TMA4315: Compulsory exercise 1 (title)

Group 0: Name1, Name2 (subtitle)

13.10.2023

```
rm(list = ls())

showsol <- FALSE

library(formatR)
library(knitr)
library(ggplot2)
library(tidyverse)
library(glue)  # fstring-like
library(doSNOW)  # Parallel computing
library(foreach)
```

Part 1

Bold

italic

To get a pdf file, make comments of the lines with the "html_document" information, and make the lines with the "pdf_document" information regular, and vice versa.

a)

The log-likelihood function for a binary regression model:

$$\ell(\beta) = \sum_{i=1}^{n} y_i ln(\pi_i) + (1 - y_i) ln(1 - \pi_i)$$

How we arrived at this expression: 1. Start with the link function, aka logarithm of the odds, $ln(\frac{\pi}{1-\pi}) = \sum_{i=0}^{n} \beta_i x_i$, where $x_0 = 1$. This can be written as $\pi = \frac{1}{1+e^{-\sum_{i=0}^{n} \beta_i x_i}}$ 2. Then we formulate the likelihood: $L(\beta) = \prod_{i=1}^{n} P(x_i|\beta) = \prod_{i=1}^{n} \left(\frac{1}{1+\exp(-(\beta x_i))}\right)^{y_i} \left(1 - \frac{1}{1+\exp(-(\beta x_i))}\right)^{1-y_i}$ (here β and x_i are vectors, and y_i is the response-variable). 3. We get the log-likelihood by taking the logarithm: $\ell(\beta) = \sum_{i=1}^{n} \left[y_i \log\left(\frac{1}{1+\exp(-(\beta x_i))}\right) + (1 - y_i) \log\left(1 - \frac{1}{1+\exp(-(\beta x_i))}\right)\right]$ 4. We put $\pi = \frac{1}{1+\exp(-(\beta x_i))}$ and get $\ell(\beta) = \sum_{i=1}^{n} \left[y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i)\right]$.

b)

```r
# importing data
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2018h/mountains"
mount <- read.table(file = filepath, header = TRUE, col.names = c("height",
    "prominence", "fail", "success"))
```

```r
# fitting model
logreg.mod = glm(cbind(success, fail) ~ height + prominence, data = mount,
    family = "binomial")
summary(logreg.mod)
```

```
##
## Call:
## glm(formula = cbind(success, fail) ~ height + prominence, family = "binomial",
##     data = mount)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.369e+01  1.064e+00  12.861  < 2e-16 ***
## height      -1.635e-03  1.420e-04 -11.521  < 2e-16 ***
## prominence  -1.740e-04  4.554e-05  -3.821 0.000133 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 715.29  on 112  degrees of freedom
## Residual deviance: 414.68  on 110  degrees of freedom
## AIC: 686.03
##
## Number of Fisher Scoring iterations: 4
```

```r
# perform likelihood ratio test
logreg.null_mod = glm(cbind(success, fail) ~ 1, data = mount, family = "binomial")
anova(logreg.null_mod, logreg.mod, "LRT")
```

```
## Analysis of Deviance Table
##
## Model 1: cbind(success, fail) ~ 1
## Model 2: cbind(success, fail) ~ height + prominence
##   Resid. Df Resid. Dev Df Deviance
## 1       112     715.29
## 2       110     414.68  2   300.61
```

```r
# creating 95% confint for beta
cat(" \n Confidence interval: \n")
```

```
##
##  Confidence interval:
```

```r
confint(logreg.mod)
```

```
##                     2.5 %        97.5 %
## (Intercept) 11.6156540313  1.578900e+01
## height      -0.0019157664 -1.359060e-03
## prominence  -0.0002633821 -8.480319e-05
```

```r
cat(" \n exp(CI): \n")
```

```
##
##  exp(CI):
```

```r
exp(confint(logreg.mod))
```

```
##                    2.5 %       97.5 %
## (Intercept) 1.108191e+05 7.195726e+06
## height      9.980861e-01 9.986419e-01
## prominence  9.997367e-01 9.999152e-01
```

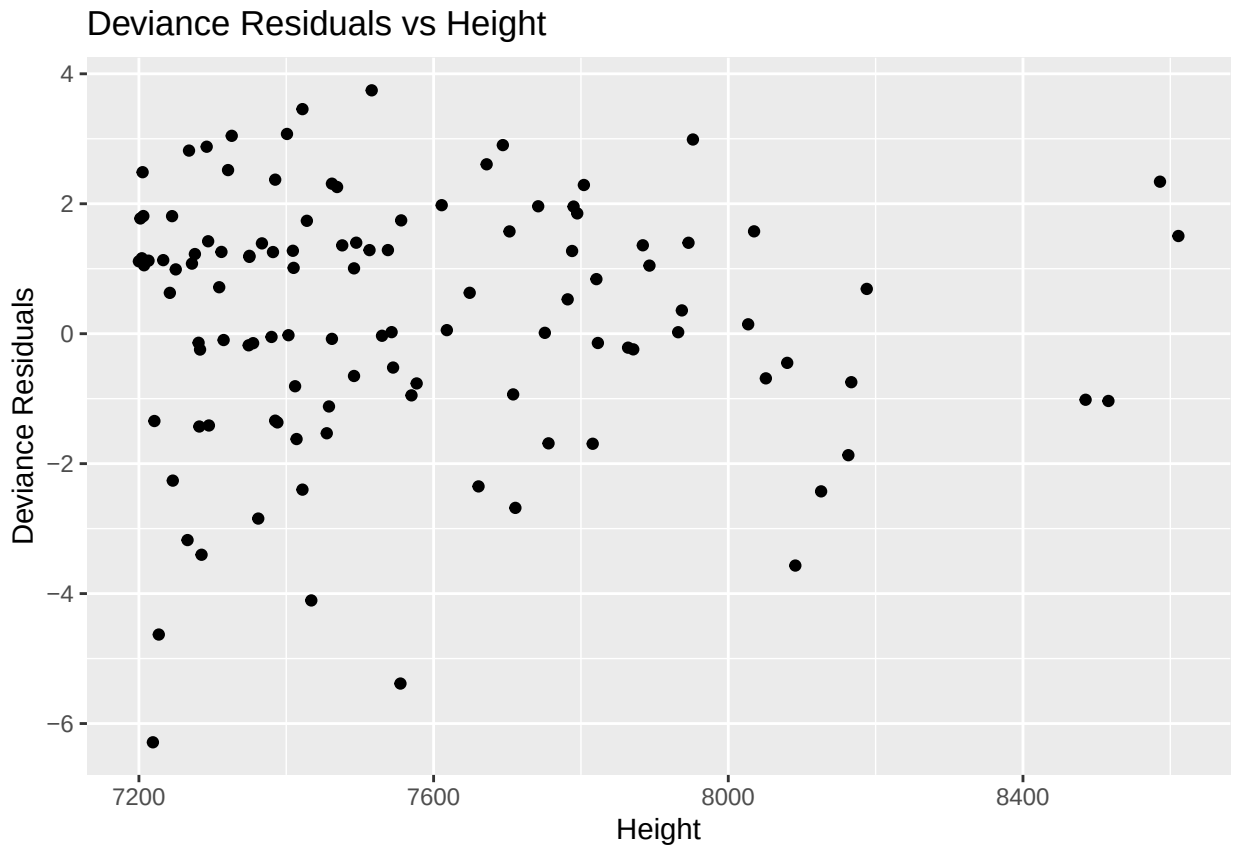
- We can see from the estimates that the log(odds) decrease when height or prominence increase, meaning the chance of success decreases.
- The p-values obtained by performed wald-test suggest that the covariates are very significant. We also performed LRT, where the decrease in residual deviance from 715.29 in Model 1 to 414.68 in Model 2 indicates that Model 2 fits the data significantly better.
- CI for height: (-0.0019157664 -1.359060e-03), which also clearly indicates a negative log-odds relation between height and success.
- In we take $(exp(\beta_L), exp(\beta_H))$ we get the odds instead of the log-odds, which is given above under exp(CI). Here, since odds-confidence interval for height and prominence displays values <1, it indicates that for unit increase in height and prominence the odds for success diminishes.

c)

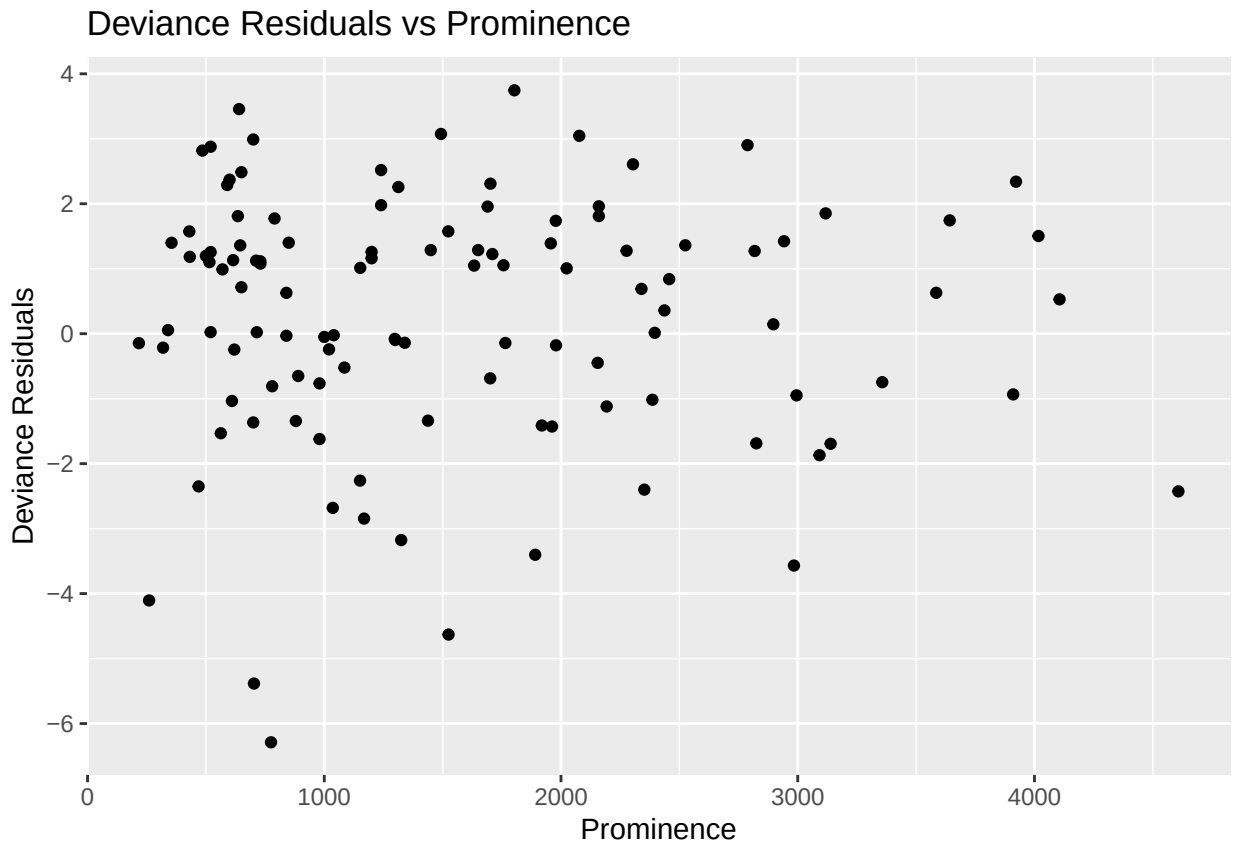
```r
dev_res <- residuals(logreg.mod, type = "deviance")

# Create a data frame to use with ggplot2
plot_data <- data.frame(Height = logreg.mod$data$height, Prominence = logreg.mod$data$prominence,
    DevRes = dev_res)

# Plotting deviance residuals against height
ggplot(plot_data, aes(x = Height, y = DevRes)) + geom_point() + labs(x = "Height",
    y = "Deviance Residuals", title = "Deviance Residuals vs Height")
```




```
# Plotting deviance residuals against prominence
ggplot(plot_data, aes(x = Prominence, y = DevRes)) + geom_point() + labs(x = "Prominence",
    y = "Deviance Residuals", title = "Deviance Residuals vs Prominence")
```



Part 2

```
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2023h/eliteserien2023.csv"
eliteserie <- read.csv(file = filepath)
eliteserie.played <- eliteserie %>%
    drop_na()
eliteserie.unplayed <- eliteserie[eliteserie %>%
    apply(1, function(row) any(is.na(row))), ]

NGames <- table(c(eliteserie.played$home[!is.na(eliteserie.played$yh)],
    eliteserie.played$away[!is.na(eliteserie.played$yh)]))
RangeofGames <- range(NGames)
```

a)

The null hypothesis is that the goals scored by a team is independent of whether they are home or away, i.e. that the goals scored by the home- and away team are independently distributed. In this case we have two team categories which can score M goals, where $M = \max_{ij} O_{ij}$, $\mathbf{O} \in \mathbb{R}^{2\times M}$ is the matrix representing the contingency table. The matrix of expected frequencies, whose entries are given by

$$E_{ij} = \frac{\sum_p \mathbf{O}_{ip} + \sum_k \mathbf{O}_{kj}}{\sum_{p,k} \mathbf{O}_{pk}}$$

can be written in matrix form as

$$\mathbf{E} = \frac{\mathbf{O}\mathbf{1}_M \mathbf{1}_2^T \mathbf{O}}{\mathbf{1}_2^T \mathbf{O} \mathbf{1}_M} \tag{1}$$

where $\mathbf{1}_k \in \mathbb{R}^k$ is the k-dimensional vector of ones. Using the definition of $\mathbf{O}$ and Equation @ref{eq:expfreq}, the χ^2-test can be written

$$\chi^2 = \sum_{i,j} \frac{\mathbf{O}_{ij} - \mathbf{E}_{ij}}{\mathbf{E}_{ij}} \tag{2}$$

This value is distributed by χ^2_{M-1} under the null hypothesis.

```
O <- rbind(table(eliteserie.played$yh), table(eliteserie.played$ya)) %>%
    as.matrix
O[2, (max(eliteserie.played$ya) + 2):(max(eliteserie.played$yh) + 1)] <- 0

M <- max(eliteserie.played$yh) + 1
E <- c(O %*% rep(1, M)) %o% c(t(O) %*% rep(1, 2))/c(rep(1, 2) %*% O %*%
    rep(1, M))
chi_sq <- sum((O - E)^2/E)
p.chi_sq <- pchisq(chi_sq, M - 1, lower.tail = FALSE)
p.chi_sq
```

```
## [1] 0.01663929
```

The p-value of the χ^2 test is significant at 0.017, and one can therefore conclude that the goals scored by the home- and away team are not independent. The assumption of independence is therefore not always reasonable.

b)

```
calculate_scores <- function(eliteserie) {
#   Given a dataframe from eliteserie, calculates the scores based on all matches
#   played so far.
#
#   Args:
#     elitesere (DataFrame): Eliteserie dataframe with columns (home, away, yh, ya)
#   Returns:
#     (Float) Score
  df.scores <- inner_join(
    eliteserie %>% # Score from home- and away matches
      group_by(home) %>%
      summarize(score=3*sum(yh > ya) + 1*sum(yh == ya)),
    eliteserie %>%
      group_by(away) %>%
      summarize(score=3*sum(yh < ya) + 1*sum(yh == ya)),
    by=c("home" = "away")
  ) %>%
    mutate(score=score.x + score.y) %>%
    select(-c("score.x", "score.y")) %>%
    rename(team=home)

  return(df.scores)
}
```

```r
scores.played <- calculate_scores(eliteserie.played) %>%
    mutate(position = rank(-score, ties.method = "min"))

scores.played %>%
    arrange(position)
```

```
## # A tibble: 16 x 3
##    team                score position
##    <chr>               <dbl>    <int>
##  1 Viking                 51        1
##  2 Bodø/Glimt             49        2
##  3 Tromsø                 48        3
##  4 Brann                  42        4
##  5 Molde                  41        5
##  6 Lillestrøm             36        6
##  7 Sarpsborg 08           34        7
##  8 Odd                    30        8
##  9 Rosenborg              29        9
## 10 Strømsgodset           27       10
## 11 HamKam                 24       11
## 12 Haugesund              21       12
## 13 Sandefjord Fotball     21       12
## 14 Vålerenga              21       12
## 15 Stabæk                 17       15
## 16 Aalesund               12       16
```

The only teams sharing scores are Haugesund, Sandefjord Fotball, and Vålerenga, each with 21 points.

```r
num_goals <- {inner_join(
  eliteserie.played %>% # Score from home- and away matches
    group_by(home) %>%
    summarize(goals=sum(yh)),
  eliteserie.played %>%
    group_by(away) %>%
    summarize(goals=sum(ya)),
  by=c("home" = "away")
)} %>%
  mutate(goals=goals.x + goals.y) %>%
  select(-c(goals.x, goals.y))

num_goals[c(5, 10, 16), ]
```

```
## # A tibble: 3 x 2
##   home               goals
##   <chr>              <int>
## 1 Haugesund             20
## 2 Sandefjord Fotball    34
## 3 Vålerenga             30
```

Thus, 12th place goes to Sandefjord Fotball, 13th place goes to Vålerenga, and 14th place goes to Haugesund.

c)

We want a model on the form

$$\ln \mathbb{E}(Y) = \beta_0 + \beta_{\text{home}} x_{\text{home}} + \mathbf{X}_{\text{teams}} \boldsymbol{\beta}_{\text{teams}}$$

where $\mathbf{X}_{\text{teams}}$ is the sub matrix of the design matrix including the one-hot encoding for all teams, and $\boldsymbol{\beta}_{\text{teams}}$ are the strength coefficients for each team.

The first step to fit the model is to transform the data into a suitable design matrix. In this case, the score should be the response, the team should be a category, and whether it is away or home should be another category. The first step to fit the model is to transform the data into a suitable design matrix. R will naturally handle the issue of linear dependence in the design matrix by making Aalesund a reference level.

```
eliteserie.played.transf <- eliteserie.played %>%
    gather(key = "is_home", value = "team", home, away) %>%
    gather(key = "y_type", value = "y", yh, ya) %>%
    filter((is_home == "home" & y_type == "yh") | (is_home == "away" &
        y_type == "ya")) %>%
    select(-c(y_type, X)) %>%
    mutate(is_home = ifelse(is_home == "home", TRUE, FALSE))
```

Then we can implement the Fisher scoring algorithm. For the Poisson family (log-link), the socre and expected Fisher information is given by

$$\boldsymbol{s}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - \exp\left(\boldsymbol{x}_i^T \beta \right) \right) \tag{3}$$

$$\mathbf{F}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T \exp\left(\boldsymbol{x}_i^T \beta \right) \tag{4}$$

and the Fisher-scoring algorithm is therefore

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \mathbf{F}(\boldsymbol{\beta}^{(t)})^{-1} \boldsymbol{s}(\boldsymbol{\beta}^{(t)}) \tag{5}$$

```
fisher_score_iteration <- function(beta, X, y) {
    n <- ncol(X)
    s <- c(t(y - exp(X %*% beta)) %*% X)
    Fisher <- matrix(rep(0, n^2), nrow = n, ncol = n)
    for (i in 1:nrow(X)) {
        # This is not ideal, but multidimensional arrays are
        # difficult to manage in R
        Fisher <- Fisher + c(exp(X[i, ] %*% beta)) * X[i, ] %o% X[i,
            ]
    }
    return(beta + solve(Fisher, s))
}
```

Reusing some of the code from `mylm`, we can easily obtain the design matrix X and the response y, and run the Fisher scoring algorithm. The algorithm converges very quickly, so 100 iterations should be more than safe. Alternatively, one can implement some stopping criterion based on the absolute difference in $\boldsymbol{\beta}^{(t)}$, or based on the likelihood function for $\boldsymbol{\beta}$.

```r
fit_coefficients <- function(formula, data, contrasts = NULL, ...) {
    # Extract model matrix & responses
    mf <- model.frame(formula = formula, data = data)
    X <- model.matrix(attr(mf, "terms"), data = mf, contrasts.arg = contrasts)
    y <- model.response(mf)
    terms <- attr(mf, "terms")

    beta <- rep(0, ncol(X))
    for (i in 1:100) {
        beta <- fisher_score_iteration(beta, X, y)
    }
    return(beta)
}
```

This leads to the following coefficient estimates:

```r
my.coeffs <- fit_coefficients(y ~ team + is_home, data = eliteserie.played.transf)
my.coeffs
```

```
##            (Intercept)              teamBrann              teamHamKam
##              0.7746635             -0.4114745             -0.6410489
##          teamHaugesund        teamLillestrøm              teamMolde
##             -1.1002138             -0.3455995             -0.1718503
##                teamOdd         teamRosenborg teamSandefjord Fotball
##             -0.7637415             -0.7439463             -0.5166907
##       teamSarpsborg 08             teamStabæk       teamStrømsgodset
##             -0.3823740             -1.1147326             -0.8010750
##             teamTromsø             teamViking          teamVålerenga
##             -0.5848816             -0.1630131             -0.6579742
##           teamAalesund            is_homeTRUE
##             -1.3386534              0.3555010
```

Asserting that they are indeed correct using the built in glm function in R:

```r
ref.coeffs <- glm(y ~ team + is_home, data = eliteserie.played.transf,
    family = "poisson")$coefficients
abs(mean(my.coeffs - ref.coeffs))
```

```
## [1] 2.407804e-11
```

Looks good!

d)

In order to simulate 1000 matches, we first need to add the coefficient of the reference level to `my.coeffs`.

```r
ref_coeff <- eliteserie$home %>%
    unique() %>%
    lapply(function(str) glue("team{str}")) %>%
    unlist() %>%
```

```
    setdiff(names(my.coeffs))

my.coeffs[[ref_coeff]] <- 0
```

Simulation can be done in the following steps: 1. Estimate $\hat{\lambda}$ for each team in each match using the model, i.e.

$$\hat{\lambda_{i,H}} = \exp\left(\beta_0 + \beta_H + \beta_H - \beta_A\right)$$
$$\hat{\lambda_{i,A}} = \exp\left(\beta_0 + \beta_A - \beta_H\right)$$

2. For m simulations, and for all i matches, simulate the goals of each team assuming they follow a Poisson distribution with rate $\hat{\lambda}_{i,H}$ and $\hat{\lambda}_{i,A}$ for the home- and away teams respectively.

We will first implement the function to estimate the rate for each match. Since I have created the model a bit differently, I will do this manually by retrieving each respective coefficient.

```
estimate_lambda <- function(match) {
    # Args: match (DataFrame row): Row of the eliteserie dataframe,
    # ya and yh may be excluded Returns: Same row but with lambda
    # estimates.  Retrieving the coefficients
    coeff.home <- my.coeffs[[glue("team{match$home}")]]
    coeff.away <- my.coeffs[[glue("team{match$away}")]]

    # Calculating rates
    match$lambda.h <- exp(my.coeffs[["(Intercept)"]] + my.coeffs[["is_homeTRUE"]] +
        coeff.home - coeff.away)
    match$lambda.a <- exp(my.coeffs[["(Intercept)"]] + coeff.away - coeff.home)

    return(match)
}
```

Now applying the above function to each row, giving us coefficient estimates of λ for each team in every match.

```
eliteserie.unplayed.est <- eliteserie.unplayed %>%
    rename(lambda.h = yh, lambda.a = ya) %>%
    {
        lapply(split(., seq_len(nrow(.))), estimate_lambda)
    } %>%
    {
        do.call(rbind, .)
    }
```

Finally, using the estimated rates in `eliteserie.unplayed.est` to simulate 1 000 matches, and then calculate the scroeboard for all simulations.

```
m <- 1000  # Number of  simulations

simulate_dataframe <- function() {
    eliteserie.unplayed.est %>%
        mutate(yh = rpois(n(), lambda.h), ya = rpois(n(), lambda.a)) %>%
        select(-c(lambda.h, lambda.a))
}
```

```
num_cores <- parallel::detectCores() - 1
cl <- makeCluster(num_cores, type = "SOCK")
registerDoSNOW(cl)

scores.sim <- foreach(i = 1:m, .packages = "tidyverse") %dopar% {
    calculate_scores(rbind(eliteserie.played, simulate_dataframe()))
} %>%
    bind_rows() %>%
    group_by(team) %>%
    summarize(scores = list(score), score.mean = mean(score), score.sd = sd(score))

stopCluster(cl)
```

```
print(length(scores.sim$scores[[1]]))
```

```
## [1] 1000
```

With all realizations being saved in the dataframe `scores.sim`, we can easily summarize the result.

```
scores.sim %>%
    mutate(position = rank(-score.mean, ties.method = "min")) %>%
    select(-scores) %>%
    merge(scores.played[c("team", "position")] %>%
        rename(position.played = position), by = "team") %>%
    mutate(pos_change = position.played - position) %>%
    select(-position.played) %>%
    arrange(position)
```

```
##                  team score.mean score.sd position pos_change
## 1              Viking     70.013 2.701875        1          0
## 2          Bodø/Glimt     69.408 2.400270        2          0
## 3              Tromsø     59.028 2.565501        3          0
## 4               Molde     57.448 3.335029        4          1
## 5               Brann     54.746 3.398976        5         -1
## 6          Lillestrøm     51.286 2.818892        6          0
## 7        Sarpsborg 08     47.020 2.341598        7          0
## 8                 Odd     37.964 2.751777        8          0
## 9  Sandefjord Fotball     36.205 3.116519        9          3
## 10          Rosenborg     35.721 2.494270       10         -1
## 11             HamKam     34.820 3.214332       11          0
## 12       Strømsgodset     34.747 3.176016       12         -2
## 13          Vålerenga     31.042 3.204451       13         -1
## 14          Haugesund     27.155 2.431231       14         -2
## 15             Stabæk     19.860 2.366432       15          0
## 16           Aalesund     12.969 1.500764       16          0
```

It is clear that Viking and Bodø/Glimt are leading by a substantial margin. Furthermore, most teams kept their positions as of October 1st, except for Molde and Brann, who switched. The models ability to estimate change in leaderboard based on the teams playing in the future is probably its best strength. The new leaderboard also appropriately positioned Haugesund, Vålerenga, and Sandefjord.

One can also observe that the estimated standard deviation starts out quite small for the worst teams, and steadily increases with increasing score. This is due to the fact that the last 80 matches (33% of the

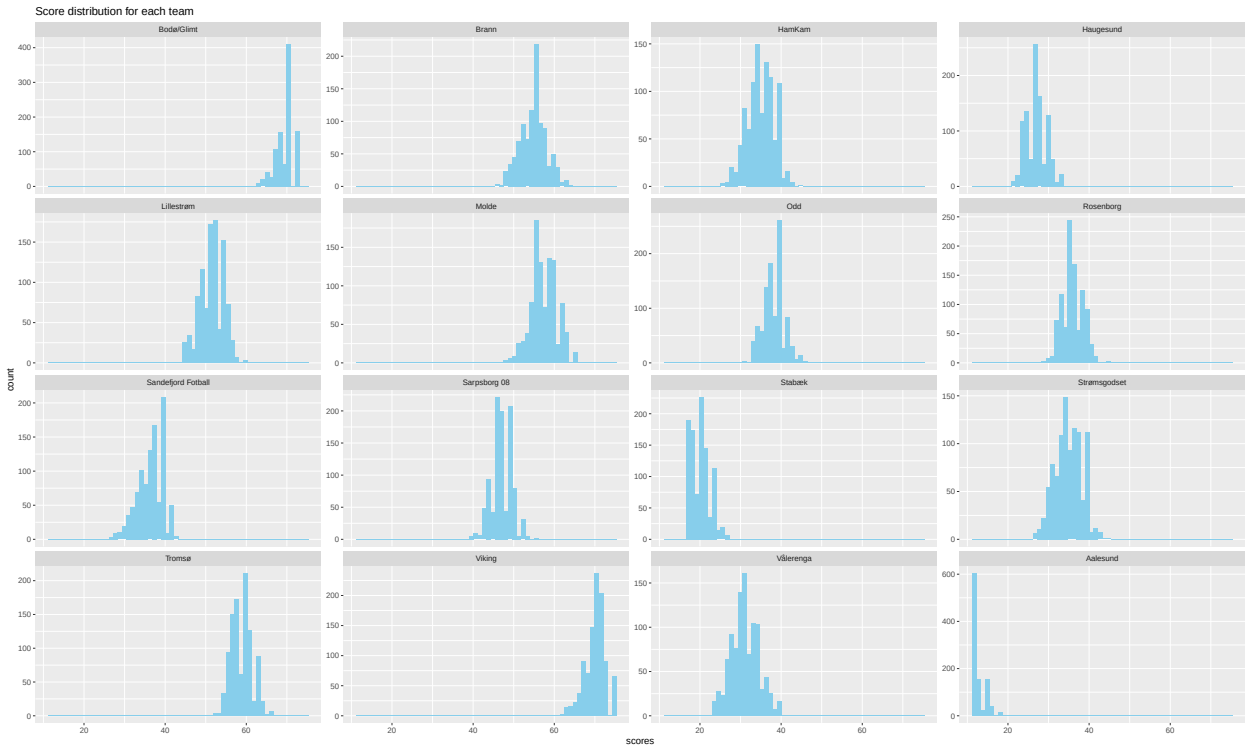
championship) are simulated using a Poisson distribution, and the rest of the data for calculating the score is identical for all simulations. Let S_i be the score for team i. The total variance for team i is given by

$$\text{Var}(S_i) = \text{Var}(\text{Played}_i) + \text{Var}(\text{Simulated}_i) = 0 + \hat{\lambda}_i = \hat{\lambda}_i \tag{6}$$

Where we in (6) used the fact that the variance of observed variables is obviously 0. And thus, the standard deviation for each team is actually the square root of their expected score increase during the last eighty matches (according to the simulation).

Plotting score distribution histograms for all teams:

```
ggplot(scores.sim %>%
    unnest(cols = scores), aes(x = scores)) + geom_histogram(bins = 60,
    fill = "skyblue", color = NA) + facet_wrap(~team, scales = "free_y") +
    labs(title = "Score distribution for each team")
```



The plots above show that the simulated score for each team is approximately normal.