Bradley Terry model for Serie A championship

Savarese Camilla 1838890

A.Y. 2021/2022

Content

Theoretical overview of the Bradley Terry model with latent variables

The Bradley Terry Model is used to describe probabilities of possible outcomes when there is a set of K elements which are compared to each other in pairs. A parameter $\lambda_l > 0$ is associated to each element for $l \in 1, 2, ...K$ and it represents its skill rating. Let's denote λ the set of all theλ_i. We denote by D the associated data: each of them represents the (binary) result of the k-th comparison between i and j.

We also define:

- w_{ij} : the number of comparisons where i beats j
- $w_i = \sum_{j=1}^{K} w_{ij}$: the number of wins of i
- $n_{ij} = w_{ij} + w_{ji}$: the number of matches between i and j

Since in the original version proposed in 1952 the model was:

$$\Pr(i \text{ beats } j) = \frac{\lambda_i}{\lambda_i + \lambda_j}$$

we can state that

$$D_{kij}|\lambda \sim Ber(\frac{\lambda_i}{\lambda_i + \lambda_j})$$

where $k = 1, 2, .., n_{ij}$ and $1 \leq i < j \leq K$.

Additionally, according to (Gormley and Murphy, 2009; Guiver and Snelson, 2009) we can assign a prior Gamma distribution to each λ_i:

$$\lambda_i \sim \Gamma(a, b)$$

.

Now we want to introduce some latent variables which are such that the resulting complete log likelihood admits a simple form. Thanks to the Thurstonian interpretation for each pair $1 \leq i < j \leq K$ and for each associated comparison $k = 1, 2, .., n_{ij}$ let $Y_{ki} \sim \mathcal{E}(\lambda_i)$, $Y_{kj} \sim \mathcal{E}(\lambda_j)$ be exponential random variable so that

$$\Pr(Y_{ki} < Y_{kj}) = \frac{\lambda_i}{\lambda_i + \lambda_j}$$

These latent variables can be interpreted as arrival times and the individual with the lowest arrival time wins. However, instead of introducing these variables, we introduce for each pair i, j the latent random variable

$$Z_{ij} = \sum_{k=1}^{n_{ij}} min(Y_{ki}, Y_{kj})$$

From the properties of the exponential distribution we know that

$$min(Y_{ki}, Y_{kj}) \sim \mathcal{E}(\lambda_i + \lambda_j)$$

We also know that the sum of identically distributed exponential random variables is a Gamma distributions with shape given by the number of variables and rate equal to the exponential rate so in our case:

$$Z_{ij} \sim \Gamma(n_{ij}, \lambda_i + \lambda_j)$$

The (conditional) of the latent variables $Z = Z_{ij}$ for $1 \leq i < j \leq K$ such that $n_{ij} > 0$ is

$$p(z|D, \lambda) = \prod_{1 \leq i < j \leq K | n_{ij} > 0} \Gamma(z_{ij}; n_{ij}, \lambda_i + \lambda_j)$$

Now we have all the elements to obtain the complete likelihood of λ:

$$L(\lambda, z) = f(D, z_{11}, .., z_{KK}|\lambda) = \prod_{1 \leq i < j \leq K} \prod_{k=1}^{n_{ij}} f(d_{kij}, z_{ij}|\lambda)$$

$$= \prod_{1 \leq i < j \leq K} \prod_{k=1}^{n_{ij}} f(d_{kij}|\lambda) f(z_{ij}|D, \lambda)$$

Remembering that

$$D_{kij}|\lambda \sim Ber(\frac{\lambda_i}{\lambda_i + \lambda_j})$$

for the first factor we have

$$\prod_{k=1}^{n_{ij}} f(d_{kij}|\lambda) = \prod_{k=1}^{n_{ij}} \frac{\lambda_i^{d_{kij}}}{(\lambda_i + \lambda_j)^{d_{kij}}} \cdot \frac{\lambda_j^{1-d_{kij}}}{(\lambda_i + \lambda_j)^{1-d_{kij}}} = \frac{\lambda_i^{w_{ij}} \cdot \lambda_j^{w_{ji}}}{(\lambda_i + \lambda_j)^{n_{ij}}}$$

where we can just rewrite the production in the numerator as

$$\prod_{1 \leq i < j \leq K} \lambda_i^{w_{ij}} \cdot \lambda_j^{w_{ji}} = \prod_{i=1}^{K} \lambda_i^{w_i}$$

Inserting tha Gamma (conditional) distribution for the Z_{ij} we obtain as "final form":

$$L(\lambda, z) = \prod_{i=1}^{K} \lambda_i^{w_i} \prod_{1 \leq i < j \leq K | n_{ij} > 0} \frac{1}{(\lambda_i + \lambda_j)^{n_{ij}}} \cdot \frac{(\lambda_i + \lambda_j)^{n_{ij}}}{\Gamma(n_{ij})} \cdot z_{ij}^{(n_{ij}-1)} \cdot e^{-(\lambda_i + \lambda_j) z_{ij}}$$

that is quite simple and easy to handle.

Our target is the posterior distribution of each λ_i, that is proportional to the prior times the likelihood:

$$\pi(\lambda_i|z,D) \propto \pi(\lambda_i)L(\lambda,z) \propto \lambda_i^{a-1}e^{-\lambda_i b}\lambda_i^{w_i} \prod_{1\leq i<j\leq K|n_{ij}>0} e^{-(\lambda_i\lambda_j)z_{ij}} \prod_{1\leq j<i\leq K|n_{ij}>0} e^{-(\lambda_i\lambda_j)z_{ij}}$$

$$= \lambda_i^{(w_i+a)-1} \cdot \exp(-\lambda_i[b + \sum_{i<j|n_{ij}>0} z_{ij} + \sum_{j<i|n_{ij}>0} z_{ji}])$$

So we obtain that $\lambda_i|z,D$ is distributed (up to a proportionality constant) as:

$$\lambda_i|z,D \sim \Gamma(a+w_i, b + \sum_{i<j|n_{ij}>0} z_{ij} + \sum_{j<i|n_{ij}>0} z_{ji})$$

Thanks to the introduction of these latent Z variables, we are able to derive our Gibbs samplers.

At iteration t we have (always considering $1 \leq i < j \leq K$ such that $n_{ij} > 0$):

$$Z_{ij}|D,\lambda^{(t-1)} \sim \Gamma(n_{ij}, \lambda_i^{(t-1)} + \lambda_j^{(t-1)})$$

and for $i = 1,..,K$

$$\lambda_i^{(t)}|D,Z^{(t)} \sim \Gamma(a+w_i, b + \sum_{i<j|n_{ij}>0} z_{ij}^{(t)} + \sum_{j<i|n_{ij}>0} z_{ji}^{(t)})$$

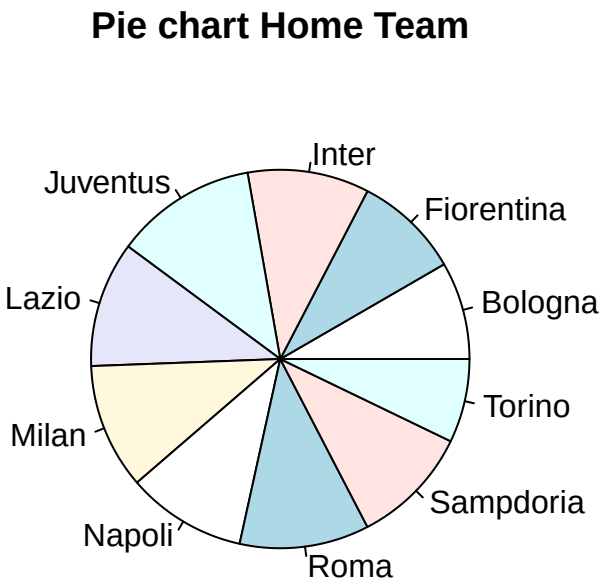
Application: Serie A championship

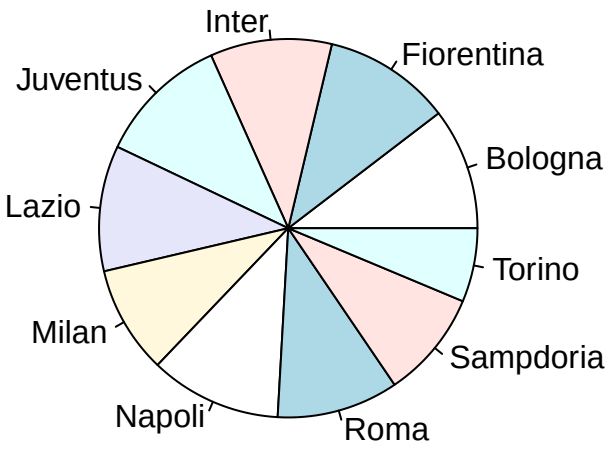
Data

The Dataset used for the application of Bradley Terry model contains 3800 Serie A matches from 2009 to 2019. Every year 20 teams were part of the competition, and there are "exchanges" between Serie A and Serie B so some are not always present. I decided to choose only the 10 teams that have always participated and that have also been present over the years 2020/2022, to be able to compare the results. In this way I was left with 812 rows. From the initial dataset I have selected only the variables that interest me:"Date", "HomeTeam", "AwayTeam", "FTR". The last one is the final result: H if home team won, A if the away team won and D if the match ended in a draw.

In the model described so far, we have not considered the ties, so for now we eliminate them from the dataset, remaining with 582 matches. Obviously we must consider that in this way some teams could have a lower number of comparisons than the others.

Luckily, looking at the plot, the number of games for each team is fairly uniform.





At this point, we want to estimate λ_i for $i = 1, .., 10$ that represents the skill rating of each time, so we mean it as a measure of how strong a team is and how much they can get to the top of the classification. These are the 10 teams we are considering:

```
##  [1] "Napoli"     "Roma"       "Lazio"      "Milan"      "Inter"
##  [6] "Fiorentina" "Bologna"    "Sampdoria"  "Torino"     "Juventus"
```

We need to calculate all the quantities needed for our model based on the data:

- n_{ij} , the number of games between i and j:

```
n<-as.matrix(table(data2$HomeTeam,data2$AwayTeam))
nij<-n+t(n)
```

For example, these are the matches that Milan has played with all the other teams:

```
##    Bologna Fiorentina       Inter    Juventus       Lazio
##         13          15          14          19          11

##     Milan    Napoli       Roma Sampdoria    Torino
##         0        12          13         14         6
```

- w_{ij} and w_{ji} , the number of games i won against j. To get them we need to compute the two matrix w_home and w_away, that contain matches which are won by only the home (or away) team:

```
#wij
home<-subset(data2, FTR=="H")
w_home<-table(home$HomeTeam,home$AwayTeam)
away<-subset(data2, FTR=="A")
w_away<-table(away$HomeTeam,away$AwayTeam)
wij<-w_home+t(w_away)
wji<-t(wij)
```

These are the matches that Milan has won with all the other teams:

```
##    Bologna Fiorentina      Inter   Juventus      Lazio
##         12          9          4          5          7

##     Milan    Napoli      Roma Sampdoria    Torino
##         0         3         6         9         5
```

- w_i the total number of games won by team i. We show it for each team.

```
##    Bologna Fiorentina      Inter   Juventus      Lazio
##         24         43         66        108         57

##     Milan    Napoli      Roma Sampdoria    Torino
##        60        86        78        41        24
```

To implement the the Gibbs sampling algorithm I have to choose the hyperparameter for each λ_i, and I simply set $a = b = 0.01$ to have a flat prior.

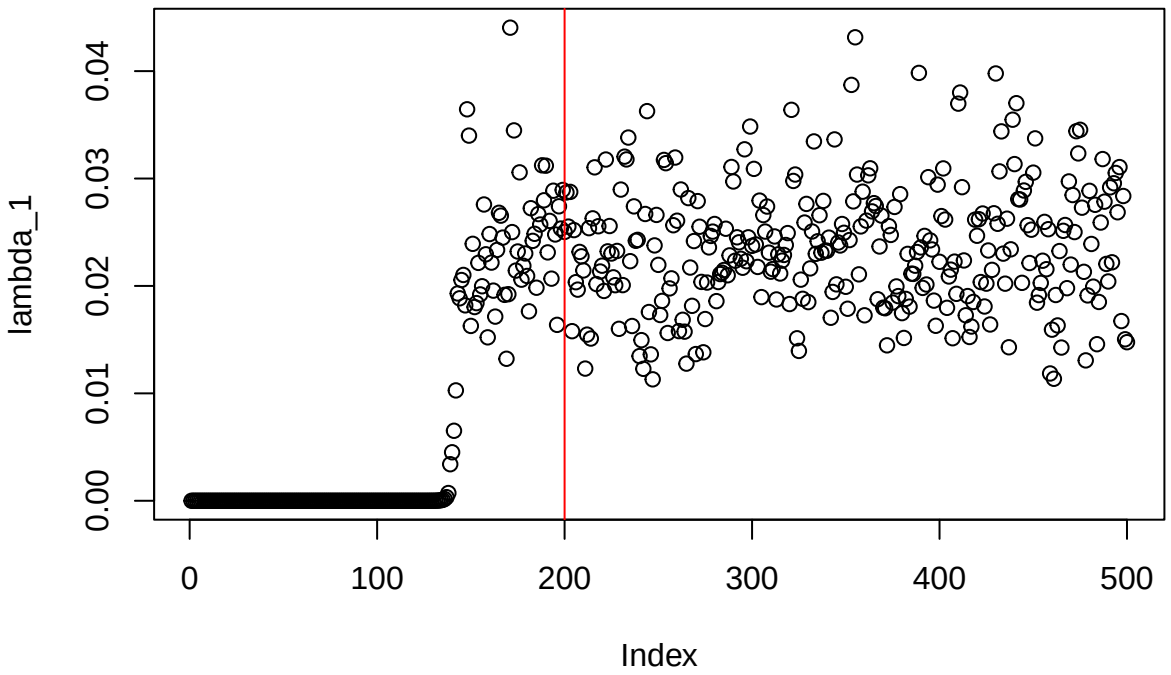
I also set the number of iteration $T = 10.000$ and a set.seed to be able to reproduce my results.

```
set.seed(1234)
N=length(data2$Date)
T<-10000
samples<-matrix(NA,nrow = T, ncol = 10)
colnames(samples)=c(paste0("lambda[",1:10,"]"))
```

```
a<-0.01
b<-0.01
lambda<-c(rgamma(10,shape = a, rate=b))
z<-matrix(0,10,10)
```

```
for(t in 1:T){

  #zij time t
  for(i in 1:(10-1)){  #i<j
    for(j in (i+1):10){
      z[i,j]=rgamma(1,shape=nij[i,j],rate=lambda[i]+lambda[j])
    }
  }

  for(i in 1:10) {
    lambda[i]=rgamma(1,shape=a+wi[i],rate=b+sum(z[i,])+sum(z[,i]))
  }

  lambda=lambda/sum(lambda) #normalization lambda

  samples[t,]=lambda
  }
```

To choose the burn-in time, I plotted the first values for λ_1 and decided to delete the first 200 iterations.


lambda_1
0.00
0.01
0.02
0.03
0.04
0
100
200
300
400
500
Index

Now we can calculate the posterior means for the parameters λ_i ad get our estimates:

```
##       [,1]          [,2]
##  [1,] "Bologna"    "0.023"
##  [2,] "Fiorentina" "0.049"
##  [3,] "Inter"      "0.093"
##  [4,] "Juventus"   "0.287"
##  [5,] "Lazio"      "0.07"
##  [6,] "Milan"      "0.089"
##  [7,] "Napoli"     "0.172"
##  [8,] "Roma"       "0.132"
##  [9,] "Sampdoria"  "0.048"
## [10,] "Torino"     "0.037"
```

How do we evaluate how good they are? We have some options.

The first one is compared our estimates with team's ELO (from the creator Arpad Elo). The ELO rating is a deterministic measure of the team's strength,and is used in various sports such as chess or American football.

It is defined such that the difference in the ratings between two players serves as a predictor of the outcome of a match, as an estimate for the expected score E_{ij} between them:

$$E_{ij} = \frac{1}{1 + 10^{\frac{ELO_j - ELO_i}{400}}}$$

Two players with equal ratings who play against each other have an equal chance of winning.

Searching the web, I found these scores calculated for the teams that participated in the 2022 Italian football championship, and therefore we can compare the two orders:

```
## [1] "Posterior means ordered"
```

```
##       [,1]          [,2]
##  [1,] "Juventus"   "0.287"
##  [2,] "Napoli"     "0.172"
##  [3,] "Roma"       "0.132"
##  [4,] "Inter"      "0.093"
##  [5,] "Milan"      "0.089"
##  [6,] "Lazio"      "0.07"
##  [7,] "Fiorentina" "0.049"
##  [8,] "Sampdoria"  "0.048"
##  [9,] "Torino"     "0.037"
## [10,] "Bologna"    "0.023"
```

```
## [1] "Elo score ordered"
```

```
##       [,1]         [,2]
##  [1,] "Juventus"   "2666"
##  [2,] "Inter"      "2569"
##  [3,] "Roma"       "2564"
##  [4,] "Lazio"      "2557"
##  [5,] "Napoli"     "2510"
##  [6,] "Torino"     "2380"
##  [7,] "Bologna"    "2363"
##  [8,] "Milan"      "2363"
##  [9,] "Fiorentina" "2345"
## [10,] "Sampdoria"  "2313"
```

We can observe that the teams' strength estimated by the Bradley Terry model are not ordered exactly in the same way of the ELO. The first and third position coincide, and in general we have that the same teams occupy the top and bottom of the ranking.

To quantify the difference between the two ranking, we can use the Kendall Tau distance, which is based on the number of agreements and disagreements :

$$\tau = \frac{(concordant\ pairs) - (discordant\ pairs)}{n(n-1)/2}$$

```
## [1] "Kendall distance posterior means-Elo score: "
```

```
## tau = 0.874, 2-sided pvalue =1.1921e-07
```

Considering that the maximum is 1, it is an acceptable value.

The differences are mainly due to the fact that the Elo are updated to 2022 while in the dataset we arrive at 2019.

To "go beyond" this problem, we can compare the order of the λ with the final rankings of the years $2020, 2021$ and 2022.

2019-2020

```
## [1] "Kendall distance posterior means-classification 2019-2020: "
```

```
## tau = 0.644, 2-sided pvalue =0.012266
```

2020-2021

```
## [1] "Kendall distance posterior means-classification 2020-2021: "
```

```
## tau = 0.6, 2-sided pvalue =0.020045
```

2021-2022

```
## [1] "Kendall distance posterior means-classification 2021-2022: "
```

```
## tau = 0.467, 2-sided pvalue =0.073638
```

These results confirm what has been said: the more time passes compared to 2019 (the last year for which we have the data), the more the predicted values deviate from the true ones.

Diagnostics

We can compute the Effective Sample Size of each Markov Chain, that is the number of independent samples with the same estimation power as the $T = 10.000$ autocorrelated samples. They are all smaller than half of T.

```
##   ESS[1]   ESS[2]   ESS[3]   ESS[4]   ESS[5]
## 4476.223 3644.453 2530.167 1317.065 3158.576
```

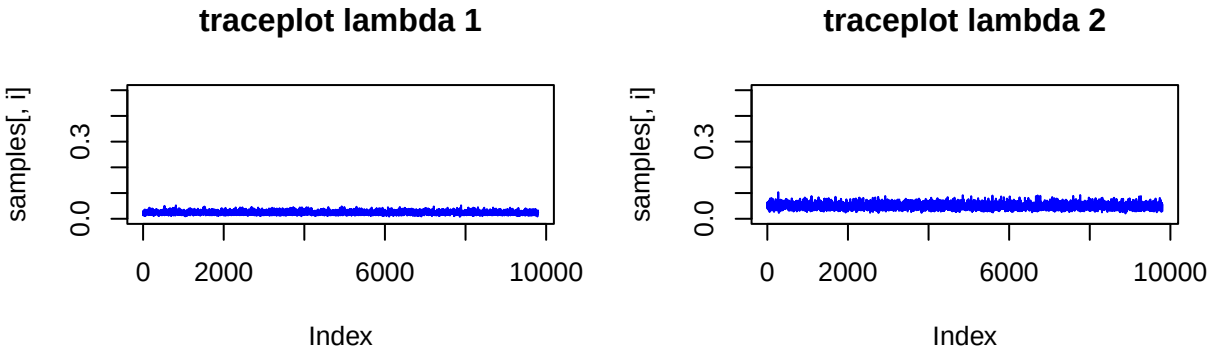
```
##   ESS[6]   ESS[7]   ESS[8]   ESS[9]  ESS[10]
## 2858.448 1852.179 2272.231 3585.162 4436.495
```

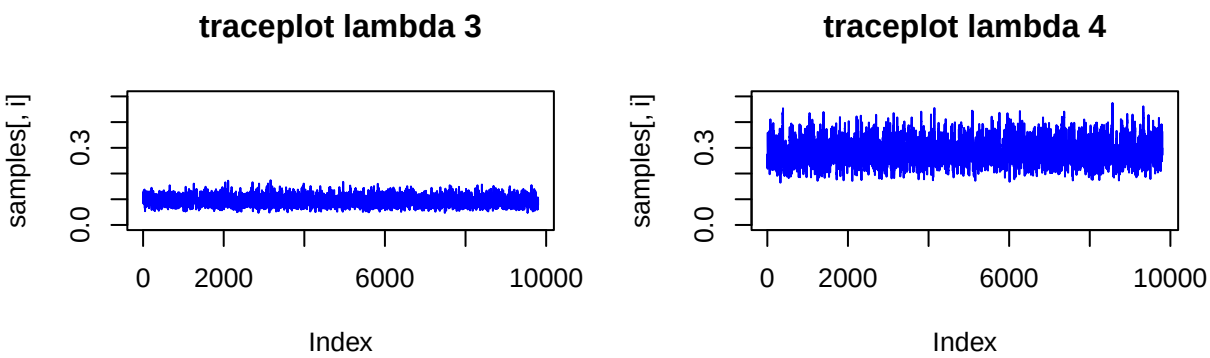
We also compute the Monte Carlo Markov Chain errors, that are almost zero:

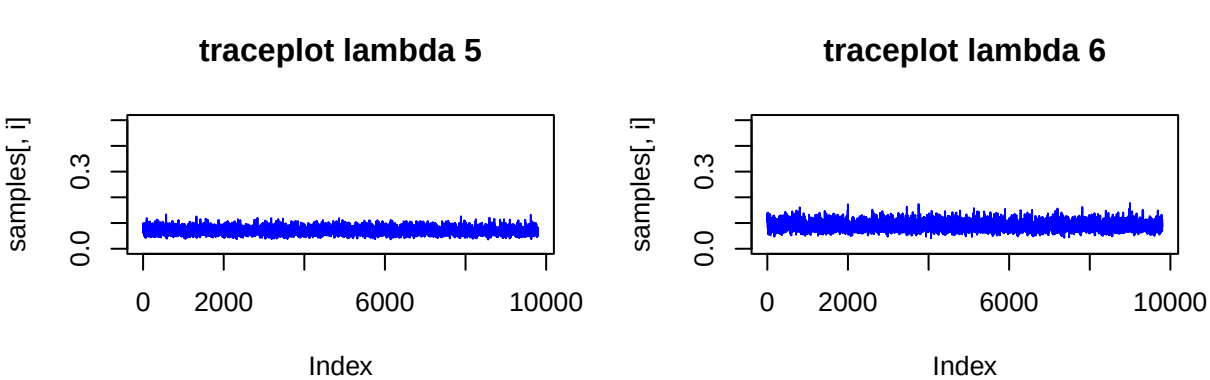
```
## MCSE[1] MCSE[2] MCSE[3] MCSE[4] MCSE[5]
## 0.00010 0.00021 0.00045 0.00164 0.00030
```

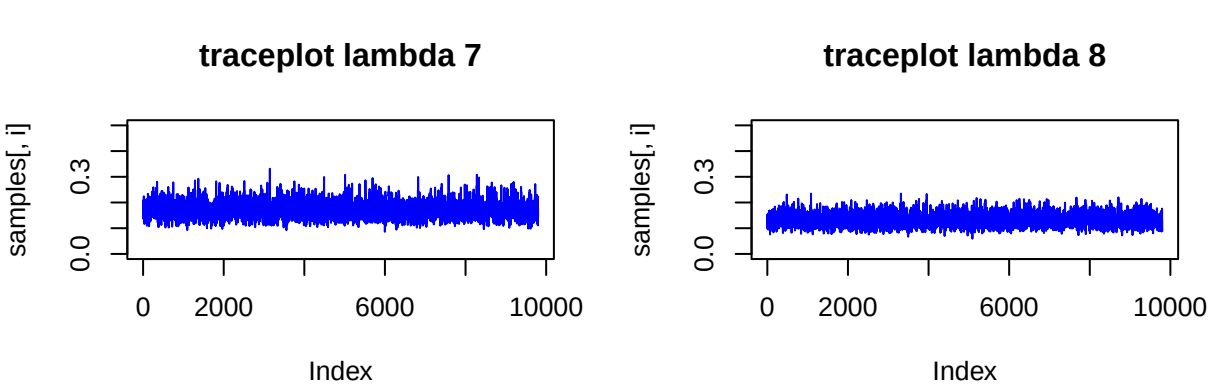
```
##  MCSE[6]  MCSE[7]  MCSE[8]  MCSE[9] MCSE[10]
##  0.00043  0.00093  0.00065  0.00021  0.00018
```

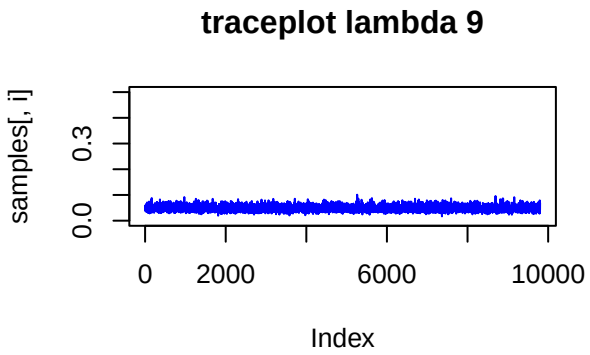
We plot the traceplot, to see if we have convergence and therefore a stationary behavior. Thanks to the Burn-in = 200 we get the desired results.

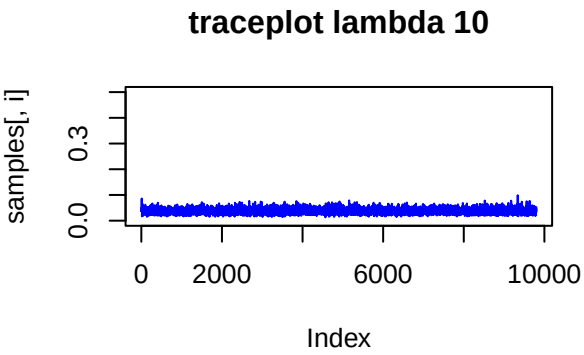




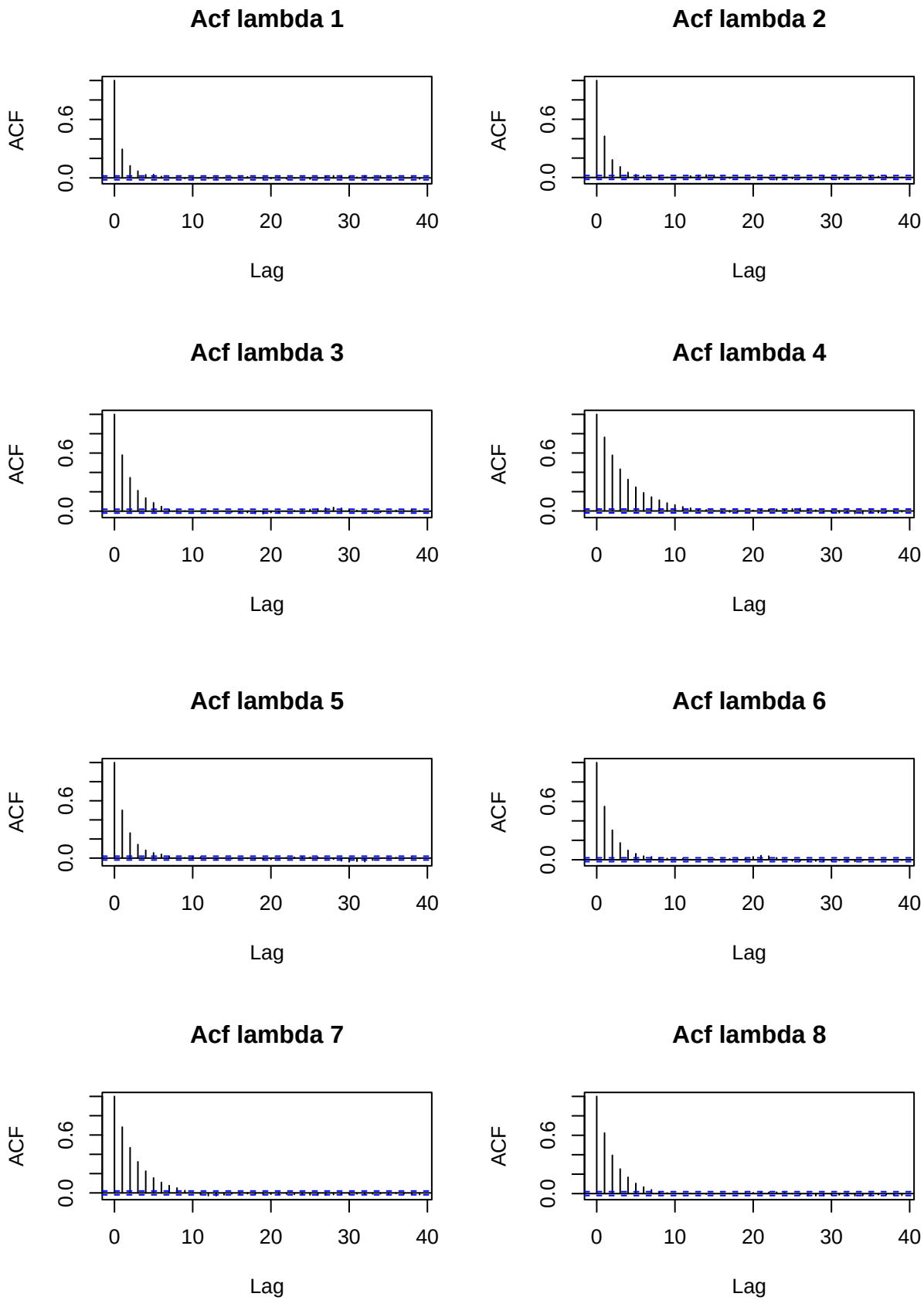


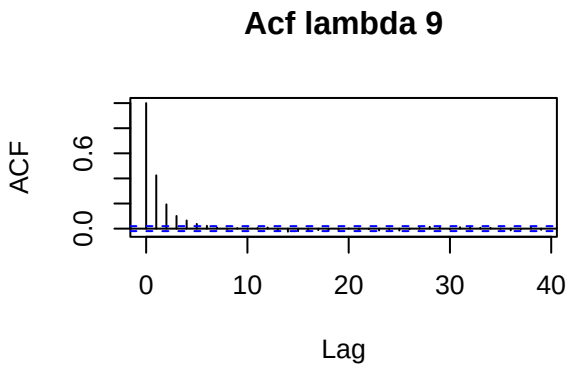


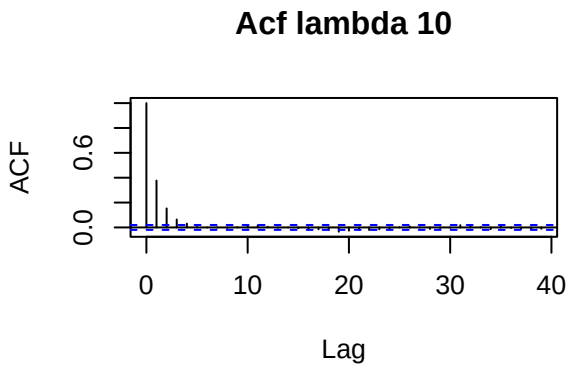

traceplot lambda 9
samples[, i]
0.0
0.3
0
2000
6000
10000
Index


traceplot lambda 10
samples[, i]
0.0
0.3
0
2000
6000
10000
Index

These are also the autocorrelation functions, to see if there is serial correlation. As expected, all autocorrelations decrease as the lag t increases, until they reach practically zero values.




Acf lambda 9
ACF
0.0
0.6
0
10
20
30
40
Lag


Acf lambda 10
ACF
0.0
0.6
0
10
20
30
40
Lag

Theoretical overview of the Bradley Terry model with ties

If we want to allow for ties in pairwise comparison, we have to refer to the model proposed by Rao and Kupper in 1967. With respect to the original version (1952) we have that :

$$\Pr(i \text{ beats } j) = \frac{\lambda_i}{\lambda_i + \theta\lambda_j} = p_1$$

$$\Pr(j \text{ beats } i) = \frac{\lambda_i}{\theta\lambda_i + \lambda_j} = p_3$$

$$\Pr(i \text{ ties } j) = \frac{(\theta^2 - 1)\lambda_i\lambda_j}{(\theta\lambda_i + \lambda_j)(\lambda_i + \theta\lambda_j)} = p_2$$

So that we can state that

$$D_{kij}|\lambda, \theta \sim Multinomial(1, (p_1, p_2, p_3))$$

The parameter θ is define for values of $\theta > 1$. If it's small, the probability of having a tie is almost 0, while if θ tends to infinity this probability tends to 1. The prior distribution for λ is the same Gamma as before. Furthermore, we suppose independent prior for the λ_i and θ. In the model it's state that if $\bar{\theta} = \theta - 1$ we have:

$$\bar{\theta} \sim \Gamma(a_\theta, b_\theta)$$

The (conditional) distribution of the latent variables is:

$$p(z|D, \lambda, \theta) = \prod_{1 \le i \ne j \le K | s_{ij} > 0} \Gamma(z_{ij}; s_{ij}, \lambda_i + \theta\lambda_j)$$

where $s_{ij} = w_{ij} + t_{ij}$ and t_{ij} is the number of ties between i and j.

Combining the previous equations we can get the likelihood: Now we have all the elements to obtain the complete likelihood of λ:

$$l(\lambda, \theta, z) = log(f(D, z_{11}, .., z_{KK}|\lambda, \theta)) =$$

$$\sum_{1 \le i \ne j \le K | s_{ij} > 0} s_{ij} log(\lambda_i) + \frac{t_{ij}}{2} log(\theta^2 - 1) + (s_{ij} - 1) log(z_{ij}) - (\lambda_i + \theta\lambda_j) z_{ij} + log(\Gamma(s_{ij}))$$

From the likelihood which we obtain the full conditional (posterior) of λ_i and θ:

$$\pi(\lambda_i|z, D, \theta) \propto \pi(\lambda_i) L(\lambda, \theta, z)$$

$$\pi(\theta|z, D, \theta, \lambda) \propto \pi(\theta) L(\lambda, \theta, z)$$

To obtain the full conditional of the λ_i the reasoning is similar to that of the original model:

$$\lambda_i|z, D, \theta \sim \Gamma(a + \sum_{i \neq j} s_{ij}, b + \sum_{i \neq j | s_{ij} > 0} z_{ij} + \theta \sum_{j \neq i | s_{ij} > 0} z_{ji})$$

while for θ we get:

$$\theta|z, D, \lambda \propto (\theta^2 - 1)^T (\theta - 1)^{a_\theta - 1} \exp(-(b_\theta + \sum_{1 \leq i \neq j \leq K | s_{ij} > 0} \lambda_j z_{ij})\theta) \mathbb{1}_{(\theta > 1)}$$

where $T = \frac{1}{2} \sum_{1 \leq i \neq j \leq K} t_{ij}$ represents the total number of ties.

Also in this case, thanks to the introduction of the latent Z variables, we are able to derive our Gibbs samplers. At iteration t we have (always considering $1 \leq i \neq j \leq K$ such that $s_{ij} > 0$):

$$Z_{ij}^{(t)}|D, \lambda^{(t-1)}, \theta^{(t-1)} \sim \Gamma(s_{ij}, \lambda_i^{(t-1)} + \theta^{(t-1)} \lambda_j^{(t-1)})$$

and for $i = 1, .., K$

$$\lambda_i^{(t)}|D, Z^{(t)}, \theta^{(t-1)} \sim \Gamma(a + \sum_{i \neq j} s_{ij}, b + \sum_{i \neq j | s_{ij} > 0} z_{ij}^{(t)} + \theta^{(t-1)} \sum_{j \neq i | s_{ij} > 0} z_{ji}^{(t)})$$

$$\theta^{(t)}|D, Z^{(t)}, \lambda^{(t-1)} \sim \pi(\theta|D, Z^{(t)}, \lambda^{(t-1)})$$

Application: Serie A championship with ties

We consider the same data as before, this time considering also the ties. We have 812 matches. The n_{ij}, w_{ij}, w_i are computed in the same way as in the previous mode.

We also need t_{ij} , the number of ties between i and j and T:

```
## [1] 10
```

```
#tij
ties<-subset(data, FTR == "D" )
t<-as.matrix(table(ties$HomeTeam,ties$AwayTeam))
tij<-t+t(t)

T<-0.5*(sum(tij))
#wij
home<-subset(data, FTR=="H")
w_home<-table(home$HomeTeam,home$AwayTeam)
away<-subset(data, FTR=="A")
w_away<-table(away$HomeTeam,away$AwayTeam)
wij<-w_home+t(w_away)
wji<-t(wij)
#wi
wi<-apply(wij,1,sum)
```

Finally $s_{ij} = w_{ij} + t_{ij}$:

```
#sij
sij=wij+tij
```

To implement the the Gibbs sampling algorithm I have to choose the hyperparameter for each λ_i , and I simply set $a = b = 0.01$ as before, while for θ I choose $a_\theta = 0.1$,$b_\theta = 0.1$ as in the paper.

Since $(\theta^2 - 1)^T$ is quite a huge number, and we can have problem from a computational point of view, the idea is to use a Metropolis Hastings step in order to sample θ values with a normal random walk proposal with standard deviation 0.1 and then take the logarithm of the acceptance rule to avoid the above problem.

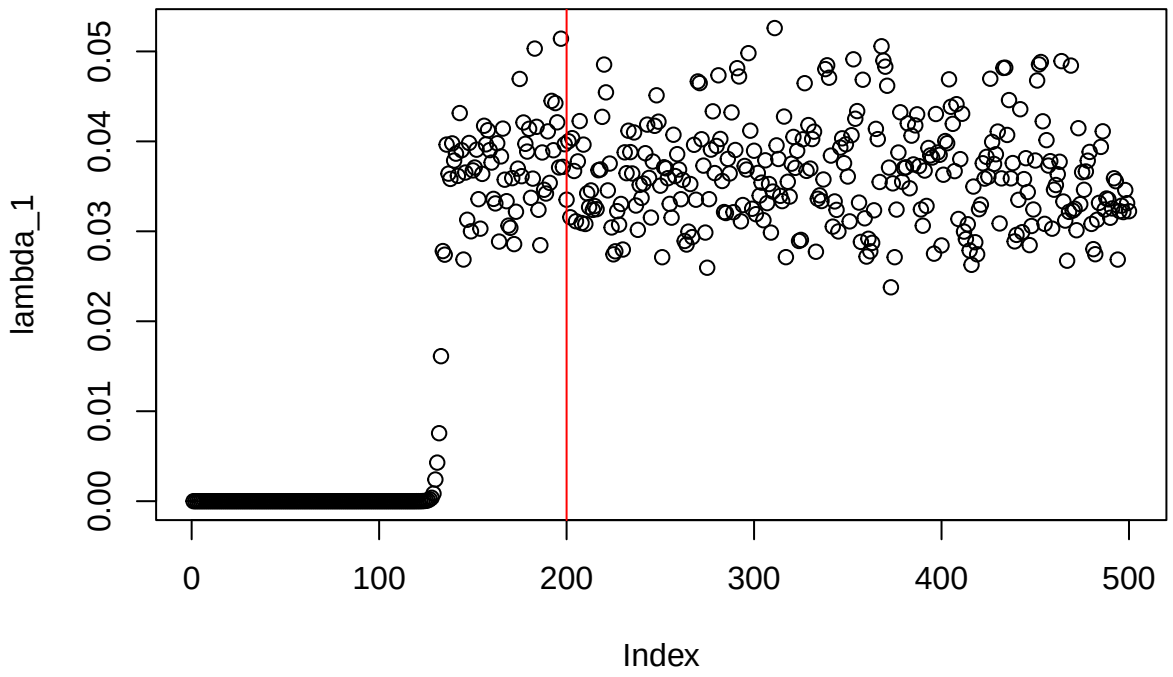
```
set.seed(1234)
N=length(data$Date)
Tg<-10000
```

```
samples<-matrix(NA,nrow = Tg, ncol = k+1) #10 lambda, 1 theta
colnames(samples)=c(paste0("lambda[",1:k,"]"),
                            paste0("theta"))

a<-0.01
b<-0.01
lambda<-c(rgamma(k,shape = a, rate=b))

a_t<-0.1
b_t<-0.1
theta<-rgamma(1,shape = a_t,rate = b_t)+1
z<-matrix(0,10,10)

#pi
logtarget =  function(theta, z, lambda, a_t, b_t) {
T*(log(theta^2-1))+(a_t-1)*(log(theta-1))-(b_t+apply(z,2,sum)%*%lambda)*th
}
#p
logproposal= function(x,mu){
  dnorm(x,mean=mu,sd=0.1,log = T)
}
```

```
#Gibbs cycle
for(t in 1:Tg){

  for(i in 1:k){
    for(j in 1:k){ if (i!=j) {
      z[i,j]=rgamma(1,sij[i,j],lambda[i]+theta*lambda[j])
    }
    }
  }

  for(i in 1:k) {
    lambda[i]=rgamma(1,a+sum(sij[i,]),b+sum(z[i,])+theta*sum(z[,i]))
  }

  lambda=lambda/sum(lambda)
  samples[t,1:k]=lambda

  x=theta  # state of the chain at the current time
```

```
  #draw a candidate for the next state of the chain
  y = rtruncnorm(1,a=1,mean=x,sd=0.1) # constraint on theta
  ## acceptance/rejection auxiliary draw
  o = runif(1,min=0,max=1)

  #acceptance rule
  ACCEPT=(log(o)<logtarget(y,z,lambda, a_t, b_t)-logtarget(theta,z,lambda,
          +logproposal(x,mu=y)-logproposal(y,mu=x))

  theta=samples[t,k+1] <- ifelse(ACCEPT,y,theta)
  }
```

Also this time,to choose the burn-in time, I plotted the first values for λ_1 and decided to delete the first 200 iterations.



As before, we can calculate the posterior means for the parameters λ_i ad get our estimates:

```
##       [,1]          [,2]
##  [1,] "Bologna"     "0.037"
##  [2,] "Fiorentina" "0.059"
##  [3,] "Inter"       "0.1"
##  [4,] "Juventus"    "0.252"
```

```
##  [5,] "Lazio"      "0.074"
##  [6,] "Milan"      "0.089"
##  [7,] "Napoli"     "0.156"
##  [8,] "Roma"       "0.126"
##  [9,] "Sampdoria"  "0.054"
## [10,] "Torino"     "0.052"
## [11,] "theta"      "1.932"
```

We look at the comparison with the ELO score.

```
## [1] "Posterior means ordered"
```

```
##       [,1]         [,2]
##  [1,] "Juventus"   "0.252"
##  [2,] "Napoli"     "0.156"
##  [3,] "Roma"       "0.126"
##  [4,] "Inter"      "0.1"
##  [5,] "Milan"      "0.089"
##  [6,] "Lazio"      "0.074"
##  [7,] "Fiorentina" "0.059"
##  [8,] "Sampdoria"  "0.054"
##  [9,] "Torino"     "0.052"
## [10,] "Bologna"    "0.037"
```

```
## [1] "Elo score ordered"
```

```
##       [,1]         [,2]
##  [1,] "Juventus"   "2666"
##  [2,] "Inter"      "2569"
##  [3,] "Roma"       "2564"
##  [4,] "Lazio"      "2557"
##  [5,] "Napoli"     "2510"
##  [6,] "Torino"     "2380"
##  [7,] "Bologna"    "2363"
##  [8,] "Milan"      "2363"
##  [9,] "Fiorentina" "2345"
## [10,] "Sampdoria"  "2313"
```

We can observe that the teams' strength estimated by the Bradley Terry model are not ordered exactly in the same way of the ELO. As in the previous model, the first and third positions coincide, and in general we have that the same teams occupy the top and bottom of the ranking.

```
## tau = 0.111, 2-sided pvalue =0.72051
```

Also this time we compare and quantify the order of the λ with the final rankings of the years $2020, 2021$ and 2022.

2019-2020

```
## [1] "Kendall distance posterior means-classification 2019-2020: "
```

```
## tau = -0.111, 2-sided pvalue =0.72051
```

2020-2021

```
## [1] "Kendall distance posterior means-classification 2020-2021: "
```

```
## tau = -0.0667, 2-sided pvalue =0.85803
```

2021-2022

```
## [1] "Kendall distance posterior means-classification 2021-2022: "
```

```
## tau = 0.156, 2-sided pvalue =0.59151
```

In this case results are worst than the previous model.

Diagnostic

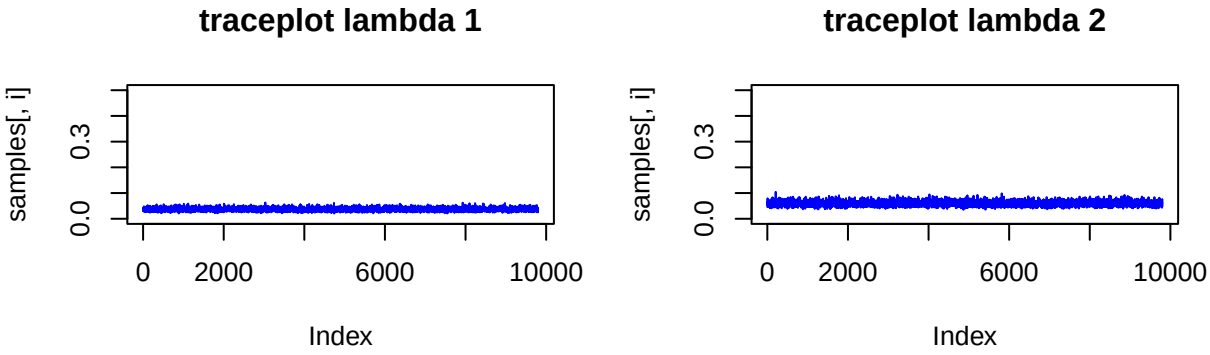
We can compute the Effective Sample Size of each Markov Chain. Note that they are all smaller than half of T. The smaller one (last one) correspond to θ.

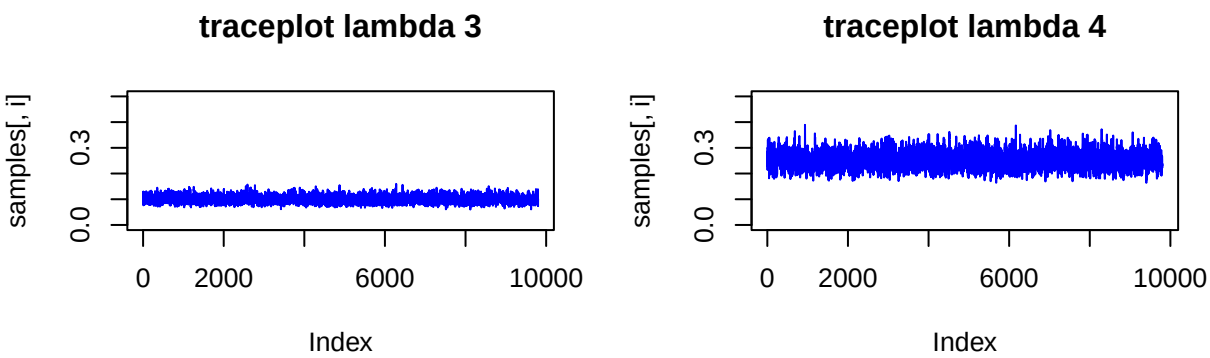
```
##   ESS[1]   ESS[2]   ESS[3]   ESS[4]   ESS[5]
## 3819.069 3332.609 2380.884 1823.892 3308.467

##   ESS[6]   ESS[7]   ESS[8]   ESS[9]  ESS[10]
## 3010.827 2375.077 2785.175 3214.543 3004.384

##  ESS[11]
## 1018.116
```

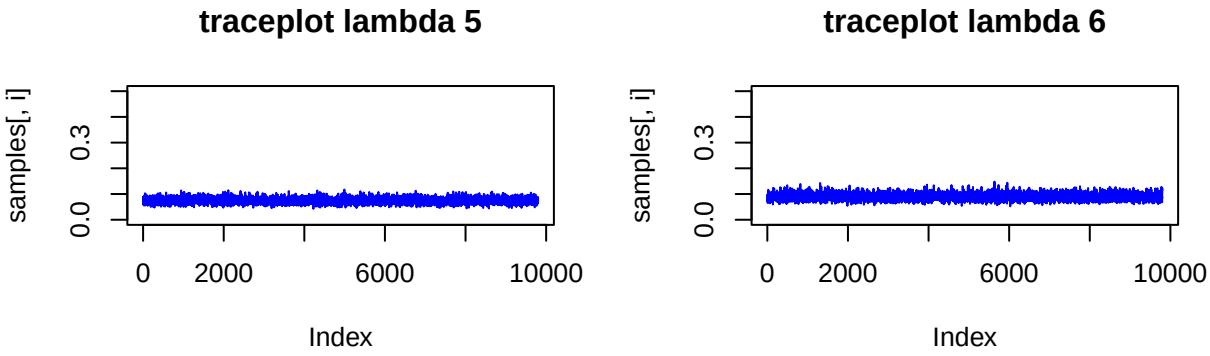
We also compute the Monte Carlo Markov Chain errors, that are almost zero:

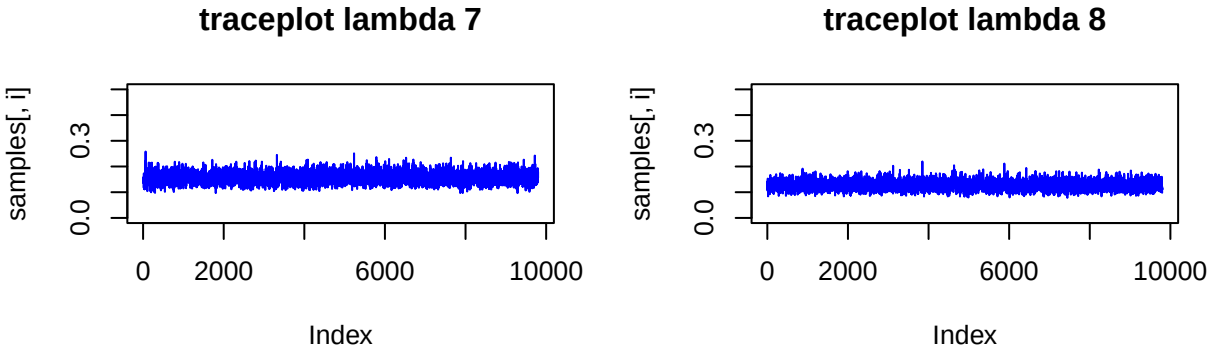
```
## MCSE[1] MCSE[2] MCSE[3] MCSE[4] MCSE[5]
## 0.00012 0.00019 0.00031 0.00095 0.00023

##  MCSE[6]  MCSE[7]  MCSE[8]  MCSE[9] MCSE[10]
##  0.00029  0.00055  0.00042  0.00019  0.00020

## MCSE[11]
##  0.00338
```

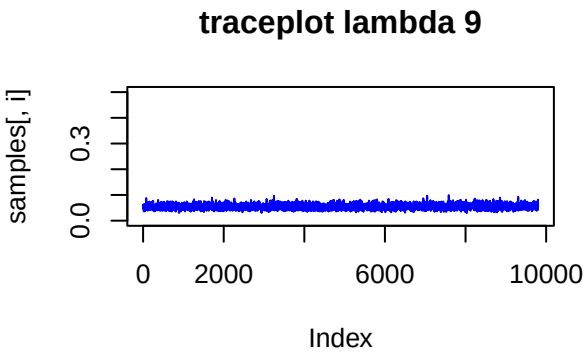
We plot the traceplot, to see if we have convergence and therefore a stationary behavior. Thanks to the Burn-in = 200 we get the desired results.

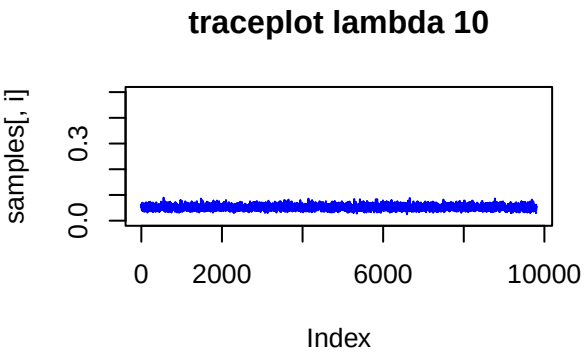


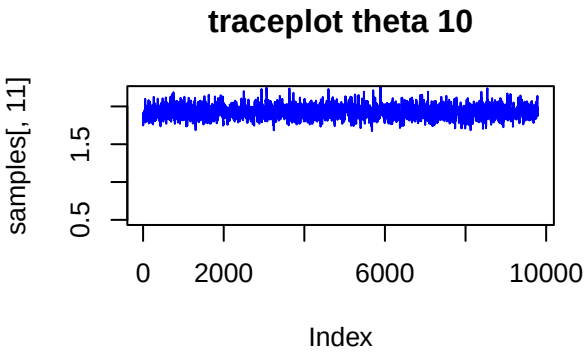




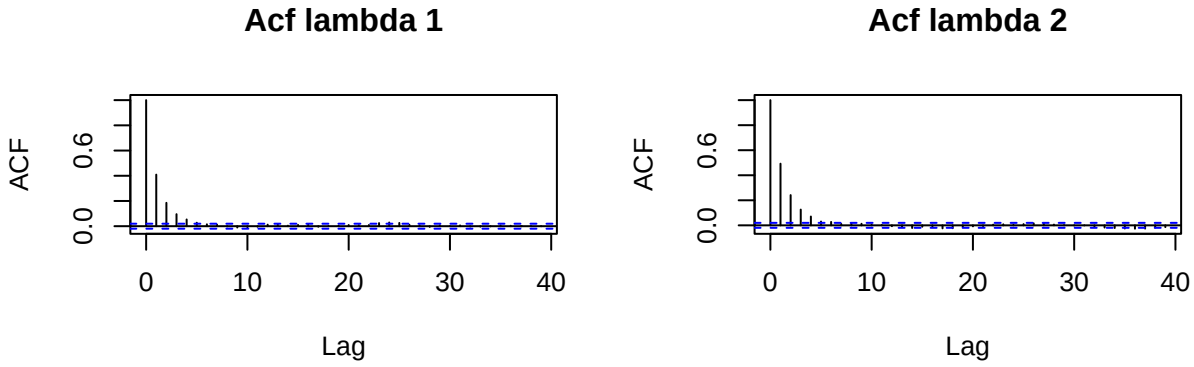


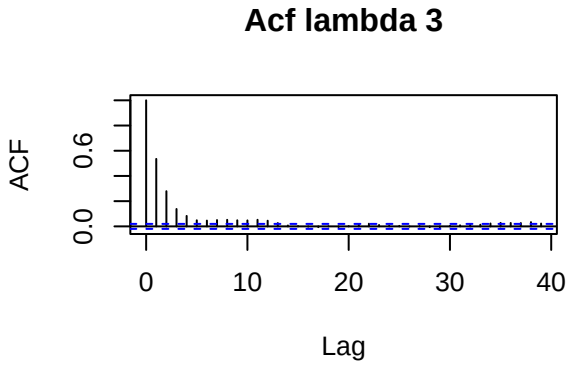

traceplot lambda 9
samples[, i]
0.0
0.3
0
2000
6000
10000
Index

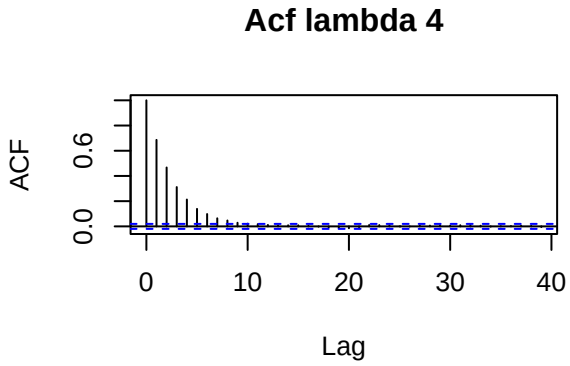

traceplot lambda 10
samples[, i]
0.0
0.3
0
2000
6000
10000
Index

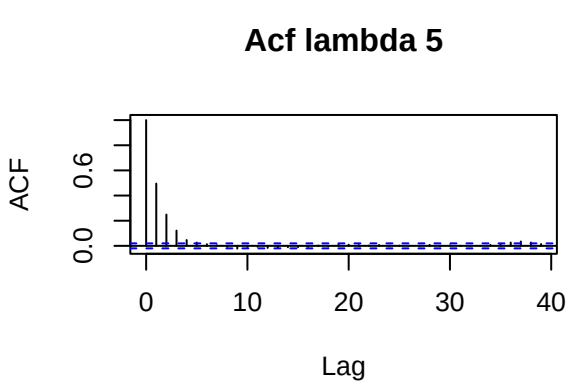

traceplot theta 10
samples[, 11]
0.5
1.5
0
2000
6000
10000
Index

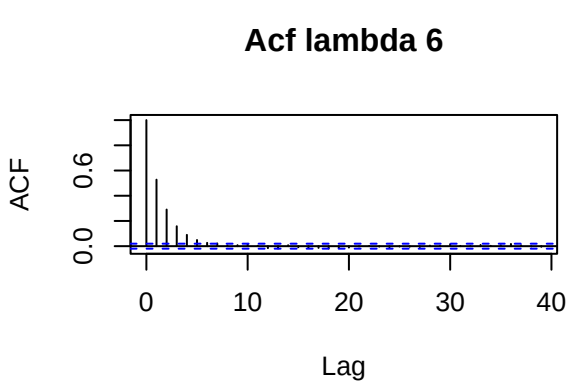
These are also the autocorrelation functions, to see if there is serial correlation.As expected, all autocorrelations decrease as the lag t increases, until they reach practically zero values. In the case of θ, they continue to have significant values for higher lags with respect to lambda's ones.

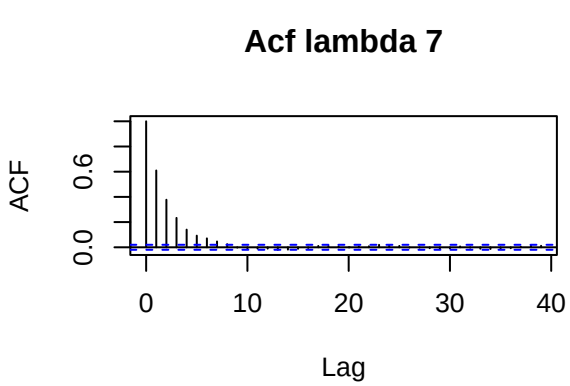


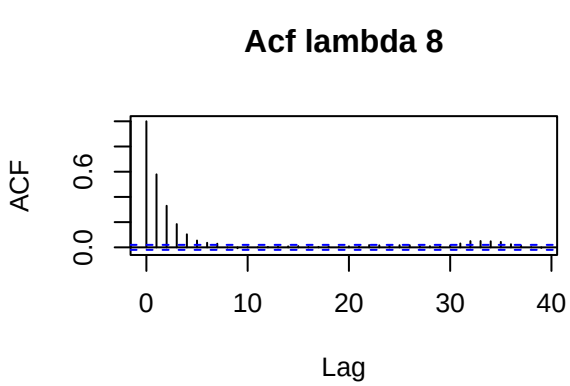


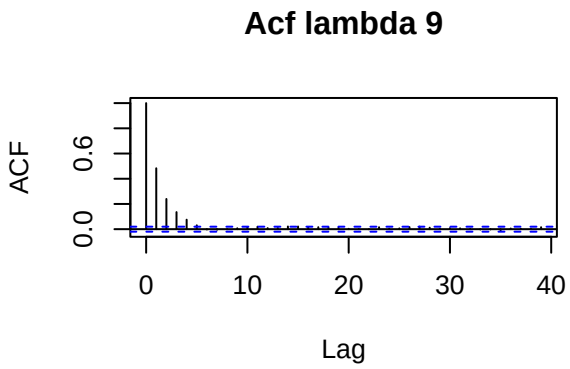


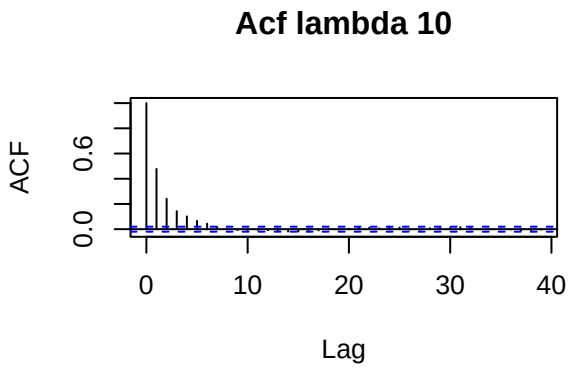


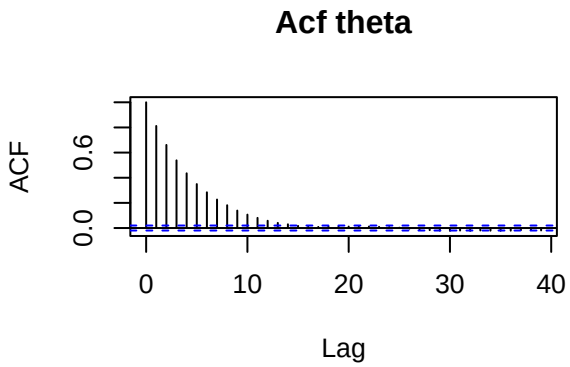







Acf lambda 9
ACF
0.0
0.6
0
10
20
30
40
Lag


Acf lambda 10
ACF
0.0
0.6
0
10
20
30
40
Lag


Acf theta
ACF
0.0
0.6
0
10
20
30
40
Lag

Model Comparison

It's not so easy to decide how to compare model the two model. As previously said, the ELO rating is a deterministic measure of the players' strength, and it is define such that the difference in rating between two players determines an estimate for the expected score between them:

$$E_{ij} = \frac{1}{2}P(i \; ties \; j) = \frac{1}{1 + 10^{\frac{ELO_j - ELO_i}{400}}}$$

we can verify which model has an the expected score closer to it.

We compute the ELO expected score for the two model:

```
e=c(2363,2345,2569,2666,2557,2363,2510,2564,2313,2380)

Expected_elo<-matrix(NA,10,10)
colnames(Expected_elo)=team
rownames(Expected_elo)=team
for(i in 1:10){
  for(j in 1:10){
    if(i!=j){
      temp=(e[j]-e[i])/400
      Expected_elo[i,j]=1/(1+10^temp)
    }
  }
}
```

```
## [1] "Expected score Elo (for both models,just a subset):"

##              Bologna Fiorentina     Inter  Juventus     Lazio
## Bologna           NA  0.5258809 0.2340053 0.1487792 0.2466139
## Fiorentina 0.4741191         NA 0.2159463 0.1361285 0.2278713

##                Milan    Napoli      Roma Sampdoria    Torino
## Bologna    0.5000000 0.3002306 0.2392039 0.5714631 0.4755545
## Fiorentina 0.4741191 0.2789218 0.2208594 0.5459219 0.4498006
```

For the model without ties the expected score are compute as follow:

$$\frac{\lambda_i}{\lambda_i + \lambda_j}$$

```
## [1] "Expected value without ties (just a subset):"
```

```
##                Bologna Fiorentina     Inter   Juventus     Lazio
## Bologna             NA  0.3194444 0.1982759 0.07419355 0.2473118
## Fiorentina 0.6805556           NA 0.3450704 0.14583333 0.4117647

##                  Milan    Napoli      Roma Sampdoria    Torino
## Bologna    0.2053571 0.1179487 0.1483871 0.3239437 0.3833333
## Fiorentina 0.3550725 0.2217195 0.2707182 0.5051546 0.5697674
```

And the corresponding mean error, computed as the mean of the absolute value of the difference between E_{ij} and expected value of the model without ties, is:

```
## [1] 0.1209448
```

While in the case with ties expected value of the model is:

$$p_1 + \frac{1}{2}p_2 = \frac{\lambda_i}{\lambda_i + \theta\lambda_j} + \frac{1}{2}\frac{(\theta^2 - 1)\lambda_i\lambda_j}{(\lambda_i + \theta\lambda_j)(\theta\lambda_i + \lambda_j)}$$

```
## [1] "Expected value with ties (just a subset):"
```

```
##                Bologna Fiorentina     Inter  Juventus     Lazio
## Bologna             NA  0.3964449 0.2887925 0.1458053 0.3484726
## Fiorentina 0.6035551           NA 0.3833127 0.2097692 0.4492417

##                  Milan    Napoli      Roma Sampdoria    Torino
## Bologna    0.3112526 0.2117891 0.2469565 0.4157343 0.4240271
## Fiorentina 0.4085096 0.2929536 0.3350295 0.5198924 0.5283569
```

And the corresponding mean error is:

```
## [1] 0.1072781
```

So the Bradly Terry model with ties could be a better fit for the data since the mean error is smaller! A possible explanation for this solution is that in the second model there are more observation (812 vs 587).

Bibliography

[1] Caron, Francois, and Arnaud Doucet. "Efficient Bayesian inference for generalized Bradley–Terry models." Journal of Computational and Graphical Statistics 21.1 (2012): 174-196.

[2] https://youhoo0521.github.io/kaggle-march-madness-men-2019/eda/pairwise_matchups.html

[3] https://en.wikipedia.org/wiki/Elo_rating_system

[4] https://www.calcioweb.eu/2019/11/dataset-calcio-performance-futuri-campioni/10375221/

[5] http://fussballelo.de/en/rating.php?league=Serie%20A

[6] Bradley, R. A. and M. E. Terry (1952). Rank analysis of incomplete block designs I: The method of paired comparisons. Biometrika 39, 324–45.

[7] Critchlow, D. E. and M. A. Fligner (1991). Paired com parison, triple comparison, and ranking experiments as generalized linear models, and their implementation in GLIM.

[9] "Bradley-Terry models in R",David Firth (University of Warwick, United Kingdom). January 2005 Journal of Statistical Software 12(i01)