Project 3 - TMA4315

martin.o.berild@ntnu.no
10014

yaolin.ge@ntnu.no
10026

November 22, 2020

Mixed Model example

In this problem we want to implement a function `mylmm` that computes the maximum likelihood or if specified the restricted maximum likelihood estimates of the parameters $(\beta_0, \beta_1, \tau_0^2, \tau_1^2, \sigma^2)$ of the mixed model

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

where $\boldsymbol{\gamma} = (\gamma_{0i}, \gamma_{1j})$ is independent and identically (i.i.d.) Gaussian distributed with zero mean and covariance matrix:

$$\begin{bmatrix} \tau_0^2 & \tau_{01} \\ \tau_{01} & \tau_1^2 \end{bmatrix},$$

and ϵ_{ij} is i.i.d. $\mathcal{N}(0, \sigma^2)$.

It can be shown that the matrix formulation has two steps. First, the measurement model can be rewritten as follows:

$$y_{ij} = \mathbf{x}_{ij}^T \boldsymbol{\beta} + \mathbf{u}_{ij}^T \boldsymbol{\gamma_i} + \epsilon_{ij}$$

In this case,

$$\mathbf{x}_{ij} = \begin{bmatrix} 1 \\ x_{ij} \end{bmatrix}; \quad \mathbf{u}_{ij} = \begin{bmatrix} 1 \\ x_{ij} \end{bmatrix}; \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}; \quad \boldsymbol{\gamma}_i = \begin{bmatrix} \gamma_{0,i} \\ \gamma_{1,i} \end{bmatrix}$$

By collecting all individual- and cluster-specific responses y_{ij}, design vectors $\mathbf{x}_{ij}, \mathbf{u}_{ij}$ and errors $\epsilon_{ij}, j = 1, \cdots, n_i$ into vectors, then it becomes:

$$\mathbf{y}_i = \begin{pmatrix} y_{i1} \\ \vdots \\ y_{ij} \\ \vdots \\ y_{in_i} \end{pmatrix}; \quad \mathbf{X}_i = \begin{pmatrix} \boldsymbol{x}_{i1}^T \\ \vdots \\ \boldsymbol{x}_{ij}^T \\ \vdots \\ \boldsymbol{x}_{in_i}^T \end{pmatrix}; \quad \mathbf{U}_i = \begin{pmatrix} \boldsymbol{u}_{i1}^T \\ \vdots \\ \boldsymbol{u}_{ij}^T \\ \vdots \\ \boldsymbol{u}_{in_i}^T \end{pmatrix}; \quad \boldsymbol{\epsilon}_i = \begin{pmatrix} \epsilon_{i1} \\ \vdots \\ \epsilon_{ij} \\ \vdots \\ \epsilon_{in_i} \end{pmatrix}$$

Thus, the measurement model in matrix notation is

$$\boldsymbol{y}_i = \boldsymbol{X}_i \boldsymbol{\beta} + \boldsymbol{U_i} \boldsymbol{\gamma_i} + \boldsymbol{\epsilon_i}, \quad i = 1, \cdots, m$$

Given that

$$\boldsymbol{\gamma}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}); \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I_{n_i}})$$

Thus, squeeze the above LMM model by defining the design matrices as follows:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_i \\ \vdots \\ \boldsymbol{y}_m \end{pmatrix}; \quad \boldsymbol{\epsilon} = \begin{pmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_i \\ \vdots \\ \boldsymbol{\epsilon}_m \end{pmatrix}; \quad \boldsymbol{\gamma} = \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_i \\ \vdots \\ \gamma_m \end{pmatrix};$$

Therefore, the LMM can be rewritten as

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

Our implementation of the `mylmm()` function is given below:

```
library(Matrix)
mylmm <- function(y, x, group, REML = FALSE){
  X = cbind(1, x) # design matrix for fixed effects
  logdet <- function(A) as.numeric(determinant(A)$mod)
  # generate the design matrix
  U = list()
  for (i in levels(group)) {
    U[[i]] = cbind(1,x[group==i])
  }
  U = bdiag(U) # design matrix for random effects

  V_func <- function(theta, U){ # function used to find V matrix
    # tau0^2 = theta[1]; tau1^2 = theta[2]; tau01 = theta[3]; sigma^2 = theta[4]
    DIM = dim(U)
    n_total = DIM[1]
    n_groups = DIM[2] / 2
    R <- diag(theta[4], nrow = n_total)
    # R is a n_total * n_total matrix for the independent variance
    Q <- matrix(c(theta[1], theta[3], theta[3], theta[2]), nrow = 2, byrow = TRUE)
    G <- list()
    for (i in c(1:n_groups)){
      G[[i]] <- Q
    }
    G <- bdiag(G)
    V <- U %*% G %*% t(U) + R # variance matrix
    return(V)
  }
  Beta_func <- function(V, X, y){ # function used to find beta
    solve(t(X) %*% solve(V) %*% X) %*% t(X) %*% solve(V) %*% y # Estimated parameters
  }
  loglik <- function(theta){ # loglik function
    V <- V_func(theta, U)
    Beta <- Beta_func(V, X, y)
    if(!REML){
      return(as.vector(- 1 / 2 * (logdet(V) +
                              t(y - X %*% Beta) %*% solve(V) %*%
                              (y - X %*% Beta)))) # profile log likelihood
    } else{
      return(as.vector( - 1 / 2 * (logdet(V) + t(y - X %*% Beta) %*%
```

```r
                              solve(V) %*% (y - X %*% Beta)) -
                           (1 / 2) * (logdet(t(X) %*% solve(V) %*% X))))
    }
  }
  theta_hat <- optim(par=c(0.1, 0.1, 0.1, 0.1), loglik,
                     method="L-BFGS-B",lower=c(.001,.001,-1,.001),
                     upper=c(Inf,Inf,Inf,Inf),control=list(fnscale=-1))$par
  beta_hat <- Beta_func(V_func(theta_hat, U), X, y)
  return(c(beta_hat = as.vector(beta_hat), theta_hat = theta_hat))
}
```

To test this implementation we compare will compare the parameter estimates it to the estimates obtained with the `lmer()` function from the `lme4` library. First, we will look at the maximum likelihood (ML) estimates by specifying `REML = FALSE`.

```r
library(lme4)
attach(sleepstudy)
mod <- lmer(Reaction ~ 1 + Days + (1 + Days|Subject),
            REML = FALSE, data = sleepstudy)
mod_sum = summary(mod)
mylmm_coef <- mylmm(sleepstudy$Reaction, sleepstudy$Days,
                    sleepstudy$Subject, REML = FALSE)
cbind(matrix(c(mod_sum$coefficients[,1],diag(mod_sum$varcor$Subject),
         mod_sum$varcor$Subject[2],mod_sum$sigma^2),nrow=6,
       dimnames = list(c("beta_hat1","beta_hat2","tau_0^2",
                         "tau_1^2","tau_01","sigma^2"),"lmer")),mylmm_coef)
```

The collected results from the `lmer()` function and our own implementation `mylmm()` is very similar. Next, we check the Restricted Maximum Likelihood (REML) estimates by specifying `REML = FALSE` in both function calls:

```r
mod_reml <- lmer(Reaction ~ 1 + Days + (1 + Days|Subject),
                 REML = TRUE, data = sleepstudy)
mod_sum = summary(mod_reml)

mylmm_coef <- mylmm(sleepstudy$Reaction, sleepstudy$Days,
                    sleepstudy$Subject, REML = TRUE)
cbind(matrix(c(mod_sum$coefficients[,1],diag(mod_sum$varcor$Subject),
         mod_sum$varcor$Subject[2],mod_sum$sigma^2),nrow=6,
       dimnames = list(c("beta_hat1","beta_hat2","tau_0^2",
                         "tau_1^2","tau_01","sigma^2"),"lmer")),mylmm_coef)
detach(sleepstudy)
```

Again, the estimates are quite similar.

Based on the estimates, we observe that the ML estimates tends to yield a smaller variance and standard error which is because of the induced bias from the $\boldsymbol{\beta}$s in the log-likelihood. The restricted log-likelihood is constructed by integrating out the betas from the log-likelihood. This is motivated by a empirical Bayesian perspective, where a constant prior distribution is assumed for the $\boldsymbol{\beta}$s.

Generalized linear mixed model example

We will now employ a generlized linear mixed model (GLMM) to describe the 2018 results of the Norwegian elite football league. Firstly, we load the data and observe the contents:

```r
long <- read.csv("https://www.math.ntnu.no/emner/TMA4315/2020h/eliteserie.csv", colClasses = c("factor"
head(long)
```

```
##                attack            defence home goals
## 1               Molde Sandefjord_Fotball  yes     5
## 2 Sandefjord_Fotball              Molde   no     0
## 3      Stroemsgodset            Stabaek  yes     2
## 4            Stabaek      Stroemsgodset   no     2
## 5                Odd          Haugesund  yes     1
## 6          Haugesund                Odd   no     2
```

Here, each match is represented by pairs of rows; the first contains the number of goals scored by one team and if they were at their home stadium, and thus, the second row contains the number of goals the opponent scored and similarly if they were at home. In this model there are four variables; `attack`and `defence` which is a factor of 16 levels (16 teams), `home` a factor of two levels (yes/no), and lastly `goals` which is a numeric value of goals scored by the factor in `attack`. There is a total of 480 rows which means that there is a total of 240 matches.

To define GLMMs, we combine the notation of generalized linear models (GLMs) with the linear mixed models (LMMs) describe above. Consider the conditional density of a response y_i given a linear predictor η_i belonging to the exponential familiy. Moreover, the linear predictor is linked to the mean μ_i through a response function $h(\eta_i)$ and inversly the link function $\eta_i = g(\mu_i)$. Now, the linear predictor in GLMMs is extended to include random effects $\boldsymbol{\gamma}$ as

$$\eta_{ij} = \boldsymbol{x}_{ij}^T\boldsymbol{\beta} + \boldsymbol{u}_{ij}^T\boldsymbol{\gamma}_i, \quad i = 1, \ldots, m, \;\; j = 1, \ldots, n_i,$$

where n_i is the number of measurements (subjects) in cluster (group) i, and m is the number of clusters. We want to model the number of goals scored with the fixed effect of `home`, and random effect of subject `attack` and `defence` using a poisson likelihood. In other words, the grouping specifies the `attack` team i with each match $j = 1, \ldots, n_i$ specifying which `defence` team it is playing against. Thus in the above equation, $\boldsymbol{\beta}$ is the fixed effects of the covariate `home` (x_{ij}), and γ_{0i} is the random effects of `attack` (group) i and γ_{1j} the random effect of `defence` team. Note, that will only deal with the canonical link. Furthermore, we assume that the random effects are Gaussian distributed with mean zero with a positive definite covariance matrix $\boldsymbol{Q} = diag(\tau_0^2, \tau_1^2)$, i.e. $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_0, \boldsymbol{\gamma}_1) \overset{\text{i.i.d}}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$, where we assume that these the random effects are independent and identically distributed. The resulting random rate of the poisson likelihood is $\lambda_{ij} = \exp(\beta_0 + x_{ij}\beta_1 + \gamma_{0i} + \gamma_{1j})$.

The poisson likelihood is commonly used in GLMMs for count responses, and in this setting the number of goals y_{ij} are conditionally idependent for i and j, but not marginally independent.

Next, we fit the model using the **R** package `glmTMB` as

```r
library(glmmTMB)
mod <- glmmTMB(goals ~ home + (1|attack) + (1|defence), family=poisson, data=long, REML=TRUE)
```

and observe the `summary()` output of the fit:

```r
summary(mod)
```

```
##  Family: poisson  ( log )
## Formula:          goals ~ home + (1 | attack) + (1 | defence)
## Data: long
##
##      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
```

Here, we there is a positive effect on the number of goals by playing at the home stadium which is reasonable since it is commonly known to be a favorable effect to winning (score more goals). The intercept has a p-value of 0.112 and looks to be insignificant. If a intercept is not to be included, given that `home` is a factor, the expected number of goals given `home=no` would be zero which is not going to be the case. In addition, if we try to fit a model without a intercept, the `glmmTMB()` function will split the factors of `home` in two fixed effects that are either on or off (on hot encoding).

Next, we are interested in observing the values of the random effects:

```r
randeff=ranef(mod)
randeff
```

```
## $attack
##                    (Intercept)
## BodoeGlimt        -0.036781062
## Brann              0.012026209
## Haugesund          0.011223106
## Kristiansund      -0.011367328
## Lillestroem       -0.049915996
## Molde              0.078390643
## Odd                0.003654179
## Ranheim_TF         0.023375599
## Rosenborg          0.050622609
## Sandefjord_Fotball -0.058333079
## Sarpsborg08        0.026946364
## Stabaek           -0.026801293
## Start             -0.060500163
## Stroemsgodset      0.024556017
## Tromsoe            0.005756700
## Vaalerenga         0.007147494
##
## $defence
##                    (Intercept)
## BodoeGlimt        -0.042616090
## Brann             -0.123934761
## Haugesund         -0.061931278
## Kristiansund       0.008112432
## Lillestroem        0.030699257
## Molde             -0.036630979
## Odd               -0.052013600
## Ranheim_TF         0.062209734
## Rosenborg         -0.152631173
## Sandefjord_Fotball  0.133164228
## Sarpsborg08        0.006574064
```

```
## Stabaek              0.085376126
## Start                0.081958112
## Stroemsgodset        0.040486666
## Tromsoe             -0.009852817
## Vaalerenga           0.031030079
```

Here, we observe that the random effects of `attack` have an average value of $5.6920614 \times 10^{-19}$, and the `defence` has $-1.517883 \times 10^{-18}$ which is to the expected value of zero defined in the model. In the summary output we also observe the variance of these effects. A high positive effect if `attack` means that the team is scoring a lot of goals, and a high negative effect of `defence` means they are good at defending and does not get scored on that much.

We are interested in calculating the exepectation and variance of the number of goals scored given a average attacking team that plays at is home field against another average defence team. Consider a `attack` team k who is playing at home, $x_{kl} = 1$ against `defence` l and the random effects $\gamma_{0k}, \gamma_{1l} = \mathbf{0}$, and thus, the expected value and variance of goals is

$$\mathbb{E}[y_k | x_{kl} = 1, (\gamma_{0l}, \gamma_{1k}) = \mathbf{0}] = \mathrm{Var}[y_k | x_{kl} = 1, (\gamma_{0k}, \gamma_{1l}) = \mathbf{0}] = \exp\{\beta_0 + \beta_1\},$$

which results in 117.0618202. And oppositely, for `attack` team l not at home $x_{lk} = 0$ and `defence` team is k we have

$$\mathbb{E}[y_l | x_{lk} = 0, (\gamma_{0l}, \gamma_{1k}) = \mathbf{0}] = \mathrm{Var}[y_k | x_{lk} = 0, (\gamma_{0l}, \gamma_{1k}) = \mathbf{0}] = \exp\{\beta_0\},$$

which gives the value `{r exp(tmp3[1,1]}`.

Next, we are interested in finding the expected number of goals scored by a random `attack` team playing against a random `defence` team. Where one of which is playing at his home stadium. To do this we must look at the law of total probability and find the moment generating functions. For $\gamma_{0i}, \gamma_{1j} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\tau_0^2, \tau_1^2)$, we have the moment generating function:

$$M_{\gamma_{0i},\gamma_{1j}}(t) = \exp\left(\frac{1}{2}\tau_0^2 t^2 + \frac{1}{2}\tau_1^2 t^2\right).$$

Thus, the expected number of goals by the law of total probability is

$$\begin{aligned}
\mathbb{E}[y_{ij}] &= \mathbb{E}\left[\mathbb{E}[y_{ij} | \gamma_{0i}, \gamma_{1j}]\right] \\
&= \mathbb{E}\left[\exp(\beta_0 + x_{ij}\beta_1 + \gamma_{0i} + \gamma_{1j})\right] \\
&= \exp(\beta_0 + x_{ij}\beta_1) \cdot \mathbb{E}[\exp(\gamma_{0i} + \gamma_{1j})] \\
&= \exp(\beta_0 + x_{ij}\beta_1) \exp\left(\frac{1}{2}\tau_0^2 + \frac{1}{2}\tau_1^2\right).
\end{aligned}$$

Furthermore, the variance is

$$\begin{aligned}
\mathrm{Var}[y_{ij}] &= \mathbb{E}\left[\mathrm{Var}[y_{ij} | \gamma_{0i}, \gamma_{1j}]\right] \\
&= \mathbb{E}\left[\exp(\beta_0 + x_{ij}\beta_1 + \gamma_{0i} + \gamma_{1j})\right] + \mathrm{Var}\left[\exp(\beta_0 + x_{ij}\beta_1 + \gamma_{0i} + \gamma_{1j})\right] \\
&= \exp(\beta_0 + x_{ij}\beta_1) \exp\left(\frac{1}{2}\tau_0^2 + \frac{1}{2}\tau_1^2\right) + \exp(2\beta_0 + 2x_{ij}\beta_1) \cdot \exp\left(\tau_0^2 + \tau_1^2\right) \cdot \left(\exp(\tau_0^2 + \tau_1^2) - 1\right)
\end{aligned}.$$

The expected number of goals given `attack` is playing at home $x_{ij} = 1$, $\mathbb{E}[y_{ij} | x_{ij} = 1]$, is:

and oppositley if `attack` is playing away $x_{ij} = 0$, $\mathbb{E}[y_{ij} | x_{ij} = 0]$, we have:

The variances of are for $x_{ij} = 1$:

and $x_{ij} = 0$:

Note, that we have used the estimated standard deviation $\tau_0 = 0.0074776$ and $\tau_1 = 0.0163833$.

To test the significants we will perform a likelihood ratio test as:

$$H_0: \ \tau_0^2 = 0 \ \text{ vs. } \ H_1: \tau_0^2 > 0,$$

or

$$H_0: \ \tau_1^2 = 0 \ \text{ vs. } \ H_1: \tau_1^2 > 0.$$

We will deal with the LRT asymptotically being a mixture $0.5\chi_0^2 : 0.5\chi_1^2$ mixture. The p-value of the LRT test for the random effect of `attack` is:

```
mod_attack <- glmmTMB(goals ~ home + (1|attack), poisson, data=long, REML=TRUE)
lrt_att = as.numeric(2 *(logLik(mod)-logLik(mod_attack)))
p_att = 0.5*pchisq(lrt_att, df=1, lower.tail=F)
p_att
```

```
## [1] 0.09843786
```

The p-value of `defence` is:

```
mod_defence <- glmmTMB(goals ~ home + (1|defence), poisson, data=long, REML=TRUE)
lrt_def =as.numeric(2*(logLik(mod)-logLik(mod_defence)))
p_def = 0.5*pchisq(lrt_def, df=1, lower.tail=F)
p_def
```

```
## [1] 0.2587558
```

And lasly `home` is:

```
mod_home = glmmTMB(goals ~ (1|attack) + (1|defence) , poisson, data=long, REML=TRUE)
lrt_home =  as.numeric(2*(logLik(mod)-logLik(mod_home)))
p_def = pchisq(lrt_home, df=1, lower.tail=F)
p_def
```

```
## [1] 1.24331e-05
```

From these test we observe that the fixed effect of `home` is the only significant effect. However, this test can be conservatitve since $\tau = 0$ is at the boundary of the parameter space of γ. An alternative can be parametetric bootstrap to obtain p-values. Lastly, assuming that we are employ a significance level of $\alpha = 0.05$ the critical value $z = \text{inv}{-}\chi^2(0.95) = 3.84$.

We want to create a function that rank the teams of the 2018 season based on the number of points they obtained. If a team wins they get 3 points, if they draw they get one point, and if they lose they get 0. Furthermore, if there is a draw in number of points the team is ranked by the number of goals scored. In the norweigan league there is more rules, but we only consider these. The function is implemented below and it is restrictive in that it takes advantage of the row order of matches in `long`.

```
rankteams <- function(df){
  teamnames = as.vector(unique(df$attack))
  tabell = data.frame(rank = numeric(length(teamnames)),
                      points = numeric(length(teamnames)),
                      goals = numeric(length(teamnames)),
                      played = numeric(length(teamnames)))
  rownames(tabell) = teamnames
  for (i  in seq(2,nrow(df),by=2)){
    tabell[as.vector(df$attack[i]),]$goals = tabell[as.vector(df$attack[i]),]$goals + df$goals[i]
    tabell[as.vector(df$attack[i-1]),]$goals = tabell[as.vector(df$attack[i-1]),]$goals + df$goals[i-1]
    tabell[as.vector(df$attack[i]),]$played = tabell[as.vector(df$attack[i]),]$played +1
    tabell[as.vector(df$attack[i-1]),]$played = tabell[as.vector(df$attack[i-1]),]$played +1
```

```r
    if (df$goals[i] > df$goals[i-1]){
      tabell[as.vector(df$attack[i]),]$points = tabell[as.vector(df$attack[i]),]$points + 3
    }else if (df$goals[i] < df$goals[i-1]){
      tabell[as.vector(df$attack[i-1]),]$points = tabell[as.vector(df$attack[i-1]),]$points + 3
    }else{
      tabell[as.vector(df$attack[i-1]),]$points = tabell[as.vector(df$attack[i-1]),]$points + 1
      tabell[as.vector(df$attack[i]),]$points = tabell[as.vector(df$attack[i]),]$points + 1
    }
  }
  tabell = tabell[order( tabell[,2], tabell[,3],decreasing = c(TRUE,TRUE)),]
  tabell$rank = seq(length(teamnames))
  tabell
}
```

To test the function we omit the NA values and find the rankings:

```r
rankteams(na.omit(long))
```

```
##                     rank points goals played
## Rosenborg              1     52    43     24
## Brann                  2     48    36     24
## Molde                  3     43    48     24
## Haugesund              4     41    36     24
## Ranheim_TF             5     38    38     24
## Vaalerenga             6     36    35     24
## Odd                    7     34    35     24
## Tromsoe                8     33    35     24
## Sarpsborg08            9     32    39     24
## Kristiansund          10     31    32     24
## BodoeGlimt            11     27    28     24
## Stroemsgodset         12     26    38     24
## Lillestroem           13     25    26     24
## Stabaek               14     23    29     24
## Start                 15     23    24     24
## Sandefjord_Fotball    16     15    24     24
```

Using the function created above, we want to simulate 1000 realization of rankings in the league. This is achieved by predicting the expected number of goals in all 240 matches or 480 rows during a season, and then run the ranking function for each realization. The code below performs this simulation and the top rows of these realizations of rankings is printed out.

```r
lambdas = predict(mod,newdata = long,type = "response")
rankings = matrix(data=NA, nrow=1000,ncol = 16)
colnames(rankings) = unique(long$attack)
tmp_long = long
for (i in seq(1000)){
  tmp = rpois(480,lambdas)
  tmp_long$goals = tmp
  tmp_rank = rankteams(tmp_long)
  rankings[i,rownames(tmp_rank)] = tmp_rank$rank
}
head(rankings)
```

```
##      Molde Sandefjord_Fotball Stroemsgodset Stabaek Odd Haugesund BodoeGlimt
## [1,]     3                 11             7      14  16        10          4
## [2,]    16                 10            12      15   5         7          6
```

```
## [3,]      8                 15             4        5  12          14           13
## [4,]     14                 16            10       15  11           2           13
## [5,]      3                 15            16       10   2           7            6
## [6,]      7                  9            12        2   8           6           10
##       Lillestroem Start Tromsoe Sarpsborg08 Rosenborg Kristiansund Vaalerenga
## [1,]           15     8       9           5         1            2          6
## [2,]           14    13       2           8         9            4         11
## [3,]           10     3      16           6         7           11          1
## [4,]            6    12       3           8         5            1          7
## [5,]            8    11      13          12         5            9         14
## [6,]           11    14       4          15        13           16          5
##       Ranheim_TF Brann
## [1,]          13    12
## [2,]           3     1
## [3,]           2     9
## [4,]           4     9
## [5,]           4     1
## [6,]           1     3
```

To summarize the realizations, we look at the probability of ranking which is obtained by the Monte Carlo estimates of the frequency of occurance of a rank for a specific team. The table below shows these probabilities.

```r
probs = matrix(NA, nrow = 16, ncol = 16)
rownames(probs) = colnames(rankings)
for (i in seq(16)){
  for (j in seq(16)){
    probs[i,j] = sum(rankings[,i]==j)/1000
  }
}
probs
```

```
##                       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
## Molde                0.117 0.108 0.094 0.085 0.082 0.091 0.071 0.054 0.053 0.059
## Sandefjord_Fotball 0.008 0.017 0.021 0.027 0.037 0.024 0.044 0.056 0.052 0.053
## Stroemsgodset      0.052 0.050 0.055 0.050 0.063 0.071 0.069 0.069 0.063 0.072
## Stabaek            0.017 0.032 0.028 0.031 0.052 0.054 0.065 0.063 0.067 0.074
## Odd                0.077 0.069 0.092 0.074 0.077 0.070 0.066 0.069 0.060 0.067
## Haugesund          0.075 0.089 0.083 0.076 0.067 0.085 0.078 0.056 0.066 0.060
## BodoeGlimt         0.062 0.058 0.063 0.056 0.063 0.064 0.058 0.064 0.065 0.069
## Lillestroem        0.019 0.033 0.034 0.048 0.058 0.051 0.065 0.052 0.064 0.066
## Start              0.013 0.023 0.018 0.037 0.042 0.041 0.039 0.059 0.059 0.073
## Tromsoe            0.054 0.058 0.059 0.085 0.066 0.078 0.068 0.071 0.061 0.052
## Sarpsborg08        0.076 0.058 0.061 0.079 0.068 0.068 0.058 0.067 0.074 0.049
## Rosenborg          0.182 0.154 0.124 0.091 0.076 0.059 0.063 0.049 0.044 0.039
## Kristiansund       0.045 0.049 0.050 0.067 0.053 0.057 0.054 0.069 0.069 0.068
## Vaalerenga         0.039 0.047 0.072 0.061 0.059 0.065 0.072 0.072 0.065 0.068
## Ranheim_TF         0.041 0.045 0.058 0.048 0.063 0.056 0.065 0.067 0.077 0.071
## Brann              0.123 0.110 0.088 0.085 0.074 0.066 0.065 0.063 0.061 0.060
##                    [,11] [,12] [,13] [,14] [,15] [,16]
## Molde              0.041 0.031 0.035 0.034 0.030 0.015
## Sandefjord_Fotball 0.069 0.082 0.086 0.106 0.128 0.190
## Stroemsgodset      0.062 0.069 0.063 0.072 0.070 0.050
## Stabaek            0.078 0.082 0.078 0.081 0.101 0.097
## Odd                0.052 0.066 0.047 0.041 0.040 0.033
## Haugesund          0.047 0.049 0.049 0.048 0.038 0.034
```

```
## BodoeGlimt         0.075 0.067 0.059 0.067 0.054 0.056
## Lillestroem        0.068 0.082 0.093 0.084 0.089 0.094
## Start              0.066 0.094 0.084 0.105 0.105 0.142
## Tromsoe            0.073 0.060 0.071 0.053 0.055 0.036
## Sarpsborg08        0.062 0.062 0.071 0.063 0.039 0.045
## Rosenborg          0.032 0.019 0.023 0.020 0.015 0.010
## Kristiansund       0.085 0.058 0.068 0.067 0.075 0.066
## Vaalerenga         0.073 0.057 0.070 0.066 0.055 0.059
## Ranheim_TF         0.068 0.076 0.061 0.063 0.081 0.060
## Brann              0.049 0.046 0.042 0.030 0.025 0.013
```

Using these probabilities, we can calculate the expected rank as

$$\mathbb{E}[r_i] = \int r \cdot \mathrm{p}_i(r) \cdot \mathrm{d}r = \sum_{k=1}^{16} k \cdot \mathrm{p}_i(k),$$

where index i specifies the team. The code below calculates the respective expected ranks.

```
exp_rank = matrix(NA,nrow = 16, ncol = 1)
rownames(exp_rank) = rownames(probs)
for (i in seq(16)){
  exp_rank[i] = sum(probs[i,]*seq(16))
}
exp_rank
```

```
##                      [,1]
## Molde               6.351
## Sandefjord_Fotball 11.601
## Stroemsgodset       8.767
## Stabaek            10.232
## Odd                 7.493
## Haugesund           7.367
## BodoeGlimt          8.523
## Lillestroem        10.038
## Start              10.986
## Tromsoe             8.187
## Sarpsborg08         8.073
## Rosenborg           5.123
## Kristiansund        9.033
## Vaalerenga          8.713
## Ranheim_TF          9.052
## Brann               6.461
```

In addition, it could be interesting to compare the random effects `attack` and `defence` with the expected rank.

```
comp_rank = cbind(randeff$cond$attack$`(Intercept)`,randeff$cond$defence$`(Intercept)`,exp_rank[row.name
rownames(comp_rank) = rownames(randeff$cond$attack)
colnames(comp_rank) = c("Attack", "Defence","E[rank]")
comp_rank
```

```
##                          Attack      Defence E[rank]
## BodoeGlimt         -0.036781062 -0.042616090   8.523
## Brann               0.012026209 -0.123934761   6.461
## Haugesund           0.011223106 -0.061931278   7.367
## Kristiansund       -0.011367328  0.008112432   9.033
## Lillestroem        -0.049915996  0.030699257  10.038
```

```
## Molde              0.078390643 -0.036630979   6.351
## Odd                0.003654179 -0.052013600   7.493
## Ranheim_TF         0.023375599  0.062209734   9.052
## Rosenborg          0.050622609 -0.152631173   5.123
## Sandefjord_Fotball -0.058333079  0.133164228  11.601
## Sarpsborg08        0.026946364  0.006574064   8.073
## Stabaek           -0.026801293  0.085376126  10.232
## Start             -0.060500163  0.081958112  10.986
## Stroemsgodset      0.024556017  0.040486666   8.767
## Tromsoe            0.005756700 -0.009852817   8.187
## Vaalerenga         0.007147494  0.031030079   8.713
```

As mentioned earlier a team is better than the rest if it has a larger `attack` and smaller `defence`, e.g. `Rosenborg` has this property and has the highest expected rank. We could also reduce the Monte Carlo error by simulating more realization of ranks.