# General linear mixture models

## Problem 1

### a)

We are considering a mixed model of the form

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \gamma_{0,i} + \gamma_{1,i} x_{ij} + \epsilon_{ij}$$

with

$$\boldsymbol{\gamma}_i = (\gamma_{0,i}, \gamma_{1,i}) \sim N\left(\mathbf{0}, \begin{bmatrix} \tau_0^2 & \tau_{0,1} \\ \tau_{0,1} & \tau_1^2 \end{bmatrix}\right)$$

and $\epsilon_{ij} \sim N(0, \sigma^2)$ for $i = 1, ..., m$ and $j = 1, ..., n_i$.

The idea of the model, is to consider responses that are correlated, for instance if they are observed from different clusters, such that the observations within a cluster actually are correlated to oneanother. In our model, $i = 1, ..., m$ would denote the amount of groups or clusters, while $n_i$ would be the number of observations within one group. The assumption of independent observations in a normal linear regression would here be violated, and the estimate of the unsecurity of the model would be incorrectly calculated by a simple linear model, hence we are adding some extra random terms, $\boldsymbol{\gamma}_i$, to correct for these correlations. $\beta_0 + \gamma_{0,i}$ would be a random intercept term for cluster $i$, while $\beta_1 + \gamma_{1,i}$ would similarly be a random slope coefficient for this group.

This can be written more compactly in the following fashion for some group $i$.

$$\boldsymbol{y_i} = \boldsymbol{X_i}\boldsymbol{\beta} + \boldsymbol{U_i}\boldsymbol{\gamma_i} + \boldsymbol{\epsilon_i} \qquad i = 1, ..., m$$

where

- $\boldsymbol{y_i}$: $n_i \times 1$ is the vector of responses
- $\boldsymbol{X_i}$: $n_i \times p$ is the vector of $p$ covariates (intercept included) for every observation
- $\boldsymbol{U_i}$: $n_i \times (q+1)$ is the vector of the desired random terms for every observation
- $\boldsymbol{\beta}$: $p \times 1$ is the vector of regression coefficients for the $p$ covariates
- $\boldsymbol{\gamma_i}$: $(q+1) \times 1$ is the vector of random contributions to the desired random terms
- $\boldsymbol{\epsilon_i}$: $n_i \times 1$ is the vector or random noise terms

If we combine all the different $m$ groups, this can be written as a general linear mixed model given by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{U}\boldsymbol{\gamma} + \boldsymbol{\epsilon}$$

where now

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \\ \vdots \\ \mathbf{Y}_m \end{pmatrix}, \mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_m \end{pmatrix}, \mathbf{U} = \begin{pmatrix} \mathbf{U}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2 & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{U}_m \end{pmatrix}, \boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \end{pmatrix}, \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_m \end{pmatrix}$$

## b)

To calclulate the estimates for the parameters, the function `mylmm()` was created. It takes in the response vector $y$, the covariate $x$ and the groupings $g$, as well as the boolean variable `REML`. It returns the estimates of the parameters of the linear mixed model, $(\beta_0, \beta_1, \tau_0^2, \tau_{01}, \tau_1^2, \sigma^2)$. The variable `REML` decides whether the estimates obtained are ordinary maximum likelihood estimates (ML) or restricted maximum likelihood (REML).

```r
library(MASS)
library(Matrix)
library(lme4)


getG <- function(theta) {
  Q <- matrix(c(theta[1], theta[3], theta[3], theta[2]), nrow = 2)
  m <- length(unique(g))
  G <- bdiag(lapply(rep(1:(m)), FUN=function(x) Q))
  return(G)}

getR <- function(theta) {
  return(diag(rep(theta[4], length(y))))}

calcV <- function(theta) {
  return(as.matrix(getR(theta) + U %*% getG(theta) %*% t(U)))}

calcBeta <- function(theta){
  vinv <- solve(calcV(theta))
  X_transpose <- t(X)
  return(solve(X_transpose %*% vinv %*% X) %*% X_transpose %*% vinv %*% y)}

calcNll <- function(theta, REML) {
  V <- calcV(theta)
  Vinv <- solve(V)
  betas <- calcBeta(V)
  nllp <- 1/2 * (determinant(V, logarithm=TRUE)$modulus + (t(y - X %*% betas) %*% Vinv %*% (y - X %*% be
  if (REML) {return(nllp + 1/2 * (determinant(t(X) %*% Vinv %*% X)$modulus))}
  else {return(nllp)}}


mylmm <- function(x,y,g,REML){
  opt <- optim(par = c(1.5, 2, 0.5, 2), fn= calcNll, gr=NULL, REML = REML,
               method = "L-BFGS-B", lower=c(0, 0, -Inf, 0), upper=c(Inf, Inf, Inf, Inf))

  theta <- opt$par
  betas <- calcBeta(theta)
  return(round(matrix(c(betas,theta),ncol=1),2))
}


data <- sleepstudy
y <- data$Reaction
x <- data$Days
g <- data$Subject
```

```
m <- length(unique(g))
U <- bdiag(lapply(seq(1:m), function(i) cbind(1, x[g == unique(g)[i]])))
X <- cbind(1, x)

est_ML <- mylmm(x,y,g,F)
est_REML <- mylmm(x,y,g,T)
results <- data.frame(cbind(est_ML,est_REML))
row.names(results) =  c("B0", "B1", "t_0^2","t_01","t_1^2","sigma^2")
names(results) = c("est_ML","est_REML")
results
```

```
##         est_ML est_REML
## B0      251.41   251.41
## B1       10.47    10.47
## t_0^2   565.53   612.09
## t_01     32.68    35.07
## t_1^2    11.05     9.60
## sigma^2 654.93   654.94
```

Here, the results are displayed. We see there is a difference between the ML and the REML estimates. This will be discussed later.

### c)

In this section we wish to compare our results to the ones of the function `lmer` in the package `lme`.

```
data <- sleepstudy
lmer_ML <- lmer(data$Reaction ~ 1 + data$Days + (1+ data$Days|data$Subject), REML=FALSE)
lmer_REML <- lmer(data$Reaction ~ 1 + data$Days + (1+ data$Days|data$Subject), REML=TRUE)
lmers <- as.data.frame(cbind(as.data.frame(VarCorr(lmer_ML))[,"vcov"],as.data.frame(VarCorr(lmer_REML))
names(lmers) <- names(results)
row.names(lmers) =  c("t_0^2","t_01","t_1^2","sigma^2")
lmers
```

```
##            est_ML   est_REML
## t_0^2   565.47697 611.897607
## t_01     32.68179  35.081069
## t_1^2    11.05512   9.613886
## sigma^2 654.94571 654.940796
```

We see that `lmer()` produce almost identical results to `mylmm()`.

### d)

For both functions, there are some slight differences between the ML-estimates and the REML-estimates. Without going into to much of the details, the problem with regular ML is that there is a (downwards) bias in the variance estimate. This appears because of the unknown estimator for the mean which is used in the expression for the variance estimator.

The REML avoids using the mean in the log-likelihood function, by integrating the mean out and computing a marginal log likelihood for the variance.

## Problem 2

### a)

We fit a Poisson general linear mixture model to the dataset. The underlying mathematical assumptions are the following

- The responses $Y_{ij}$ for some group $j$ are conditionally independent and Poisson distributed with some conditional mean $E(Y_{ij}|\gamma_{ij}) = \lambda_{ij}$, given random effects $\gamma_{ij}$ and covariates $\boldsymbol{x}_{ij}$,$\boldsymbol{u}_{ij}$.
- The conditional distribution $f(y_{ij}|\gamma_{ij})$ belongs to an exponential family.
- The mean $\lambda_{ij}$ is connected to the linear predictor $\eta_{ij} = \boldsymbol{x}_{ij}^T\boldsymbol{\beta} + \boldsymbol{u}_{ij}\gamma_i$ through the log link function $ln(\lambda_{ij}) = \eta_{ij}$

In this case the random effect parameters are associated to each team and for the two covariates "attack" and "defence". That is because, dependent on how good the different teams are at either attacking or defending, the number of goals to either team in a match is dependent of the teams playing, hence the response variables "goal" are correlated for the games featuring the same team. These random effects are there to adjust for this correlation. $\boldsymbol{u}_{ij}$ is the design matrix with the desired covariates to randomize. Here we are considering a random intercept model, with two random effects. $\boldsymbol{x}_{ij}$ is the design matrix with the considered covariates, here we only have $x_1$ : "Home". $Y_{ij}$ in this model would be the goals scored from the attacking team in match $i$, and this team being of group $j$. The linear predictor for one observation $i$ of group/team $j$ would look like

$$\eta_{ij} = \boldsymbol{x^t}\boldsymbol{\beta} + \boldsymbol{u}_{ij}^t\boldsymbol{\gamma_{ij}}$$

where

$$\mathbf{x} = \begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}, \boldsymbol{u_{ij}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \boldsymbol{\gamma_i} = \begin{pmatrix} \gamma_{1j} \\ \gamma_{2j} \end{pmatrix},$$

where $\gamma_{1j}$ is the random effect for team $j$ in the group "attacking" and $\gamma_{2j}$ is the random effect for team $j$ in the group "defence", representing the defensive team. $\boldsymbol{\gamma_{ij}}$ are normally distributed with mean 0 and some covariance matrix. In this model we assume that there is no covariance between the $\gamma_{1j}$ and $\gamma_{2j}$ so the non-diagonal elements are zero. So,

$$\boldsymbol{\gamma_{ij}} \sim \left(\mathbf{0}, \begin{bmatrix} \tau_1^2 & 0 \\ 0 & \tau_2^2 \end{bmatrix}\right)$$

A poisson distribution for modeling the amount of goals scored in a football match might be reasonable, as we then get get all positive discrete values including zero. Also, these events are assumed to be independent. Distributions like the bernoulli, the normal one or the gamma distribution yield binomial or continous values respecetively, and would not be suitable. Also, the time between events in a Poisson process are exponentially distributed and independent. This would then model the time between each goal in the match, and the exponential distribution is memoryless, which means that the history of the process itself, or the current time since a goal was scored is uninteresting. The probability of a new goal is the same regardless of the history.

One could argue that this is a decent assumption for a football game. However, if you actually study statistics about football games, one could also argue that this is not really true, since there are certain times within the game where there is statistically more like to be a goal, than other times. But we would agree that overall it is a decent assumption, but not perfect, which obviously is true for most simplified models one makes.

```r
library(glmmTMB)
```

```
## Warning in checkMatrixPackageVersion(): Package version inconsistency detected.
## TMB was built with Matrix version 1.2.18
## Current Matrix version is 1.2.17
## Please re-install 'TMB' from source using install.packages('TMB', type = 'source') or ask CRAN for a
```

```
fotballdata <- read.csv("https://www.math.ntnu.no/emner/TMA4315/2020h/eliteserie.csv", colClasses = c("
fotballdata_real <- fotballdata[c(1:384),]
mod <- glmmTMB(goals ~ home + (1|attack) + (1|defence), poisson, data=fotballdata_real, REML=TRUE)
summary(mod)
```

```
##  Family: poisson  ( log )
## Formula:          goals ~ home + (1 | attack) + (1 | defence)
## Data: fotballdata_real
## 
##      AIC      BIC   logLik deviance df.resid 
##   1147.2   1163.1   -569.6   1139.2      382 
## 
## Random effects:
## 
## Conditional model:
##  Groups  Name        Variance Std.Dev.
##  attack  (Intercept) 0.007478 0.08647 
##  defence (Intercept) 0.016383 0.12800 
## Number of obs: 384, groups:  attack, 16; defence, 16
## 
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept)  0.12421    0.07809   1.591    0.112    
## homeyes      0.40716    0.08745   4.656 3.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
randeffs <- ranef(mod)
```

### b)

From investgating the fitted model, we observe that the estimted coefficients $\beta_0$ and $\beta_1$ was estimated to 0.12421 and 0.40716 respectively. If we evaluate the expected value of goals with the link function, using these estimates and ommit the random effects for now, we observe that if a team is playing away from home, the expected values of goals can be estimated to

$$\lambda_i = exp(\beta_0) = exp(0.12421)$$

similarly, for the home team in this case the expected value of goals would be

$$\lambda_i = exp(\beta_0 + \beta_1) = exp(0.12421 + 0.40716)$$

```
beta0 = mod$fit$parfull[1]
beta1 = mod$fit$parfull[2]
exp_av_away = exp(beta0)
exp_av_home = exp(beta0 + beta1)
cat("Estimated goals away team without random effects:", exp_av_away)
```

```
## Estimated goals away team without random effects: 1.132258
```

```
cat("Estimated goals home team without random effects:", exp_av_home)
```

```
## Estimated goals home team without random effects: 1.701265
```

We see that the expected goals would be approximately 1.13 and 1.7 for the away and the home team respectively, which are numbers that intuitively makes sense for a football match. It is not that often that too many goals are scored, and we would expect the home team to have an advantage, meaning they would be expected to score more goals.

Studying the random effects, we see that the averages for both attack and defence are approximately zero, which we also would expect for these random terms, as they are distributed with zero mean.

```r
av_attack <- mean(randeffs$cond$attack$`(Intercept)`)
av_def <- mean(randeffs$cond$defence$`(Intercept)`)
cat("Average attack random effects:", av_attack)
```

```
## Average attack random effects: -1.870249e-18
```

```r
cat("Average defence random effects:", av_def)
```

```
## Average defence random effects: -1.008308e-17
```

Hence, studying the expected outcome of a match between an "average" attacking team and an "average" defensive team, would be equal to set these random effects equal to zero, and calculate the expected outcome in terms of goals. This is indeed what we did previously in order to evaluate whether the estimates seemed reasonable.

In addition, we know that the variance of the poisson variable also equals $lambda_i$, so we conclude that the expected goals and variance for an average hometeam is $E(Y_i|\gamma_i = 0) = Var(Y_i|\gamma_i = 0) \approx 1.7$, and for an average away team we get the following $E(Y_i|\gamma_i = 0) = Var(Y_i|\gamma_i = 0) \approx 1.13$.

## c)

Here, we use the total law of expectation and variance to calculate the estimated goals scored and the corresponding variance for a randomly chosen match in the league.

```r
att_values <- randeffs$cond$attack$`(Intercept)`
def_values <- randeffs$cond$defence$`(Intercept)`
N <- length(att_values)
lambdas_home <- c()
lambdas_away <- c()
for (i in 1:N){
  for (j in 1:N){
    if (j != i){
      lambdas_home = append(lambdas_home,exp(beta0 + beta1 + att_values[i] + def_values[j]))
      lambdas_away = append(lambdas_away,exp(beta0 + att_values[i] + def_values[j]))
    }
  }
}

random_exp_home <- mean(lambdas_home)
random_exp_away <- mean(lambdas_away)
random_var_home <- mean(lambdas_home) + var(lambdas_home)
random_var_away <- mean(lambdas_away) + var(lambdas_away)

cat("Expected value of goals for random chosen home team:", random_exp_home)
```

```
## Expected value of goals for random chosen home team: 1.707288
```

```r
cat("Variance for random chosen home team:", random_var_away)
```

```
## Variance for random chosen home team: 1.145341
```

```r
cat("Expected value of goals for random chosen away team:", random_exp_home)
```

```
## Expected value of goals for random chosen away team: 1.707288
```

```r
cat("Variance for random chosen away team:", random_var_away)
```

```
## Variance for random chosen away team: 1.145341
```

By using the total law of expectation and variance, the values are pretty similar to the ones for avarage teams, which is to be expected, as we also here capture the average to some extend.

For finding the proportions of variance being explained by team strength and randomness respectively, we consider the decomposition of the law of total variance, namely

$$Var(Y_{ij}) = E[Var(Y_{ij}|\gamma_{ij})] + Var(E[Y_{ij}|\gamma_{ij}])$$

and then we assumed that the first term, which explains the variance that we can expect, given the strengths of the teams, actually explains this portion given by team strengths. Similary, the second term, which explains the variance of the expected value given some teams, was assumed to explain uncaptured noise in the football game. Hence we found these proportions relative to the total variance. We did this for home teams and away teams seperately.

```r
prop_strength_home <- mean(lambdas_home) / random_var_home
prop_random_home <- var(lambdas_home)/ random_var_home
prop_strength_away <-mean(lambdas_away)/random_var_away
prop_random_away <-var(lambdas_away)/random_var_away

cat("Proportion of variance being the strength of the team for home teams:", prop_strength_home)
```

```
## Proportion of variance being the strength of the team for home teams: 0.9881421
```

```r
cat("Proportion of variance being randomness for home teams:", prop_random_home)
```

```
## Proportion of variance being randomness for home teams: 0.01185793
```

```r
cat("Proportion of variance being the strength of the team for away teams:", prop_strength_away)
```

```
## Proportion of variance being the strength of the team for away teams: 0.9920767
```

```r
cat("Proportion of variance being randomness for away teams:", prop_random_away)
```

```
## Proportion of variance being randomness for away teams: 0.007923339
```

Most of the variance seems to come from the strength of the teams, which sound reasonable, in that this is obviously the most influencial factor for a game. Also, what we found interesting, was that it seems this proportion is even higher when you play away from home. We thought this might make sense, because for the hometeam there might be other uncaptured factors influencing the team, like the fans, the known circumstances and so on, while for away teams, you have to rely more on your actual team strengths.

## d)

For testing the significance of the random effects, we are using the restricted likelihood ratio test, as this is the preferred one over the normal likelihood ratio test. We want to test whether the variance of the random effects are equal to zero, so we construct two tests

$$H_0 : \tau_1^2 = 0 \quad \text{vs} \quad H_1 : \tau_1^2 \neq 0$$

$$H_0 : \tau_2^2 = 0 \quad \text{vs} \quad H_1 : \tau_2^2 \neq 0$$

The test statistics read

$$2\Big(l_r(\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\vartheta}}) - l_r(\tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\vartheta}})\Big)$$

where $l_r(\hat{\boldsymbol{\beta}},\hat{\boldsymbol{\vartheta}})$ is the restricted likelihood of the full model, while $l_r(\tilde{\boldsymbol{\beta}},\tilde{\boldsymbol{\vartheta}})$ is the restricted likelihood under $H_0$. Since $\tau_1^2$ and $\tau_2^2$ are variances, we are actually here checking whether the values lie on the boundary of the set of possible parameter values. Therefor it can be shown that the test statistic has a mixed distribution of $\chi_0^2$ and $\chi_1^2$, with the two being equally weighted. The p-values and threshold value are calculated accordingly, and considering the upper tails of the distribution.

```
mod2 <- glmmTMB(goals ~ home + (1|defence), poisson, data=fotballdata_real, REML=TRUE)
mod3 <- glmmTMB(goals ~ home + (1|attack), poisson, data=fotballdata_real, REML=TRUE)

##Calculating test-statistics##
loglikefull = -mod$fit$objective
loglikeatt = -mod2$fit$objective
loglikedef = -mod3$fit$objective

lrtestattacks = 2*(loglikefull-loglikeatt)

lrtestdefence = 2*(loglikefull-loglikedef)

##Calculating P-values##
p_attacks = 0.5*pchisq(q=lrtestattacks,df = 0,lower.tail = FALSE) + 0.5*pchisq(q=lrtestattacks,df = 1,l
p_defence =  0.5*pchisq(q=lrtestdefence,df = 0,lower.tail = FALSE) + 0.5*pchisq(q=lrtestdefence,df = 1,l
threshold = 0.5*qchisq(0.05,1,lower.tail = FALSE) + 0.5*qchisq(0.05,0,lower.tail = FALSE)

cat("P-value for random effect of attack strength :", p_attacks)
```

```
## P-value for random effect of attack strength : 0.2587558
```

```
cat("P-value for random effect of defence strength :", p_defence)
```

```
## P-value for random effect of defence strength : 0.09843786
```

```
cat("Threshold value for rejecting H_0 for random effects:", threshold)
```

```
## Threshold value for rejecting H_0 for random effects: 1.920729
```

We conclude, based on the p-values being $\approx 0.26$ and $\approx 0.1$, that we dont reject any of the null hypotheses with a 0.05 significance level for the random effects, and we can hence not conclude that they are significant in the model. The threshold value for the test-statistic was calculated to be $\approx 1.92$, while our values were $\approx 0.42$ and $\approx 1.67$ for attack and defence respectively.

For testing the hypotheses that the home game advantage is non-existent, that is

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_1 : \beta_1 \neq 0$$

we also use the likelihood ratio test in similar fashion as previously. Here however, the test statistic is chi squared distributed with 1 degree of freedom, as we removed one parameter from the model. The calculations are done here

```
### Calculate p-values beta ###
mod4 <- glmmTMB(goals ~ (1|attack) + (1|defence), poisson, data=fotballdata_real, REML=TRUE)
loglikehome = -mod4$fit$objective
lrtesthome = 2*(loglikefull-loglikehome)
thresholdHome = qchisq(0.05,1,lower.tail = FALSE)
p_home = pchisq(lrtesthome,df=1,lower.tail = FALSE)

cat("P-value for fixed effect of home game :", p_home)
```

```
## P-value for fixed effect of home game : 1.24331e-05
```

```
cat("Threshold value for rejecting H_0 for fixed effects:", thresholdHome)
```

```
## Threshold value for rejecting H_0 for fixed effects: 3.841459
```

The p-value is well lower than 0.05, and hence we reject the null hypotheses and conclude that the effect of home advantage seems to be significant. The threshold value for rejecting the null hypotheses was calculated to be ≈ 3.84 for the test statistic, which in our case takes the value ≈ 19.09.

## e)

In the following piece of code, we have calculated how the current league table would look like, based on the already known results in the league. If a team wins, they get 3 points. For a draw, both teams are awarded with 1 point. We have also presented the goal difference in the table, as well as the random effects for defence and attack for each team, which was modelled already in b).

```
teams = unique(fotballdata_real$attack)
tabell <- data.frame("lag" = teams,"poeng" = rep(0,16),"gd" = rep(0,16))
row.names(tabell) <- tabell$lag
gamma_attack = data.frame(randeffs$cond$attack)
gamma_defence = data.frame(randeffs$cond$defence)
tabell = merge(tabell,gamma_attack,by = 0)
row.names(tabell) = tabell$lag
tabell = merge(tabell,gamma_defence,by=0)
```

```
## Warning in merge.data.frame(tabell, gamma_defence, by = 0): column name
## 'Row.names' is duplicated in the result
```

```
names(tabell)[6] <- "attacks"
names(tabell)[7] <- "defence"
tabell = tabell[,!(names(tabell) == "Row.names")]

for (i in seq(1,lengths(fotballdata_real[1]),2)){
  goalhome = fotballdata_real$goals[i]
  goalaway = fotballdata_real$goals[i+1]
  gdhome = goalhome - goalaway
  gdaway = goalaway - goalhome
  if (goalhome > goalaway){
    tabell[tabell$lag == fotballdata_real$attack[i],2] = tabell[tabell$lag == fotballdata_real$attack[i
  }
  else if (goalhome == goalaway){
    tabell[tabell$lag == fotballdata_real$attack[i],2] = tabell[tabell$lag == fotballdata_real$attack[i
    tabell[tabell$lag == fotballdata_real$attack[i+1],2] = tabell[tabell$lag == fotballdata_real$attack
  }
  else {
    tabell[tabell$lag == fotballdata_real$attack[i+1],2] = tabell[tabell$lag == fotballdata_real$attack
  }
  tabell[tabell$lag == fotballdata_real$attack[i],3] = tabell[tabell$lag == fotballdata_real$attack[i],
  tabell[tabell$lag == fotballdata_real$attack[i+1],3] = tabell[tabell$lag == fotballdata_real$attack[i
}
tabell = tabell[order( tabell[,2], tabell[,3],decreasing = TRUE ),]
row.names(tabell)=NULL
tabell
```

```
##                     lag poeng  gd      attacks      defence
## 1           Rosenborg    52  23  0.050622609 -0.152631173
## 2               Brann    48  13  0.012026209 -0.123934761
## 3               Molde    43  18  0.078390643 -0.036630979
## 4           Haugesund    41   8  0.011223106 -0.061931278
## 5          Ranheim_TF    38  -2  0.023375599  0.062209734
## 6          Vaalerenga    36  -2  0.007147494  0.031030079
## 7                 Odd    34   6  0.003654179 -0.052013600
## 8             Tromsoe    33   2  0.005756700 -0.009852817
## 9         Sarpsborg08    32   5  0.026946364  0.006574064
## 10       Kristiansund    31  -3 -0.011367328  0.008112432
## 11         BodoeGlimt    27  -2 -0.036781062 -0.042616090
## 12      Stroemsgodset    26   0  0.024556017  0.040486666
## 13        Lillestroem    25 -11 -0.049915996  0.030699257
## 14            Stabaek    23 -14 -0.026801293  0.085376126
## 15              Start    23 -18 -0.060500163  0.081958112
## 16 Sandefjord_Fotball    15 -23 -0.058333079  0.133164228
```

## f,g)

Now we want to simulate the remaining matches. For each match we simulate $N = 100,000$ times the number of goals for each team based on the expected value. In other words, we simulated the score of each match $N$ times. The most frequent outcome (among home-win, draw, away-win) is set to be the most probable. Based on this we award the winning team 3 points, the losing team 0, or 1 point to each team in case of draw. The expected goal difference is computed by taking the mean of goal differences in the $N$ simulated matches. Based on these calculations, Rosenborg is expected to win the league by one point, as the next table shows.

```r
newdata = fotballdata[c(385:480),]

predicted_scores <- predict(mod, newdata=newdata, type="response")

for (i in seq(1,length(predicted_scores)[1],2)){
  goalhome = rpois(100000,predicted_scores[i])
  goalaway = rpois(100000,predicted_scores[i+1])

  outcomes=table(sign(goalhome-goalaway))
  winner=strtoi(names(outcomes[outcomes==max(outcomes)]))

  gdhome = mean(goalhome - goalaway)
  gdaway = mean(goalaway - goalhome)
  if (winner == 1){
    tabell[tabell$lag == newdata$attack[i],2] = tabell[tabell$lag == newdata$attack[i],2] + 3
  }
  else if (winner == 0){
    tabell[tabell$lag == newdata$attack[i],2] = tabell[tabell$lag == newdata$attack[i],2] + 1
    tabell[tabell$lag == newdata$attack[i+1],2] = tabell[tabell$lag == newdata$attack[i+1],2] + 1
  }
  else {
    tabell[tabell$lag == newdata$attack[i+1],2] = tabell[tabell$lag == newdata$attack[i+1],2] + 3
  }
  tabell[tabell$lag == newdata$attack[i],3] = tabell[tabell$lag == newdata$attack[i],3] + gdhome
  tabell[tabell$lag == newdata$attack[i+1],3] = tabell[tabell$lag == newdata$attack[i+1],3] + gdaway
}
tabell = tabell[order( tabell[,2], tabell[,3],decreasing = TRUE ),]
```

```r
row.names(tabell)=NULL
tabell
```

```
##                    lag poeng        gd      attacks      defence
## 1            Rosenborg    61  24.96975  0.050622609 -0.152631173
## 2                Brann    60  15.12391  0.012026209 -0.123934761
## 3            Haugesund    50   8.59810  0.011223106 -0.061931278
## 4                Molde    49  18.13979  0.078390643 -0.036630979
## 5           Ranheim_TF    47  -2.42057  0.023375599  0.062209734
## 6           Vaalerenga    45  -1.97761  0.007147494  0.031030079
## 7                  Odd    43   6.21128  0.003654179 -0.052013600
## 8              Tromsoe    42   2.20578  0.005756700 -0.009852817
## 9         Kristiansund    40  -2.82223 -0.011367328  0.008112432
## 10         Sarpsborg08    38   4.22775  0.026946364  0.006574064
## 11          BodoeGlimt    36  -1.95393 -0.036781062 -0.042616090
## 12       Stroemsgodset    35   0.17465  0.024556017  0.040486666
## 13             Stabaek    35 -13.95517 -0.026801293  0.085376126
## 14         Lillestroem    34 -12.08740 -0.049915996  0.030699257
## 15               Start    32 -19.56300 -0.060500163  0.081958112
## 16 Sandefjord_Fotball    24 -24.87110 -0.058333079  0.133164228
```

We have included the $\gamma$-values in the table above, to investigate whether there is an clear relationship between placement and $\gamma$-value. It seems to be a trend that the$\gamma_{1j}$-value, corresponding to attack, decreases as the the lower on the table a team ends up, while the $\gamma_{2j}$-value, corresponding to defence, increases. Meaning that a team lower on the table will often be weaker in attack, as well as weaker in defence, as is expected.