
369	Aris	Larisa	1	0	201	121	5	2	0	0	3	4
370	Volos	Panathinai	0	2	162	118	5	2	2	0	10	2
371	Olympiakc	Paok	3	0	213	143	8	2	0	0	1	4
372	Ofi	Tripoli	0	1	195	104	3	3	0	0	4	2
373	Panetolikc	Atromitos	1	1	147	132	4	2	0	0	6	2
374	Lamia	Apollon	1	0	163	96	5	0	1	0	4	2
375	Giannena	Atromitos	0	1	126	134	2	4	3	0	9	1
376	Larisa	Volos	0	0	119	150	0	5	0	0	2	10
377	Apollon	Olympiakc	1	3	96	259	1	4	1	3	1	9
378	Tripoli	Aris	2	1	174	170	1	5	2	1	1	7
379	Ofi	Aek	0	2	110	153	1	2	0	1	3	6
380	Panathinai	Lamia	0	0	155	141	6	2	0	0	8	2
381	Paok	Panetolikc	5	0	230	94	5	2	4	0	4	5
382	Aek	Aris	0	2	135	175	3	3	0	3	4	3
383	Atromitos	Panathinai	2	3	197	153	3	1	0	2	6	6
384	Volos	Tripoli	0	1	130	147	2	2	0	0	3	0
385	Olympiakc	Ofi	3	0	287	114	10	1	3	0	7	1
386	Panetolikc	Giannena	1	2	142	134	3	4	1	0	4	5
387	Paok	Apollon	2	2	233	160	6	4	1	0	15	2
388	Lamia	Larisa	2	1	127	108	2	2	2	1	3	2
389	Giannena	Paok	0	2	123	209	1	7	0	0	2	7
390	Larisa	Aek	2	4	144	239	5	6	0	1	2	3
391	Apollon	Atromitos	2	1	156	187	2	3	0	2	1	3
392	Aris	Panetolikc	0	0	162	120	2	1	1	0	7	2

393	Tripoli	Lamia	0	0	202	149	3	1	0	0	11	4
394	Ofi	Volos	1	2	113	150	4	2	1	0	3	1
395	Panathinai	Olympiakc	2	1	177	262	1	5	1	3	4	2
396	Aek	Tripoli	2	2	253	188	10	4	0	0	7	5
397	Giannena	Ofi	1	0	173	118	2	0	2	0	3	5
398	Atromitos	Larisa	1	1	224	158	3	6	0	0	5	5
399	Volos	Apollon	2	0	190	147	8	5	0	1	2	6
400	Olympiakc	Aris	1	1	182	186	2	3	1	0	10	2
401	Panetolikc	Panathinai	1	0	116	162	2	5	0	1	2	8
402	Paok	Lamia	4	0	312	95	10	0	4	0	4	2
403	Larisa	Panetolikc	1	0	162	148	1	0	0	1	6	2
404	Apollon	Ofi	2	1	201	201	3	5	2	0	5	4
405	Aris	Atromitos	3	0	216	142	5	0	1	0	5	2
406	Tripoli	Paok	2	1	174	197	5	2	0	0	2	2
407	Volos	Olympiakc	1	2	139	163	1	1	0	1	1	4
408	Panathinai	Aek	1	1	149	161	2	3	0	0	4	3
409	Lamia	Giannena	0	0	127	105	2	0	1	0	4	1
410	Aek	Apollon	2	0	206	96	6	2	2	0	4	1
411	Giannena	Panathinai	1	0	156	127	3	0	0	0	5	2
412	Atromitos	Tripoli	1	1	148	152	0	4	0	1	1	2
413	Olympiakc	Lamia	3	0	237	107	4	0	0	0	6	3
414	Ofi	Larisa	2	3	137	140	2	6	0	0	5	1
415	Panetolikc	Volos	1	0	146	149	1	4	1	0	1	3
416	Paok	Aris	2	2	202	186	6	5	2	1	7	3
417	Larisa	Olympiakc	1	3	186	228	4	4	1	1	2	5
418	Apollon	Panetolikc	1	0	157	145	2	5	0	0	3	4
419	Aris	Ofi	1	0	182	120	5	2	2	0	7	1
420	Tripoli	Giannena	0	1	106	148	3	5	0	0	5	4
421	Volos	Aek	1	0	120	106	3	4	0	0	4	3
422	Panathinai	Paok	2	1	163	215	2	6	1	2	1	6
423	Lamia	Atromitos	0	0	138	146	2	1	0	0	5	4

A2. R-Code

Function bivpois.table

```
"bivpois.table" <-
function(x, y, lambda = c(1, 1, 1))
{

      j<-0
      n <- length(x)
      maxy <- c(max(x), max(y))   #Set   initial   values   for
parameters
      lambda1 <- lambda[1]
      lambda2 <- lambda[2]
      lambda3 <- lambda[3]
      if((x == 0) | (y == 0)) {
            prob  <-  matrix(NA,  nrow  =  maxy[1]  +  1,  ncol  =
maxy[2]+1, byrow = T)
            prob[maxy[1] + 1, maxy[2] + 1] <- exp( - lambda3) *
                  dpois(x[j],    lambda1[j])    *    dpois(y[j],
lambda2[j])
```

```
}
      else {
            prob  <-  matrix(NA,  nrow  =  maxy[1]  +  1,  ncol  =
maxy[2]+1, byrow = T)
            k <- 1
            m <- 1
            prob[k, m] <- exp( - lambda1 - lambda2 - lambda3)
            for(i in 2:(maxy[1] + 1)) {
                  prob[i, 1] <- (prob[i - 1, 1] * lambda1)/(i -
1)
            }
            for(j in 2:(maxy[2] + 1)) {
                  prob[1, j] <- (prob[1, j - 1] * lambda2)/(j -
1)
            }
            for(j in 2:(maxy[2] + 1)) {
                  for(i in 2:(maxy[1] + 1)) {
                        prob[i, j] <- (lambda1 * prob[i - 1, j] +
                          lambda3 * prob[i - 1, j - 1])/(i - 1)
                  }
            }
      }
      result <- prob
      result
}
```

Function lm.bp

```
"lm.bp" <-
function(    l1,    l2,    l1l2=NULL,    l3=~1,    data,
common.intercept=FALSE,  zeroL3=FALSE,  maxit=300,  pres=1e-8,
verbose=getOption('verbose') )
#
{
options(warn=-1)
#
# definition of function call
templist<-list(    l1=l1,    l2=l2,    l1l2=l1l2,    l3=l3,
data=substitute(data),      common.intercept=common.intercept,
zeroL3=zeroL3, maxit=maxit, pres=pres, verbose=verbose)
tempcall<-as.call( c(expression(lm.bp), templist))
rm(templist)
```

```
# l1                              :  formula  for  the  first
linear predictor (of lambda1)
# l2                              :  formula  for  the  second
linear predictor (of lambda2)
# l1l2                            : formula for common variables
on both lambda1 and lambda2
# l3                              : formula for the third first
linear predictor/covariance parameter (lambda3)
# common.intercept: logical argument defining whether common
intercept should be used for lamdba1,lambda2
#
# data          :  data.frame  which  contains  data  {required
arguement}
# zeroL3    : Logical argument controlling whether lambda3 is
zero (DblPoisson) or not
# maxit    : maximum number of iterations
# pres      : precision of the relative likelihood difference
after which EM stops
# verbose   :   Logical   argument   controlling   whether   beta
parameters will we
#                 printed while EM runs. Default value is taken
options()$verbose value.
# ---------------------------------------------------------------
-----------------------
#
#
#
# set common or noncommon intercept
if (common.intercept){ formula1.terms<-'1'  }
else {formula1.terms<-'internal.data1$noncommon' }
#
#
namex<-as.character(l1[2])
namey<-as.character(l2[2])
x<-data[,names(data)==namex]
y<-data[,names(data)==namey]
#
# Data length
n<-length(x)
lengthpvec<-1
#
#
#
# initial values
```

```
s<-rep(0,n)
like<-1:n*0
zero<- ( x==0 )|( y==0 )
if   (zeroL3) { lambda3<-rep(0,n) }
else                              {   lambda3<-rep(   max(0.1,
cov(x,y,use='complete.obs')), n) }
#
#
# form dataframes used
# data1 includes modelling on lambda1 and lambda2
# data2 includes modelling on lambda3
# internal.data1 and internal.data2 are data frames used for
additional internal variables
#
internal.data1<-data.frame( y1y2=c( x, y) )
internal.data2<-data.frame( y3 = rep(0, n ) )
#
p<-length(as.data.frame(data))
data1<-rbind(data, data)
names(data1)<-names(data)
#
# removing x and y
data1<-data1[ , names(data1)!=namex]
data1<-data1[ , names(data1)!=namey]
#
#
# define full model
if    (as.character(l1[3])=='.')    {    l1<-formula(    paste(
as.character(l1[2]), paste( names(data1),'',collapse='+',sep=''
), sep='~')      )    }
if    (as.character(l2[3])=='.')    {    l2<-formula(    paste(
as.character(l2[2]), paste( names(data1),'',collapse='+',sep=''
), sep='~')  )   }
if (as.character(l3[2])=='.') { l3<-formula( paste( '', paste(
names(data1),'',collapse='+',sep='' ) , sep='~') )   }
#
# define the formula used for covariance term
formula2<-
formula(paste('internal.data2$y3~',as.character(l3[2]),sep=''))
#
internal.data1$noncommon<- as.factor(c(1:n*0,1:n*0+1))
contrasts(internal.data1$noncommon)<-contr.treatment(2, base=1)
internal.data1$indct1<-c(1:n*0+1,1:n*0  )
internal.data1$indct2<-c(1:n*0  ,1:n*0+1)
```

```
#
#
if (!zeroL3){
          data2<-data1[1:n,]
          names(data2)<-names(data1)
          }
#
#####
#
# add the common terms
#
if ( !is.null(l1l2) ) {
     formula1.terms<-paste(                    formula1.terms,
as.character(l1l2[2]),sep='+')
     }
#
# add the special common terms (if any)
#
#
#
# in this section we identify non-common parameters
# if a variable X is common in all formulas the we use term
x*noncommon to include x+x:noncommon terms
# otherwise use I(internal.data1$indct1*x) to add sepererate
parameter on lambda1
#
templ1<- labels(terms(l1))
#
# run this only if there are terms in l1 formula
if (length( templ1 )>0){
     for ( k1 in 1:length( templ1 ) ){
          if    (    !is.null(l1l2)    )    {    checkvar1<-
sum(labels(terms(l1l2))==templ1[k1] )==1    }
          else{ checkvar1<-FALSE }
          checkvar2<-sum(labels(terms(l2))==templ1[k1] )==1
          if     (checkvar1&checkvar2)     {formula1.terms<-
paste(formula1.terms,
paste('internal.data1$noncommon*',templ1[k1],sep=''),  sep='+')
     }
          else{
               formula1.terms<-paste(formula1.terms,
paste('+I(internal.data1$indct1*',templ1[k1],sep=''),   sep='')
```

```
                formula1.terms<-paste(formula1.terms,
')',sep='')
                }
     }
}
#
# if a variable X is not common st
# otherwise use I(internal.data1$indct1*x) to add sepererate
parameter on lambda1
#
templ2<- labels(terms(l2))
#
# run this only if there are terms in l1 formula
if (length( templ2 )>0){
     for ( k1 in 1:length( templ2 ) ){
          if     (     !is.null(l1l2)     )     {checkvar1<-
(sum(labels(terms(l1l2))==templ2[k1]
)+sum(labels(terms(l1))==templ2[k1] ))!=2        }
          else{ checkvar1<-TRUE }
          if ( checkvar1 ) {
                formula1.terms<-paste(formula1.terms,
paste('+I(internal.data1$indct2*',templ2[k1],sep=''), sep='')
                formula1.terms<-paste(formula1.terms,
')',sep='')
                }
     }
}
#
rm(templ1)
rm(templ2)
rm(Checkvar1)
rm(Checkvar2)
#
#
#
#
#
#
# This bit creates labels for special terms of type c(x1,x2)
used in l1l2
#
#
formula1<-
formula(paste('internal.data1$y1y2~',formula1.terms,sep=''))
```

```
tmpform1<-as.character(formula1[3])
newformula<-formula1
while( regexpr('c\\(',tmpform1) != -1)
{
      temppos1<-regexpr('c\\(',tmpform1)[1]
      tempfor <-substring( tmpform1, first = temppos1+ 2 )
      temppos2<-regexpr('\\)' , tempfor)[1]
      tempvar  <-substring(  tempfor   ,  first  =  1,    last  =
temppos2-1 )
      temppos3<-regexpr(', ' , tempvar)[1]
      tempname1<-substring(tempfor      ,  first  =  1,    last  =
temppos3-1 )
      tempname2<-substring(tempfor       ,  first  =  temppos3+2,
last=temppos2-1)
      tempname2<-sub( '\\)','', tempname2  )
      tempvar1<-data[, names(data)==tempname1]
      tempvar2<-data[, names(data)==tempname2]
      data1$newvar1<-c(tempvar1, tempvar2)
#
      if( is.factor(tempvar1)& is.factor(tempvar2) ){
            data1$newvar1<-as.factor(data1$newvar1)
            if (all(levels(tempvar1)==levels(tempvar2))){
                  attributes(data1$newvar1)<-
attributes(tempvar1)}
      }
      tempvar<-sub( ', ' , '..',  tempvar )
      names(data1)[names(data1)=='newvar1']<-tempvar
      newformula<-sub( 'c\\(','', tmpform1  )
      newformula<-sub( '\\)','', newformula  )
      newformula<-sub( ', ' , '..', newformula  )
      tmpform1<-newformula
      formula1<-
formula(paste('internal.data1$y1y2~',newformula,sep=''))
}
#####
rm(temppos1)
rm(temppos2)
rm(temppos3)
rm(tmpform1)
rm(tempfor)
rm(tempvar)
rm(tempvar1)
rm(tempvar2)
rm(tempname1)
```

```
rm(tempname2)
#
#
# Initial values for lambda
#
lambda<-glm(formula1,family=poisson, data=data1)$fitted
#
lambda1<-lambda[1:n]
lambda2<-lambda[(n+1):(2*n)]
#
difllike<-100.0
loglike0<-1000.0
i<-0
#
# fitting the Double Poisson Model
if (zeroL3) {
      #
      # fit the double Poisson model
      y0<-c(x,y)
      m<-glm( formula1, family=poisson, data=data1 )
      p3<-length(m$coef)
      beta<-m$coef
# -----------------------------------------------------
#     creating names for parameters
#
      names(beta)<-newnamesbeta( beta )
#
#     end of name creations (l1, l2, l2-l1, blank)
#  --------------------------------------------------
      betaparameters<-splitbeta( beta )
#
      lambda<-fitted(m)
      lambda1<-lambda[1:n]
      lambda2<-lambda[(n+1):(2*n)]
      like<-dpois(x, lambda1) * dpois( y, lambda2 )
      loglike<-sum(log(like))
#
#     calculation of BIC and AIC for bivpoisson model
      noparams<- m$rank
      AIC<- -2*loglike + noparams * 2
      BIC<- -2*loglike + noparams * log(2*n)
#
#
#     Calculation of BIC, AIC of Poisson saturated model
```

```
	x.mean<-x
	x.mean[x==0]<-1e-12
	y.mean<-y
	y.mean[y==0]<-1e-12
	AIC.sat <-  sum(log( dpois( x , x.mean ) ) + log( dpois( y , y.mean ) ))
	BIC.sat <-  -2 * AIC.sat + (2*n)* log(2*n)
	AIC.sat <-  -2 * AIC.sat + (2*n)* 2
#
#
	AICtotal<-c(AIC.sat, AIC);
	BICtotal<-c(BIC.sat, BIC );
	names(AICtotal)<-c('Saturated', 'DblPois')
	names(BICtotal)<-c('Saturated', 'DblPois')
#
#	putting all betas in one vector
	allbeta<-c(betaparameters$beta1,betaparameters$beta2)
	names(allbeta)<-c( paste( '(l1):', names(betaparameters$beta1), sep='' ),paste('(l2):', names(betaparameters$beta2), sep='' ) )

	result<-list(coefficients=allbeta,
fitted.values=data.frame(x=m$fitted[1:n],y=m$fitted[(n+1):(2*n)]),
	residuals=data.frame(x=x-m$fitted[1:n],y=y-m$fitted[(n+1):(2*n)]),
	beta1=betaparameters$beta1, beta2=betaparameters$beta2, lambda1=m$fitted[1:n], lambda2=m$fitted[(n+1):(2*n)], lambda3=0, loglikelihood=loglike, iterations=1, parameters=noparams, AIC=AICtotal, BIC=BICtotal, call=tempcall)
}
else {
	loglike<-rep(0,maxit)
	while ( (difllike>pres) && (i <= maxit) ) {
	i<-i+1
	#####   E step  ######
	for (j in 1:n) {
		if (zero[j]) {
			s[j]<-0.0;
			like[j]<- log(dpois(x[j], lambda1[j]))+log(dpois(y[j],lambda2[j])) - lambda3[j];
			}
		else {
```

```
                lbp1<-pbivpois(x[j]-1,                    y[j]-
1,lambda=c(lambda1[j],lambda2[j],lambda3[j]), log=TRUE);
                lbp2<-pbivpois(x[j]                ,        y[j]
,lambda=c(lambda1[j],lambda2[j],lambda3[j]), log=TRUE);
#
                s[j]<-exp(log(lambda3[j])+lbp1-lbp2);
                like[j]<-lbp2;
           }
      }
      ##### end of E step  ######
      x1<-x-s
      x2<-y-s

      x1[ (x1<0)&(x1>-1.0e-8)]<-0.00
      x2[ (x2<0)&(x2>-1.0e-8)]<-0.00

      loglike[i]<-sum(like)
      difllike<-abs( (loglike0-loglike[i])/loglike0 )
      loglike0<-loglike[i]
      #
      #
      #####   M step  ######
      #
      #    fit model on lambda3
      internal.data2$y3<-s
      m0<-glm( formula2, family=poisson, data=data2 )
      beta3<-m0$coef
      lambda3<-m0$fitted
      #
      #    fit model on lambda1 & lambda2
      internal.data1$y1y2<-c(x1,x2)

      m<-glm( formula1, family=poisson, data=data1 )
      p3<-length(m$coef)
      beta<-m$coef
#     creating names for parameters
      names(beta)<-newnamesbeta( beta )
#
#

      lambda<-fitted(m)
      lambda1<-lambda[1:n]
      lambda2<-lambda[(n+1):(2*n)]
      #####   end of M step  ######
```

```
#
#     detailed or compressed printing during the EM iterations
      if (verbose) {
            printvector<-c( i, beta, beta3,loglike[i], difllike
)
            names(printvector)<-c(      'iter',      names(beta),
paste('(l3):',names(beta3),sep=''),                   'loglike',
'Rel.Dif.loglike')}
      else {
            printvector<-c( i, loglike[i], difllike )
            names(printvector)<-c(       'iter',       'loglike',
'Rel.Dif.loglike')}
#
      lengthpvec<-length(printvector)
      print.default( printvector, digits=4 )
      }
#
#     calculation of BIC and AIC for bivpoisson model
      noparams<- m$rank + m0$rank
      AIC<- -2*loglike[i] + noparams * 2
      BIC<- -2*loglike[i] + noparams * log(2*n)
#
#
#     Calculation of BIC, AIC of Poisson saturated model
      x.mean<-x
      x.mean[x==0]<-1e-12
      y.mean<-y
      y.mean[y==0]<-1e-12
      AIC.sat <-  sum(log( dpois( x , x.mean ) ) + log( dpois(
y , y.mean ) ))
      BIC.sat <-  -2 * AIC.sat + (2*n)* log(2*n)
      AIC.sat <-  -2 * AIC.sat + (2*n)* 2
#
#
      AICtotal<-c(AIC.sat, AIC);
      BICtotal<-c(BIC.sat, BIC );
      names(AICtotal)<-c('Saturated', 'BivPois')
      names(BICtotal)<-c('Saturated', 'BivPois')
#
#     spliting parameter vector
      betaparameters<-splitbeta( beta )
#
#     putting all betas in one vector
```

```
      allbeta<-c(betaparameters$beta1,betaparameters$beta2,
beta3)
      names(allbeta)<-c(             paste(             '(l1):',
names(betaparameters$beta1),       sep=''       ),paste('(l2):',
names(betaparameters$beta2),       sep=''       ),paste('(l3):',
names(beta3), sep='' ) )
#
#  Calculation of output
      result<-list(coefficients=allbeta,
fitted.values=data.frame(x=m$fitted[1:n]+lambda3,y=m$fitted[(n+
1):(2*n)]+lambda3),
      residuals=data.frame(x=x-m$fitted[1:n]-lambda3,y=y-
m$fitted[(n+1):(2*n)]-lambda3),
      beta1=betaparameters$beta1,    beta2=betaparameters$beta2,
beta3=beta3,                                lambda1=m$fitted[1:n],
lambda2=m$fitted[(n+1):(2*n)],                   lambda3=lambda3,
loglikelihood=loglike[1:i],  parameters=noparams,  AIC=AICtotal,
BIC=BICtotal,iterations=i, call=tempcall )
#
#
} # end of elseif
options(warn=0)
#
class(result)<-c('lm.bp', 'lm')

result
#
#
}
```

Function pbivpois

```
"pbivpois" <-
function(x, y=NULL, lambda = c(1, 1, 1), log=FALSE) {


      if ( is.matrix(x) ) {
            var1<-x[,1]
            var2<-x[,2]
      }
      else if (is.vector(x)&is.vector(y)){
            if (length(x)==length(y)){
                  var1<-x
                  var2<-y
```

```
			}
			else{
				stop('lengths of x and y are not equal')
			}
		}
		else{
			stop('x is not a matrix or x and y are not vectors')
		}
		n <- length(var1)
		logbp<-vector(length=n)
#
		for (k in 1:n){
			x0<-var1[k]
			y0<-var2[k]
			xymin<-min( x0,y0 )
			lambdaratio<-lambda[3]/(lambda[1]*lambda[2])
#
			i<-0:xymin
			sums<-		-lgamma(var1[k]-i+1)-lgamma(i+1)-
lgamma(var2[k]-i+1)+i*log(lambdaratio)
			maxsums <- max(sums)
			sums<- sums - maxsums
			logsummation<- log( sum(exp(sums)) ) + maxsums
			logbp[k]<- -sum(lambda) + var1[k] * log( lambda[1] )
+ var2[k] * log( lambda[2] ) + logsummation
		}
		if (log) { result<-    logbp }
		else     { result<-exp(logbp)  }
		result
#	end of function bivpois
}
```

Function lm.dibp

```
"lm.dibp" <-
function
(  l1,  l2,  l1l2=NULL,  l3=~1,  data,  common.intercept=FALSE,
zeroL3=FALSE,     distribution='discrete',     jmax=2,maxit=300,
pres=1e-8, verbose=getOption('verbose') )
{
options(warn=-1)
#
# definition of function call
```

```
templist<-list( l1=l1, l2=l2, l1l2=l1l2, l3=l3, data=substitute(data), common.intercept=common.intercept, zeroL3=zeroL3, distribution=distribution, jmax=jmax, maxit=maxit, pres=pres, verbose=verbose)
tempcall<-as.call( c(expression(lm.dibp), templist))
rm(templist)
#
# PARAMETERS COMMON WITH lm.bp
# l1                              : formula for the first linear predictor (of lambda1)
# l2                              : formula for the second linear predictor (of lambda2)
# l1l2                            : formula for common variables on both lambda1 and lambda2
# l3                              : formula for the third first linear predictor/covariance parameter (lambda3)
# common.intercept: logical argument defining whether common intercept should be used for lamdba1,lambda2
#
# data          : data.frame which contains data {required arguement}
# zeroL3    : Logical argument controlling whether lambda3 is zero (DblPoisson) or not
# maxit    : maximum number of iterations
# pres      : precision of the relative likelihood difference after which EM stops
# verbose   : Logical argument controlling whether beta parameters will we
#                printed while EM runs. Default value is taken options()$verbose value.
#
# PARAMETERS ADDITIONAL TO lm.bp
# distribution : Selection of diagonal inflation distribution.
#                Three choices are available:
#                ='discrete'  : Discrete, jmax is the number of diagonal elements [0,1,...,]
#                ='poisson'   : Poisson with mean theta.
#                ='geometrics': Geometric with success probability theta.
#                Default is DISCRETE(2). theta[1] and theta[2] stand for theta_1, theta_2
#                while theta_0=1-theta[1]-theta[2].
```

```
# jmax          : Used only for DISCRETE diagonal distribution
(distribution='discrete').
#                    Indicates the number of parameters of the
DISCRETE distribution.
# ---------------------------------------------------------------
# set common or noncommon intercept
if (common.intercept){ formula1.terms<-'1'  }
else {formula1.terms<-'internal.data1$noncommon' }
#
#
namex<-as.character(l1[2])
namey<-as.character(l2[2])
x<-data[,names(data)==namex]
y<-data[,names(data)==namey]
#
#
# Data length
n<-length(x)
lengthprintvec<-1
#
#
#
# definition of diagonal inflated distribution
      maxy<-max(c(x,y))
#
#  changing distribution to codes 1,2,3
      dist<-distribution
      if        ( charmatch( dist, 'poisson'   , nomatch=0) ==1 )
{distribution<-2}
      else if ( charmatch( dist, 'geometric', nomatch=0) ==1 )
{distribution<-3}
      else if ( charmatch( dist, 'discrete' , nomatch=0) ==1 )
{distribution<-1}
      if ( distribution==1 ){
                  dilabel<-paste('Inflation         Distribution:
Discrete with J=',jmax)
                  if (jmax==0) {theta<-0}
                  else         { theta<-1:jmax*0+1/(jmax+1) }
                  di.f<-function (x, theta){
                  JMAX<-length(theta)
                  if      (x>JMAX) { res<-0 }
                  else if (x==0)   { res<-1-sum(theta) }
                  else             { res<-theta[x] }
                  res
```

```
                }
      }
      else if ( distribution==2 ){
                dilabel<-'Inflation Distribution: Poisson'
                theta<-1.0;
                di.f<-function (x, theta){
                                          if   (theta>0)   {   res<-
dpois( x, theta ) }
                                          else {
                                                if (x==0) { res<-1}
                                                else {res<-1e-12}
                                          }
                }
      }
      else if ( distribution==3 ){
                dilabel<-'Inflation Distribution: Geometric'
                theta<-0.5;
                di.f<-function (x, theta){
                                          if (theta>0) {
                                                if(theta==1)
{theta<-0.9999999}
                                                res<-dgeom(       x,
theta ) }
                                          else if (theta==1){
                                                if (x==0) { res<-1}
                                                else {res<-1e-12}
                                          }
                                          else {res<-1e-12}
                             }
      }
      else {
            stop(paste(distribution, 'Not known distribution.',
sep=': '))
      }
# ------
# setting up data frames, vectors and data
#
# form dataframes used
# data1 includes modelling on lambda1 and lambda2
# data2 includes modelling on lambda3
# internal.data1 and internal.data2 are data frames used for
additional internal variables
#
internal.data1<-data.frame( y1y2=c( x, y) )
```

```
internal.data2<-data.frame( y3 = rep(0, n ) )
#
p<-length(as.data.frame(data))
data1<-rbind(data, data)
names(data1)<-names(data)
#
# removing x and y
data1<-data1[ , names(data1)!=namex]
data1<-data1[ , names(data1)!=namey]
#
#
#
# define full model
if    (as.character(l1[3])=='.')    {    l1<-formula(    paste(
as.character(l1[2]), paste( names(data1),'',collapse='+',sep=''
), sep='~')      )   }
if    (as.character(l2[3])=='.')    {    l2<-formula(    paste(
as.character(l2[2]), paste( names(data1),'',collapse='+',sep=''
), sep='~')  )   }
if (as.character(l3[2])=='.') { l3<-formula( paste( '', paste(
names(data1),'',collapse='+',sep='' ) , sep='~') )   }
#
# define the formula used for covariance term
formula2<-
formula(paste('internal.data2$y3~',as.character(l3[2]),sep=''))
#
internal.data1$noncommon<- as.factor(c(1:n*0,1:n*0+1))
contrasts(internal.data1$noncommon)<-contr.treatment(2, base=1)
internal.data1$indct1<-c(1:n*0+1,1:n*0  )
internal.data1$indct2<-c(1:n*0  ,1:n*0+1)
#
#
if (!zeroL3){
          data2<-data1[1:n,]
          names(data2)<-names(data1)
          }
#####
#
# add the common terms
#
if ( !is.null(l1l2) ) {
     formula1.terms<-paste(                    formula1.terms,
as.character(l1l2[2]),sep='+')
     }
```

```
#
# add the special common terms (if any)
# in this section we identify non-common parameters
# if a variable X is common in all formulas the we use term
x*noncommon to include x+x:noncommon terms
# otherwise use I(internal.data1$indct1*x) to add sepererate
parameter on lambda1
#
templ1<- labels(terms(l1))
#
# run this only if there are terms in l1 formula
if (length( templ1 )>0){
     for ( k1 in 1:length( templ1 ) ){
          if   (   !is.null(l1l2)   )   {   checkvar1<-
sum(labels(terms(l1l2))==templ1[k1] )==1   }
          else{ checkvar1<-FALSE }
          checkvar2<-sum(labels(terms(l2))==templ1[k1] )==1
          if    (checkvar1&checkvar2)    {formula1.terms<-
paste(formula1.terms,
paste('internal.data1$noncommon*',templ1[k1],sep=''),  sep='+')
     }
          else{
               formula1.terms<-paste(formula1.terms,
paste('+I(internal.data1$indct1*',templ1[k1],sep=''),   sep='')

               formula1.terms<-paste(formula1.terms,
')',sep='')
               }
     }
}
#
# if a variable X is not common st
# otherwise use I(internal.data1$indct1*x) to add sepererate
parameter on lambda1
#
templ2<- labels(terms(l2))
#
# run this only if there are terms in l1 formula
if (length( templ2 )>0){
     for ( k1 in 1:length( templ2 ) ){
          if    (    !is.null(l1l2)    )    {checkvar1<-
(sum(labels(terms(l1l2))==templ2[k1]
)+sum(labels(terms(l1))==templ2[k1] ))!=2      }
          else{ checkvar1<-TRUE }
```

```
            if ( checkvar1 ) {
                  formula1.terms<-paste(formula1.terms,
paste('+I(internal.data1$indct2*',templ2[k1],sep=''), sep='')
                  formula1.terms<-paste(formula1.terms,
')',sep='')
                  }
      }
}
#
rm(templ1)
rm(templ2)
rm(Checkvar1)
rm(Checkvar2)
#
# This bit creates labels for special terms of type c(x1,x2)
used in l1l2
#
#
formula1<-
formula(paste('internal.data1$y1y2~',formula1.terms,sep=''))
tmpform1<-as.character(formula1[3])
newformula<-formula1
while( regexpr('c\\(',tmpform1) != -1)
{
      temppos1<-regexpr('c\\(',tmpform1)[1]
      tempfor <-substring( tmpform1, first = temppos1+ 2 )
      temppos2<-regexpr('\\)' , tempfor)[1]
      tempvar  <-substring(  tempfor  ,  first  =  1,    last  =
temppos2-1 )
      temppos3<-regexpr(', ' , tempvar)[1]
      tempname1<-substring(tempfor      ,  first  =  1,   last  =
temppos3-1 )
      tempname2<-substring(tempfor       ,  first  =  temppos3+2,
last=temppos2-1)
      tempname2<-sub( '\\)','', tempname2  )
      tempvar1<-data[, names(data)==tempname1]
      tempvar2<-data[, names(data)==tempname2]
      data1$newvar1<-c(tempvar1, tempvar2)
#
      if( is.factor(tempvar1)& is.factor(tempvar2) ){
            data1$newvar1<-as.factor(data1$newvar1)
            if (all(levels(tempvar1)==levels(tempvar2))){
                  attributes(data1$newvar1)<-
attributes(tempvar1)}
```

```
      }
      tempvar<-sub( ', ' , '..',  tempvar )
      names(data1)[names(data1)=='newvar1']<-tempvar
      newformula<-sub( 'c\\(','', tmpform1  )
      newformula<-sub( '\\)','', newformula  )
      newformula<-sub( ', ' , '..', newformula  )
      tmpform1<-newformula
      formula1<-
formula(paste('internal.data1$y1y2~',newformula,sep=''))
}
#####
rm(temppos1)
rm(temppos2)
rm(temppos3)
rm(tmpform1)
rm(tempfor)
rm(tempvar)
rm(tempvar1)
rm(tempvar2)
rm(tempname1)
rm(tempname2)
# ------
# initial values for parameters
prob<-0.20
s<-rep(0,n)
vi<-1:n*0
v1<-1-c(vi,vi)
like<-1:n*0
zero<- ( x==0 )|( y==0 )
if   (zeroL3) { lambda3<-rep(0,n) }
else                              {   lambda3<-rep(   max(0.1,
cov(x,y,use='complete.obs')), n) }
#
#
#
#
# Initial values for lambda

internal.data1$v1<-1-c(vi,vi);

lambda<-glm(     formula1,     family=poisson,     data=data1,
weights=internal.data1$v1, maxit=100)$fitted
#
lambda1<-lambda[1:n]
```

```
lambda2<-lambda[(n+1):(2*n)]
#
difllike<-100.0
loglike0<-1000.0
i<-0
ii<-0

if (zeroL3) {
      #
      # fit double poisson diagonal inflated model
      loglike<-rep(0, maxit)
      lambda3<-1:n*0
      while ( (difllike>pres) && (i <= maxit) ) {
      i<-i+1
      #####   E step  ######
      for (j in 1:n) {
            if (zero[j]) {
                  s[j]<-0;
#                 calculation of log-likelihood
                  if (x[j]==y[j]) {
                                    density.di<-di.f( 0.0, theta )
                                    like[j]<-log(    (1-prob)*exp(-
lambda1[j]-lambda2[j])+prob*density.di );
                                    vi[j]<-prob*density.di*exp(-
like[j]) }
                  else{
                         like[j]<-log(1-
prob)+log(dpois(x[j],lambda1[j]))+log(dpois(y[j],lambda2[j]));
                                    vi[j]<-0.0 ;}
            }
            else {
                  if (x[j]==y[j]) {
                              density.di<-di.f( x[j],theta );
                              like[j]<-log(        (1-prob)*dpois(
x[j],lambda1[j] )*dpois( y[j],lambda2[j] ) + prob*density.di );
                              vi[j]    <-  prob*density.di*exp(  -
like[j] ) }
                  else  {
                              vi[j]<-0.0;
                              like[j]<-log(1-prob)+log(
dpois(x[j],lambda1[j])*dpois(y[j],lambda2[j]) )}
            }
      }
#### end of E-step #########
```

```
      x1<-x;
      x2<-y;
      loglike[i]<-sum( like ) ;
      difllike<-abs( (loglike0-loglike[i])/loglike0 )
      loglike0<-loglike[i]
      #
      #
########### M-step ############
      # estimate mixing proportion
      prob<-sum(vi)/n
      #
      # maximization of each theta parameter
      if ( distribution == 1 ) {
            # calculation of theta_j, j=1,...,jmax ; theta_0=1-
sum(theta)
            if (jmax==0) { theta<-0 }
            else {
                  for (ii in 1:jmax) {
                        temp<-as.numeric(( (x==ii) & (y==ii) ));
                        theta[ii]<-sum(temp*vi)/sum(vi)
                        }
                  }
      }
      else if (distribution==2){
                  # calculation of theta for poisson diagonal
inflation
                  theta<- sum(vi*x)/sum(vi) }
      else if (distribution==3){
                  # calculation of theta for geometric diagonal
inflation
                  theta<- sum(vi)/( sum(vi*x)+sum(vi) ) }
      #
      #     fit model on lambda1 & lambda2
      #
      internal.data1$v1<- 1-c(vi,vi);
      internal.data1$v1[
(internal.data1$v1<0)&(internal.data1$v1>-1.0e-10) ]<-0.0
#
      x1[(x1<0)&(x1>-1.0e-10)]<-0.0
      x2[(x2<0)&(x2>-1.0e-10)]<-0.0
      internal.data1$y1y2<-c(x1,x2)
      m<-glm(     formula1,     family=poisson,     data=data1,
weights=internal.data1$v1 , maxit=100)
```

```
      p3<-length(m$coef)
      beta<-m$coef
# ---------------------------------------------------
#     creating names for parameters
      names(beta)<-newnamesbeta( beta )
#
#     end of name creations (l1, l2, l2-l1, blank)
#  -------------------------------------------------
      betaparameters<-splitbeta( beta )
#
      lambda<-fitted(m)
      lambda1<-lambda[1:n]
      lambda2<-lambda[(n+1):(2*n)]
      #
      #####   end of M step  ######
#
#  printing also beta
      if (verbose) {
            printvec<- c( i,beta,100.0*prob, theta,  loglike[i],
difllike );
            names(printvec)<-c(       'iter',       names(beta),
'Mix.p(%)',    paste(    'theta',    1:length(theta),sep=''    ),
'loglike', 'Rel.Dif.loglike')
            }
#  limited print out
      else {
            printvec<-  c(  i,  100.0*prob,  theta,   loglike[i],
difllike );
            names(printvec)<-c(     'iter','Mix.p(%)',     paste(
'theta',       1:length(theta),sep=''       ),       'loglike',
'Rel.Dif.loglike')
      }
      lengthprintvec<-length(printvec)
      print.default( printvec, digits=4 )
      }

#
#     calculation of BIC and AIC for double poisson model
      if ( (distribution==1)&&(jmax==0) ){noparams<- m$rank +1}
      else                                  {noparams<- m$rank +
length( theta ) +1}
      AIC<- -2*loglike[i] + noparams * 2
      BIC<- -2*loglike[i] + noparams * log(2*n)
#
```

```
#
#    Calculation of BIC, AIC of Poisson saturated model
     x.mean<-x
     x.mean[x==0]<-1e-12
     y.mean<-y
     y.mean[y==0]<-1e-12
     AIC.sat <-  sum(log( dpois( x , x.mean ) ) + log( dpois(
y , y.mean ) ))
     BIC.sat <-  -2 * AIC.sat + (2*n)* log(2*n)
     AIC.sat <-  -2 * AIC.sat + (2*n)* 2
#
#
     AICtotal<-c(AIC.sat, AIC);
     BICtotal<-c(BIC.sat, BIC );
     names(AICtotal)<-c('Saturated', 'DblPois')
     names(BICtotal)<-c('Saturated', 'DblPois')
#
     allbeta<-c(betaparameters$beta1,betaparameters$beta2)
     names(allbeta)<-c(          paste(           '(l1):',
names(betaparameters$beta1),     sep=''      ),paste('(l2):',
names(betaparameters$beta2), sep='' ) )
     allparameters<-c(allbeta, prob, theta)
     if     (distribution==1){     names(allparameters)<-c(
names(allbeta), 'p', paste('theta', 1:length(theta),sep='') ) }
     else   {names(allparameters)<-c(   names(allbeta),   'p',
'theta') }
#
#    calculation of fitted values
#  ---------------------------
     fittedval1<-(1-prob)*m$fitted[1:n]
     fittedval2<-(1-prob)*m$fitted[(n+1):(2*n)]
#
     meandiag<-0
     if   ((distribution==1)&&(jmax>0))   {   meandiag<-sum(
theta[1:jmax]*1:jmax ) }
     else if (distribution==2) { meandiag<-theta }
     else if (distribution==3) { meandiag<- (1-theta)/theta }
#
     fittedval1[x==y]<-prob*meandiag + fittedval1[x==y]
     fittedval2[x==y]<-prob*meandiag + fittedval2[x==y]
#
     result<-list(coefficients=allparameters,
```

```
      fitted.values=data.frame(x=fittedval1,y=fittedval2),
residuals=data.frame(x=x-fittedval1,y=y-fittedval2),
                            beta1=betaparameters$beta1,
beta2=betaparameters$beta2,         p=prob,         theta=theta,
diagonal.distribution=dilabel,
                            lambda1=m$fitted[1:n],
lambda2=m$fitted[(n+1):(2*n)],      loglikelihood=loglike[1:i],
parameters=noparams, AIC=AICtotal,
                            BIC=BICtotal,iterations=i          ,
call=tempcall)
#
#
# end of diagonal inflated double poisson model
}
else {
      loglike<-rep(0,maxit)
      while ( (difllike>pres) && (i <= maxit) ) {
      i<-i+1
      #####   E step  ######
      for (j in 1:n) {
            if (zero[j]) {
                  s[j]<-0;
#                 calculation of log-likelihood
                  if (x[j]==y[j]) {
                        density.di<-di.f( 0.0, theta )
                        like[j]<-  log(  (1-prob)*exp(-lambda1[j]-
lambda2[j]-lambda3[j])+prob*density.di );
                        vi[j]<-prob*density.di*exp(-like[j]) }
                  else{
                           like[j]<-log(1-prob)-lambda3[j]
+log(dpois(x[j],lambda1[j]))
                  +log(dpois(y[j],lambda2[j]));
                        vi[j]<-0.0 ;}
            }
            else {
                  lbp1<-pbivpois(x[j]-1,                  y[j]-1,
lambda=c(lambda1[j],lambda2[j],lambda3[j]),  log=TRUE );
                  lbp2<-pbivpois(x[j]         ,    y[j]          ,
lambda=c(lambda1[j],lambda2[j],lambda3[j]), log=TRUE );
                  s[j]<-exp( log(lambda3[j]) + lbp1 - lbp2 );
#                 like[j]<-lbp2;
                  if (x[j]==y[j]) {
                              density.di<-di.f( x[j],theta );
```

```
                            like[j]<-log(  (1-prob)*exp(lbp2)  +
prob*density.di );
                            vi[j]    <-  prob*density.di*exp(  -
like[j] ) }
                 else  {
                            vi[j]<-0.0;
                            like[j]<-log(1-prob)+lbp2 }
           }
      }
#### end of E-step #########
      x1<-x-s;
      x2<-y-s;
      loglike[i]<-sum( like ) ;
      difllike<-abs( (loglike0-loglike[i])/loglike0 )
      loglike0<-loglike[i]
      #
      #
########### M-step ############
      # estimate mixing proportion
      prob<-sum(vi)/n
      #
      # maximization of each theta parameter
      if ( distribution == 1 ) {
           # calculation of theta_j, j=1,...,jmax ; theta_0=1-
sum(theta)
#          cat (c('1:discrete, jmax=', jmax), '\n')
           if (jmax==0){ theta<-0}
           else{
                 for (ii in 1:jmax) {
                      temp<-as.numeric(( (x==ii) & (y==ii) ));
                      theta[ii]<-sum(temp*vi)/sum(vi)
#                     print(    c(ii,    sum(temp),    sum(vi),
sum(temp*vi) ) )
                      }
#                     cat( c('2:discrete, jmax=', jmax), '\n')
                 }
      }
      else if (distribution==2){
                 #  calculation  of  theta  for  poisson  diagonal
inflation
                 theta<- sum(vi*x)/sum(vi) }
      else if (distribution==3){
#     else {
```

```
                # calculation of theta for geometric diagonal
inflation
                theta<- sum(vi)/( sum(vi*x)+sum(vi) ) }
      #    fit model on lambda3
      internal.data2$v1<- 1-vi;
      internal.data2$v1[
(internal.data2$v1<0)&(internal.data2$v1>-1.0e-10) ]<-0.0
#
      internal.data2$y3<-s;
      m0<-glm(     formula2,     family=poisson,     data=data2,
weights=internal.data2$v1 , maxit=100)
      beta3<-m0$coef
      lambda3<-m0$fitted
      #
      #    fit model on lambda1 & lambda2
      internal.data1$v1<- 1-c(vi,vi);
      internal.data1$v1[
(internal.data1$v1<0)&(internal.data1$v1>-1.0e-10) ]<-0.0
#
      x1[(x1<0)&(x1>-1.0e-10)]<-0.0
      x2[(x2<0)&(x2>-1.0e-10)]<-0.0
      internal.data1$y1y2<-c(x1,x2)
      m<-glm(     formula1,     family=poisson,     data=data1,
weights=internal.data1$v1 , maxit=100)
      p3<-length(m$coef)
      beta<-m$coef
#  ----
#     creating names for parameters
      names(beta)<-newnamesbeta( beta )
#  ----
      lambda<-fitted(m)
      lambda1<-lambda[1:n]
      lambda2<-lambda[(n+1):(2*n)]
      #
      #####   end of M step  ######
#
#     print all parameters including beta
      if (verbose) {
            printvec<-   c(   i,beta,beta3,100.0*prob,   theta,
loglike[i], difllike );
            names(printvec)<-c(                           'iter',
names(beta),paste('l3_',names(beta3),sep=''),       'Mix.p(%)',
paste(    'theta',    1:length(theta),sep=''    ),    'loglike',
'Rel.Dif.loglike')
```

```
          }
#
#  limited print out
     else {
          printvec<-  c(  i,  100.0*prob,  theta,   loglike[i],
difllike );
          names(printvec)<-c(   'iter',   'Mix.p(%)',   paste(
'theta',      1:length(theta),sep=''       ),       'loglike',
'Rel.Dif.loglike')
          }
#
     lengthprintvec<-length(printvec)
     print.default( printvec, digits=4 )
     }
#
#    calculation of BIC and AIC for bivpoisson model
     if  (  (distribution==1)&&(jmax==0)  ){noparams<-  m$rank  +
m0$rank + 1}
     else                                {noparams<- m$rank +
m0$rank + length( theta ) +1}
     AIC<- -2*loglike[i] + noparams * 2
     BIC<- -2*loglike[i] + noparams * log(2*n)
#
#
#    Calculation of BIC, AIC of Poisson saturated model
     x.mean<-x
     x.mean[x==0]<-1e-12
     y.mean<-y
     y.mean[y==0]<-1e-12
     AIC.sat <-  sum(log( dpois( x , x.mean ) ) + log( dpois(
y , y.mean ) ))
     BIC.sat <-  -2 * AIC.sat + (2*n)* log(2*n)
     AIC.sat <-  -2 * AIC.sat + (2*n)* 2
#
#
     AICtotal<-c(AIC.sat, AIC);
     BICtotal<-c(BIC.sat, BIC );
     names(AICtotal)<-c('Saturated', 'BivPois')
     names(BICtotal)<-c('Saturated', 'BivPois')
#  -------
#
#    spliting parameter vector
     betaparameters<-splitbeta( beta )
#
```

```
#     putting all betas in one vector
      allbeta<-c(betaparameters$beta1,betaparameters$beta2,
beta3)
      names(allbeta)<-c(             paste(             '(l1):',
names(betaparameters$beta1),       sep=''       ),paste('(l2):',
names(betaparameters$beta2),       sep=''       ),paste('(l3):',
names(beta3), sep='' ) )
      allparameters<-c(allbeta, prob, theta)
      if       (distribution==1){       names(allparameters)<-c(
names(allbeta), 'p', paste('theta', 1:length(theta),sep='') ) }
      else    {names(allparameters)<-c(    names(allbeta),    'p',
'theta') }
#
#     calculation of fitted values
#  ----------------------------
      fittedval1<-(1-prob)*(m$fitted[1:n]         + lambda3)
      fittedval2<-(1-prob)*(m$fitted[(n+1):(2*n)] + lambda3)
#
      meandiag<-0
      if    ((distribution==1)&&(jmax>0))    {    meandiag<-sum(
theta[1:jmax]*1:jmax ) }
      else if (distribution==2) { meandiag<-theta }
      else if (distribution==3) { meandiag<- (1-theta)/theta }
#
      fittedval1[x==y]<-prob*meandiag + fittedval1[x==y]
      fittedval2[x==y]<-prob*meandiag + fittedval2[x==y]
#
#
#     saving output
      result<-list(coefficients=allparameters,
fitted.values=data.frame(x=fittedval1,y=fittedval2),
residuals=data.frame(x=x-fittedval1,y=y-fittedval2),
                 beta1=betaparameters$beta1,
beta2=betaparameters$beta2, beta3=beta3,   p=prob,  theta=theta,
diagonal.distribution=dilabel,
                 lambda1=m$fitted[1:n],
lambda2=m$fitted[(n+1):(2*n)],                   lambda3=lambda3,
loglikelihood=loglike[1:i],
                 parameters=noparams,               AIC=AICtotal,
      BIC=BICtotal,iterations=i , call=tempcall)
#
} # end of elseif
#
options(warn=0)
```

```
class(result)<-c('lm.dibp', 'lm')
#
result
#
#
}
```

Function newnamesbeta

```
"newnamesbeta" <-
function( bvec ) {
#    Internal  function  for  renaming  parameters  according  to
their interpretation

      names(bvec)<-sub('\\)','',names(bvec))
                       #remove right parenthesis

      names(bvec)<-
sub('\\(Intercept','(Intercept)',names(bvec))
      # replace "(Intercept" with "(Intercept)"
      names(bvec)[pmatch('internal.data1$noncommon2',names(bvec
))]<-'(l2-l1):(Intercept)' #                              replace
'internal.data1$noncommon2' with 'l2-l1' for intercept
      names(bvec)<-sub('internal.data1\\$noncommon2:','(l2-
l1):',names(bvec))               #  the  same  for  the  rest  of
parameters
      names(bvec)<-
sub('internal.data1\\$noncommon0:','(l1):',names(bvec))
            # replace 'internal.data1\\$noncommon0:' by '(l1)'
      names(bvec)<-
sub('internal.data1\\$noncommon1:','(l2):',names(bvec))
            # replace 'internal.data1\\$noncommon1:' by '(l2)'

      names(bvec)<-sub(':internal.data1\\$noncommon2','(l2-
l1):',names(bvec))              #  same  as  above  with  ":"  in
front of expressions
      names(bvec)<-
sub(':internal.data1\\$noncommon0','(l1):',names(bvec))
      names(bvec)<-
sub(':internal.data1\\$noncommon1','(l2):',names(bvec))

      names(bvec)<-sub('I\\(internal.data1\\$indct1          \\*
','(l1):',names(bvec))          #                          replace
'I(internal.data1$indct1 * ' with '(l1):'
```

```
      names(bvec)<-sub('I\\(internal.data1\\$indct2          \\*
','(l2):',names(bvec))          #                      replace
'I(internal.data1$indct2 * ' with '(l2):'

      names(bvec)
}
```

Function splitbeta

```
"splitbeta" <-
function( bvec ){

#     Internal function for spliting beta parameters according
to their interpretation
#
      p3<-length(bvec)

      indx1<-grep(  '\\(l1\\):',  names(bvec)  )  #  identify
parameters for lambda1
      indx2<-grep(  '\\(l2\\):',  names(bvec)  )  #  identify
parameters for lambda2
      indx3<-grep(  '\\(l2-l1\\):',  names(bvec)  )  #  identify
difference parameters for lambda2
#
#     create temporary labels to identify common parameters
      tempnames<-sub( '\\(l2-l1)\\:', 'k', names(bvec)  )
      tempnames<-sub( '\\(l2)\\:', 'k', tempnames  )
      tempnames<-sub( '\\(l1)\\:', 'k', tempnames  )

      indx4<-tempnames%in%names(bvec) # common parameters are
identified as TRUE
#
      beta1<-c(bvec[indx4],bvec[indx1])
      beta2<-c(bvec[indx4],bvec[indx3],bvec[indx2])
      indexbeta2<-c(  rep(0,sum(indx4)),  rep(1,length(indx3)),
rep(2,length(indx2)) )

      names(beta1)<-sub('\\(l1\\):','',names(beta1))
      names(beta2)<-sub('\\(l2\\):','',names(beta2))
      names(beta2)<-sub('\\(l2-l1\\):','',names(beta2))

      beta1<-beta1[order(names(beta1))]
      indexbeta2<-indexbeta2[order(names(beta2))]
      beta2<-beta2[order(names(beta2))]
```

```
	ii<-1:length(beta2)
	ii<-ii[indexbeta2==0]
	for ( i in ii ) {
#		beta2[i]<-sum(	beta2[	grep(	names(beta2)[i],
names(beta2) ) ] )
		beta2[i]<-sum(  beta2[  names(beta2)[i]==names(beta2)
] )
	}
	beta2<-beta2[indexbeta2%in%c(0,2)]

	btemp<-list(beta1=beta1,beta2=beta2)
	btemp
}
```

Main Part

```
#code
sl=read.csv("data/sl.csv",stringsAsFactors=T)
levels(sl[,2])
#Evaluation of Covariates
attach(sl)
fit1=glm(g1~rat1+penbox1+goalbox1+corner1,family="poisson",data
=sl)
summary(fit1)
fit2=glm(g2~rat2+penbox2+goalbox2+corner2,family="poisson",data
=sl)
summary(fit2)
cor1=read.csv("data/correlation1.csv")
cor1
C=cor(cor1)
rownames(C)=c("Rating","PenaltyBox","GoalBox","Corner")
colnames(C)=c("Rating","PenaltyBox","GoalBox","Corner")
C
#Fitting the bivariate Poisson model
biv=lm.bp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+goal
box2+corner2,l1l2=NULL,data=sl)
biv$coefficients
biv$parameters
biv$iterations
biv$loglikelihood
biv$lambda1
```

```
biv$lambda2
biv$lambda3
biv$fitted.values
plot(biv$fitted.values[,1],biv$fitted.values[,2],main="Expected
Goals",xlab="Home Team",ylab="Away Team")
plot(sl[,3],sl[,4])
plot(infg$loglikelihood)
biv$AIC
biv$BIC
plot(1:biv$iterations,biv$loglikelihood,xlab="Iterations",ylab=
"Log-Likelihood")
dbp=lm.bp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+goal
box2+corner2,l1l2=NULL,data=sl,zeroL3=TRUE)
dbp$AIC
dbp$BIC
#Fitting the Diagonal Inflated Bivariate Poisson model
(geometric)
infg=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+g
oalbox2+corner2,l1l2=NULL,data=sl,distribution="geometric")
infg$coefficients
infg$fitted.values
infg$diagonal.distribution
infg$loglikelihood
infg$AIC
infg$BIC
##Fitting the Diagonal Inflated Bivariate Poisson model
(Discrete)
inf1=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+g
oalbox2+corner2,l1l2=NULL,data=sl,jmax=1)
inf1$coefficients
inf1$diagonal.distribution
inf1$loglikelihood
inf1$AIC
inf1$BIC
#Fitting the Inflated Double Poisson model
infdp=lm.dibp(g1~rat1+penbox1+goalbox1+corner1,g2~rat2+penbox2+
goalbox2+corner2,l1l2=NULL,data=sl,zeroL3=TRUE,jmax=1)
infdp$coefficients
infdp$fitted.values
infdp$loglikelihood
infdp$AIC
infdp$BIC
sum=rbind(c(biv$parameters,-1029.576,        2081.151,2133.271
,0),c(inf1$parameters,-1029.576,2085.153  ,2146.749  ,  1.305e-
```

```
02),c(infg$parameters,-1029.576,2085.151   ,2146.747   ,1.680e-05),c(infdp$parameters,-1030.476,2084.952 ,2141.810,4.602e-02))
rownames(sum)=c("Bivariate       Poisson","Inflated       with Discrete(1)","Inflated   with   Geometric","Inflated   Double-Poisson")
colnames(sum)=c("Parameters","Loglikelihood","AIC","BIC","Mix.Prop(p)")
sum
#Karlis and Ntzoufras model
slsc=read.csv("data/sl_scores.csv",stringsAsFactors=T)
slsc
form1=~c(team1,team2)+c(team2,team1)
bivsc=lm.bp(g1~1,g2~1,l1l2=form1,data=slsc)
bivsc$coefficients
bivsc$AIC
bivsc$BIC
#comparison
comp=rbind(c(-1.029576e+03,              2081.151,2133.271),c(-1.098e+03,2269.030,2444.342))
rownames(comp)=c("Bivariate Poisson","Bivariate Poisson (goals as cov)")
colnames(comp)=c("Loglikelihood","AIC","BIC")
comp
#PREDICTION
ratA=(141+169+146+123+237+98+234+169+174+202+174+106+162)/13;ratA
penboxA=(3+1+3+4+1+6+2+4+3+1+3+5+3)/13;penboxA
goalboxA=(0+0+0+0+0+1+0+1+2+1+0+0+0)/13;goalboxA
cornerA=(1+3+5+5+4+1+3+2+4+1+11+2+5)/13;cornerA
ratP=(114+127+172+181+84+153+144+121+194+118+153+162+127)/13;ratP
penboxP=(0+2+4+9+1+2+2+4+4+2+1+5+0)/13;penboxP
goalboxP=(1+0+1+2+2+2+2+1+1+0+0+1+1)/13;goalboxP
cornerP=(3+4+5+9+5+2+1+3+0+2+6+8+2)/13;cornerP
l1=exp(-1.281071082+0.008715946*ratA+0.024295455*penboxA+0.100234116*goalboxA-0.030432260*cornerA);l1
l2=exp(-1.705548672+0.009189555*ratP+0.087018910*penboxP+0.197741080*goalboxP-0.048838609*cornerP);l2
l3=0.0665655;l3
pred=bivpois.table(8,8,lambda=c(l1,l2,l3));pred
sum(diag(pred))
print(sum(pred[lower.tri(pred)]))
```

```
print(sum(pred[upper.tri(pred)]))
#VERIFICATION
x=0.3004748+0.4240807+0.2754379
x
```

A3. The Newton-Raphson method

The Newton-Raphson method which is named after Isaac Newton and Joseph Raphson, is an iterative technique for finding the root in functions when this cannot be found in a straightforward way.
Let us consider the non-linear equation,

$$x: f(x) = 0$$

By starting with some value x_0, the method computes a sequence of approximations $x_1, x_2, \ldots$ which converge to the solution x^* ($f(x^*) = 0$) of the non-linear equation.
We start from the Taylor expansion of function f around the point x_n,

$$f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n) + \frac{(x_{n+1} - x_n)^2}{2} f''(x_n) + \cdots$$

If we neglect the higher order terms, we find

$$f(x_{n+1} = f(x_n) + (x_{n+1} - x_n)f'(x_n)$$

If we then require $f(x_{n+1})$ to be equal to zero, we obtain

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

There fore,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

$$\vdots$$

The Newton-Raphson method is generalized for the case of systems with n equations with n unknowns. We may write the system

$$\begin{cases} f_1(x_1, \dots, x_n) = 0 \\ f_2(x_1, \dots, x_n) = 0 \\ \quad\vdots \\ f_n(x_1, \dots, x_n) = 0 \end{cases}$$

We consider $f: X \subseteq \mathbb{R}^n \to \mathbb{R}^n$ defined as $f(x) = f_1(x), \dots, f_2(x)$

We want to find a vector $r = (r_1, \dots, r_n)$ such that $f(r) = 0$. To approximate such a vector r, we may make an initial guess $x_0 \in \mathbb{R}^n$. If f is differentiable, then we know that $y = f(x)$ is approximated by the equation

$$y = f(x_0) + Df(x_0)(x - x_0)$$

where $Df(x_0)$ is the $n \times n$ matrix of the first derivative of f.

We set $y = 0$ in order to find where this approximating function is zero. Thus, we solve the matrix equation

$$f(x_0) + Df(x_0)(x_1 - x_0) = 0$$

with x_1 giving a revised approximation to the root r. Evidently the equation above is equivalent to

$$Df(x_0)(x_1 - x_0) = -f(x_0)$$

To continue our argument, suppose that $Df(x_0)$ is an invertible $n \times n$ matrix. Then we multiply the equation by $[Df(x_0)]^{-1}$ to obtain

$$I_n(x_1 - x_0) = -[Df(x_0)]^{-1}f(x_0).$$

Similarly to the one-variable case of the method of the Newton-Raphson method, we may iterate the formula to define a sequence $\{x_k\}$ of vectors by,

$$x_k = x_{k-1} - [Df(x_0)]^{-1}f(x_{k-1})$$