# Compulsory exercise 2: Group 13

TMA4315: Generalized linear (mixed) models H2023

Benjamin Sigbjørnsen, Johan Vik Mathisen, and Martinius Theodor Singdahlsen

09 October, 2023

Packages:

```
library(car)
library(ggplot2)
library(GGally)
library(mylm)
library(reshape2)
library(kableExtra)
library(dplyr)

set.seed(42)
```

## Part 2)

```
# Reading the data
rm(list = ls())
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2023h/eliteserien2023.csv"
eliteserie <- read.csv(file = filepath)

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

### a)

To test the assumption of independence, we have to perform the test

$H_0$ : Independence between the goals made by the home and away teams $H_1$: Dependence between the goals made by the home and away teams

We test using a Pearson's $\chi^2$ test on the contingency table of the goals

```
# Creating contingency table
contingencyTab = table(eliteserie$yh, eliteserie$ya)
# Testing the independence between the goals made by the home and away teams
chisq.test(contingencyTab)
```

```
##
##  Pearson's Chi-squared test
##
## data:  contingencyTab
## X-squared = 33.527, df = 35, p-value = 0.5393
```

The p-value of the test is 0.54, and we therefore keep our null hypothesis using every reasonable significance level.

## b)

The preliminary ranking for the season as of October 1st is presented in table 1.

```
tableFunc = function(eliteserie) {
    # Removing matches that are not played
    eliteserie = eliteserie[!is.na(eliteserie$yh), ]

    # Points home team
    eliteserie$ph = 0
    eliteserie[eliteserie$yh > eliteserie$ya, ]$ph = 3
    eliteserie[eliteserie$yh == eliteserie$ya, ]$ph = 1

    # Points away team
    eliteserie$pa = 1
    eliteserie[eliteserie$ph != eliteserie$pa, ]$pa = 3 - eliteserie[eliteserie$ph !=
        eliteserie$pa, ]$ph

    # Creating table
    table = data.frame(team = c(eliteserie$home[1:8], eliteserie$away[1:8]), points = 0,
        goal_difference = 0)

    # Giving teams points
    for (i in 1:nrow(eliteserie)) {
        teamHome = eliteserie$home[i]
        teamAway = eliteserie$away[i]
        table[table$team == teamHome, ]$points = table[table$team == teamHome, ]$points +
            eliteserie$ph[i]
        table[table$team == teamAway, ]$points = table[table$team == teamAway, ]$points +
            eliteserie$pa[i]
        table[table$team == teamHome, ]$goal_difference = table[table$team == teamHome,
            ]$goal_difference + eliteserie$yh[i] - eliteserie$ya[i]
        table[table$team == teamAway, ]$goal_difference = table[table$team == teamAway,
            ]$goal_difference + eliteserie$ya[i] - eliteserie$yh[i]
    }

    # Sorting by points, and then goal difference
    table = table[order(table$points, table$goal_difference, decreasing = TRUE),
        ]
    return(table)
}
table = tableFunc(eliteserie)
rownames(table) <- 1:nrow(table)
table %>%
    kbl(caption = "Preliminary ranking for the season as of October 1st \\label{tab:prelim_rank}") %>%
    kable_classic(full_width = F, html_font = "Cambria")
```

## c)

In this exercise we implement a function estimating the intercept, home advantage and strength parameters for each team in Tippeligaen 2023 based on the 192 games played as of October 1.

Table 1: Preliminary ranking for the season as of October 1st

| team | points | goal_difference |
|---|---|---|
| Viking | 51 | 21 |
| Bodø/Glimt | 49 | 28 |
| Tromsø | 48 | 11 |
| Brann | 42 | 14 |
| Molde | 41 | 24 |
| Lillestrøm | 36 | 7 |
| Sarpsborg 08 | 34 | 5 |
| Odd | 30 | -3 |
| Rosenborg | 29 | -6 |
| Strømsgodset | 27 | -3 |
| HamKam | 24 | -15 |
| Vålerenga | 21 | -8 |
| Sandefjord Fotball | 21 | -9 |
| Haugesund | 21 | -14 |
| Stabæk | 17 | -16 |
| Aalesund | 12 | -36 |

We build our design matrix $X$ as explained by the hint in the project description.

```
# Removing nans.
eliteserie_clean <- na.omit(eliteserie)
# Extracting nr of games played so far this season.
games_played = dim(eliteserie_clean)[1]
# Extracting team names.
teams = as.array(unique(eliteserie$home))


# Initialize the design matrix X, which is a 2*192 x 18 matrix.  Each row in X
# corresponds to a played game. Once from each side. Therefore 2*games_played
# rows.

# Initialize all zeros matrix
X = matrix(0, nrow = games_played * 2, ncol = 18)

# Set proper feature names
colnames(X) = c("Intercept", "HomeAdvantage", teams)

# First we alter the design matrix for all 'home games'
for (i in 1:games_played) {
    # Intercept
    X[i, 1] = 1

    # HomeAdvantage = 1
    X[i, 2] = 1

    # x_homeTeam = 1
    X[i, match(eliteserie_clean[i, 2], colnames(X))] = 1

    # x_awayTeam = -1
    X[i, match(eliteserie_clean[i, 3], colnames(X))] = -1
```

```r
}

# Then alter design matrix for all 'away games'
for (i in 1:games_played) {

    # Intercept
    X[games_played + i, 1] = 1

    # Same as above, but inverted:

    # x_home = 0 from initialization.

    # x_homeTeam = -1
    X[games_played + i, match(eliteserie_clean[i, 2], colnames(X))] = -1

    # x_awayTeam is set to 1
    X[games_played + i, match(eliteserie_clean[i, 3], colnames(X))] = 1
}

# Response vector, all goals.
Y = c(eliteserie_clean$yh, eliteserie_clean$ya)
Y = matrix(Y, length(Y), 1)


# Removed to make mle solution unique and optim to work.
X <- X[, -which(colnames(X) == "Bodø/Glimt")]
```

For parameter estimation we use maximum likelihood estimation. For a Poisson regression model we know that the MLE does not have a closed form solution, hence we turn to numerics in order to obtain our estimates.

Since out 16 team covariates sum to 0, the MLE has infinitely many solutions. We do at proposed by the exercise text and force $x_{\text{Bodø/Glimt}}$ to be zero, leaving us with a linearly independent set of covariates. In out model the strength parameter for Bodø/Glimt will then be the reference value 0.

The log likelihood of a Poisson sample is

$$l(\beta) = \sum_{i=1}^{n} \left[ y_i \ln(\lambda_i) - \lambda_i - \ln(y_i!) \right].$$

As we use the `R` function `Optim`, and this by default minimizes, we implement the negative log likelihood below.

By including the data-dependent term $\ln(y_i!)$ in our calculations we got extremely accurate results, and therefore choose not to include gradient information to further increas accuracy.

```r
neg_log_likelihood_poisson <- function(beta) {
    eta <- X %*% beta
    lambda <- exp(eta)
    return(-(t(Y) %*% eta - sum(lambda - log(factorial(Y)))))
}
my_poisson <- function(X, Y) {
    "
X: Design matrix
Y: Response (possion distributed)
```

Table 2: Power ranking based on Poisson regresson coefficients.

| Names | Estimate |
|---|---|
| Bodø/Glimt | 0.00000 |
| Molde | -0.10314 |
| Viking | -0.12260 |
| Brann | -0.24826 |
| Tromsø | -0.26013 |
| Lillestrøm | -0.32865 |
| Sarpsborg 08 | -0.34396 |
| Odd | -0.44285 |
| Strømsgodset | -0.47908 |
| Sandefjord Fotball | -0.50867 |
| Rosenborg | -0.51028 |
| Vålerenga | -0.52032 |
| Haugesund | -0.57215 |
| Stabæk | -0.63302 |
| HamKam | -0.63641 |
| Aalesund | -0.91919 |

```
"
    # Initial value for beta.
    beta_init <- rep(0.5, 17)

    beta <- optim(beta_init, neg_log_likelihood_poisson, method = "BFGS")$par
    result <- data.frame(Names = colnames(X), Estimate = round(beta, 5))
    return(result)

}
result <- my_poisson(X, Y)

# Remove intercept and homeAdvantage.
power_ranking <- result[-(1:2), ]
# Adding Bodø/Glimt back
power_ranking <- rbind(power_ranking, list(Names = "Bodø/Glimt", Estimate = 0))

# Order teams by power parameter.
power_ranking <- power_ranking[order(power_ranking[, 2], decreasing = TRUE), ]
rownames(power_ranking) <- 1:nrow(power_ranking)

# Power rankings
power_ranking %>%
    kbl(caption = "Power ranking based on Poisson regresson coefficients.\\label{tab:power_rank}") %>%
    kable_classic(full_width = F, html_font = "Cambria")
```

```
# Parameter estimates
result %>%
    kbl(caption = "Estimates for the regressson coefficients.\\label{tab:reg_coeff}") %>%
    kable_classic(full_width = F, html_font = "Cambria")
```

**Preliminary ranking vs strength parameter ranking** When we compare the ranking based on the estimated strengths presented in table 2 to the preliminary standings in table 1, we observe that the strength

Table 3: Estimates for the regressson coefficients.

| Names | Estimate |
|---|---|
| Intercept | 0.16644 |
| HomeAdvantage | 0.35856 |
| Rosenborg | -0.51028 |
| Aalesund | -0.91919 |
| HamKam | -0.63641 |
| Sarpsborg 08 | -0.34396 |
| Stabæk | -0.63302 |
| Tromsø | -0.26013 |
| Brann | -0.24826 |
| Lillestrøm | -0.32865 |
| Viking | -0.12260 |
| Haugesund | -0.57215 |
| Molde | -0.10314 |
| Odd | -0.44285 |
| Sandefjord Fotball | -0.50867 |
| Strømsgodset | -0.47908 |
| Vålerenga | -0.52032 |

parameter follows the goal difference rather than the points. This is not very surprising as our model only takes goal difference into account.

**Our results compared to `glm()`** From the summary of the `glm()` model below and our estimated regression coefficients presented in table 3, we see that the estimated coefficients are identical. This leaves us confident in the correctness of our implementation.

```
##
## Call:
## glm(formula = Y ~ -1 + X, family = "poisson")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2652  -1.1581  -0.1047   0.5316   2.4022
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## XIntercept           0.16644    0.06849   2.430 0.015091 *
## XHomeAdvantage       0.35856    0.08745   4.100 4.13e-05 ***
## XRosenborg          -0.51028    0.16443  -3.103 0.001914 **
## XAalesund           -0.91919    0.16662  -5.517 3.46e-08 ***
## XHamKam             -0.63641    0.16580  -3.838 0.000124 ***
## XSarpsborg 08       -0.34396    0.16839  -2.043 0.041089 *
## XStabæk             -0.63302    0.16909  -3.744 0.000181 ***
## XTromsø             -0.26013    0.16452  -1.581 0.113831
## XBrann              -0.24826    0.16821  -1.476 0.139978
## XLillestrøm         -0.32865    0.16986  -1.935 0.053014 .
## XViking             -0.12260    0.16428  -0.746 0.455518
## XHaugesund          -0.57215    0.16436  -3.481 0.000499 ***
## XMolde              -0.10314    0.16687  -0.618 0.536537
## XOdd                -0.44285    0.16478  -2.688 0.007197 **
## XSandefjord Fotball -0.50867    0.17038  -2.986 0.002830 **
```

```
## XStrømsgodset       -0.47908    0.17080  -2.805 0.005032 **
## XVålerenga          -0.52032    0.16594  -3.136 0.001715 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 555.74  on 358  degrees of freedom
## Residual deviance: 397.07  on 341  degrees of freedom
## AIC: 1102
##
## Number of Fisher Scoring iterations: 5
```

## d)

In this exercise we estimate the rankings by means of simmulation, based on the estimated strengths found in the previous task. We do this by simmulating the remaining 61 matches 1000 times.

In order to obtain the probabilities of the different outcomes of the remaining matches we observe that team $A$ winning over team $B$ is the same as the score $Y_A$ being greater than the score $Y_B$. From our model-assumptions these variables are independent of each other. From this we obtain that

$$\begin{aligned} P(A \text{ winns over } B) &= P(A \text{ scores more than } B) \\ &= \sum_{k=1}^{\infty} P(Y_A = k, Y_B < k) \\ &= \sum_{k=1}^{\infty} \sum_{j=1}^{k} P(Y_A = k) P(Y_B = j) \end{aligned}$$

Similarly for draw and loss we have

$$P(A \text{ loses to } B) = \sum_{k=1}^{\infty} \sum_{j=1}^{k} P(Y_B = k) P(Y_A = j),$$

and

$$P(\text{Draw}) = \sum_{k=0}^{\infty} P(Y_A = k) P(Y_B = k).$$

Let $A$ be the home team. Then from our model we have that $Y_A$ is Poisson with

$$\ln E(Y_A) = \ln(\lambda_A) = \eta_A = \beta_0 + \beta_{\text{home}} + \beta_A - \beta_B$$

and $Y_B$ is Poisson with

$$\ln E(Y_B) = \ln(\lambda_B) = \eta_B = \beta_0 - \beta_A + \beta_B$$

when $A$ plays against $B$. From this we can calculate the desired probabilities.

The sums involved when calculating the probabilities converge extremely quickly, even for $\lambda_{\text{Bodø/Glimt}} = 4.238418$, which is the largest of all teams, the summands are of magnitude less than $10^{-8}$ for $k = 20$. Thus we take the liberty to say that the sums have converged when $k = 20$.

```r
predict_season <- function(Unplayed, Estimates) {
    "
  Unplayed: home, away
  Estimates: df of poisson regression estimates with names.

  returns: df with match outcome probabilities. Home, Away, Win, Draw, Loss
  "
    Predict <- data.frame(Home = character(0), Away = character(0), Win = numeric(0),
        Draw = numeric(0), Loss = numeric(0))

    for (i in 1:nrow(Unplayed)) {
        home <- Unplayed$home[i]
        away <- Unplayed$away[i]
        p_w <- prob_win(home, away, Estimates)
        p_d <- prob_draw(home, away, Estimates)
        p_l <- prob_loss(home, away, Estimates)
        data <- list(Home = home, Away = away, Win = p_w, Draw = p_d, Loss = p_l)
        Predict <- rbind(Predict, data)
    }
    return(Predict)
}

sim <- function(Predict) {
    "
  Uses Predict df to simulate the remaining season
  "
    outcome <- apply(Predict[, -(1:2)], 1, function(row) {
        return(sample(c("Win", "Draw", "Loss"), 1, prob = row))
    })
    return(outcome)
}
```

```r
predict_to_table <- function(Predict, outcome, current_standings) {
    "
    Updates the standings according to the
    simulation.

    Predict: Home, Away, probabilities,
    outcome: Sim; result of simulation

    current_standings: Team, points
    "
    Predict <- cbind(Predict, Sim = outcome)

    table <- current_standings
    for (i in 1:nrow(Predict)) {
        if (Predict$Sim[i] == "Win") {
            table[table$team == Predict$Home[i], 2] <- table[table$team == Predict$Home[i],
                2] + 3
        } else if (Predict$Sim[i] == "Loss") {
            table[table$team == Predict$Away[i], 2] <- table[table$team == Predict$Away[i],
                2] + 3
        } else {
            # Draw
```

```
            table[table$team == Predict$Home[i], 2] <- table[table$team == Predict$Home[i],
                2] + 1
            table[table$team == Predict$Away[i], 2] <- table[table$team == Predict$Away[i],
                2] + 1
        }
    }
    table <- table[order(table$points, decreasing = TRUE), ]
    return(table)
}
```

```
sim_season_finish <- function(number_of_sims = 1000, current_standing, Unplayed,
    Estimate) {
    "
Combines the above functions to simulate the season number_of_sims times.
"
    Predict <- predict_season(Unplayed, Estimate)
    outcome <- sim(Predict)
    table <- predict_to_table(Predict, outcome, current_standing)
    for (i in 1:number_of_sims) {
        outcome <- sim(Predict)
        table_1 <- predict_to_table(Predict, outcome, current_standing)
        table <- inner_join(table, table_1, by = "team")
    }
    # Renames the columns such that results from simulation i is in column 'i'
    colnames(table)[2:ncol(table)] <- 1:(ncol(table) - 1)
    as.data.frame(table)
    return(table)
}
```

```
get_summary_data <- function(sim_result) {
    "
  Counts the number of times each team finishes at the top of the table.
  Calculates mean points obtained for each team.
  Calculates standard deviation of points for each team.
  "
    N <- ncol(sim_result) - 1  #not including team column. This is the number of sims.
    series_win <- data.frame(`SeasonsWon(%)` = rep(0, length(sim_result$team)))
    rownames(series_win) <- sim_result$team
    for (i in 2:ncol(res)) {
        series_win[res[which.max(sim_result[, i]), 1], ] <- series_win[sim_result[which.max(res[,
            i]), 1], ] + 1
    }
    mean <- apply(sim_result[, -1], 1, function(row) {
        return(mean(row))
    })
    std.d <- apply(sim_result[, -1], 1, function(row) {
        return(sd(row))
    })
    summary_data <- list(Mean = mean, Std.dev = std.d, SeasonsWon = series_win/N)
    summary_data <- as.data.frame(summary_data)

    return(summary_data[order(-summary_data$Mean), ])
}
```

Table 4: Summary data from 1000 end of season simulations.

| | Mean | Std.dev | SeasonsWon... |
|---|---|---|---|
| Bodø/Glimt | 66.45854 | 3.236129 | 0.4685315 |
| Viking | 66.28172 | 3.465913 | 0.5174825 |
| Tromsø | 59.75125 | 3.124910 | 0.0129870 |
| Molde | 55.46853 | 3.527784 | 0.0009990 |
| Brann | 54.10989 | 3.667412 | 0.0000000 |
| Lillestrøm | 47.99800 | 3.487979 | 0.0000000 |
| Sarpsborg 08 | 44.83017 | 3.240236 | 0.0000000 |
| Odd | 39.75125 | 3.230025 | 0.0000000 |
| Rosenborg | 36.88312 | 3.145048 | 0.0000000 |
| Strømsgodset | 36.37063 | 3.367120 | 0.0000000 |
| Sandefjord Fotball | 32.34466 | 3.502298 | 0.0000000 |
| HamKam | 31.36364 | 3.176104 | 0.0000000 |
| Vålerenga | 30.93906 | 3.550392 | 0.0000000 |
| Haugesund | 30.18482 | 3.149732 | 0.0000000 |
| Stabæk | 25.09291 | 3.400347 | 0.0000000 |
| Aalesund | 15.29371 | 2.385718 | 0.0000000 |

In table 4 we see mean number of points and the associated standard deviations, together with the proportion of seasons won by each of the teams in the 1000 simmulated season endings. The low standard deviation stands to us out as the least true to reality for a game of this nature. In figure 2 and 3 we see the empirical point distributions for each of the teams, both jointly and separately.

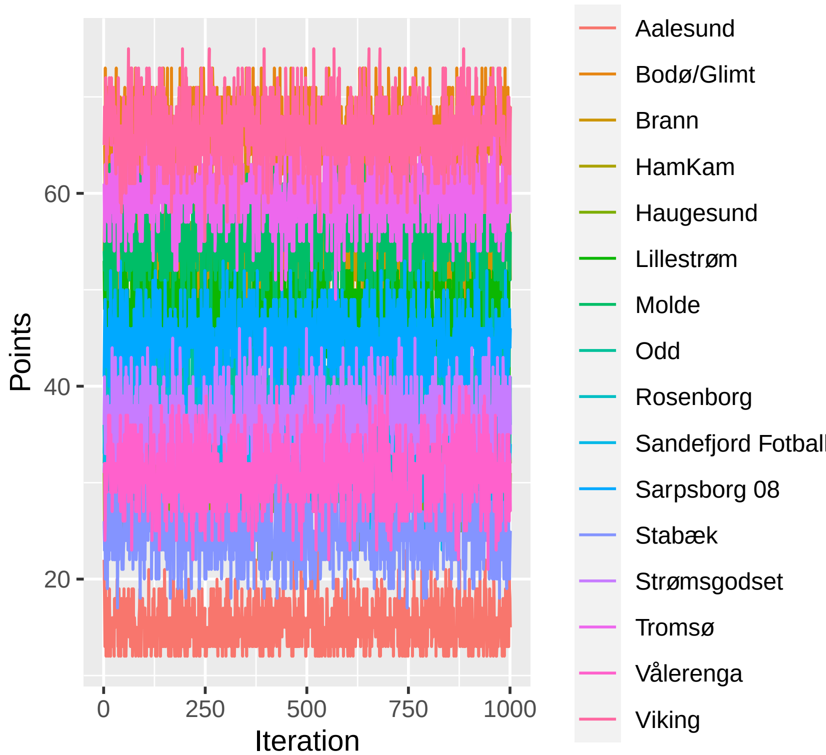


Figure 1: Simulated points at the end of the season, given the standings as of Oct. 1.

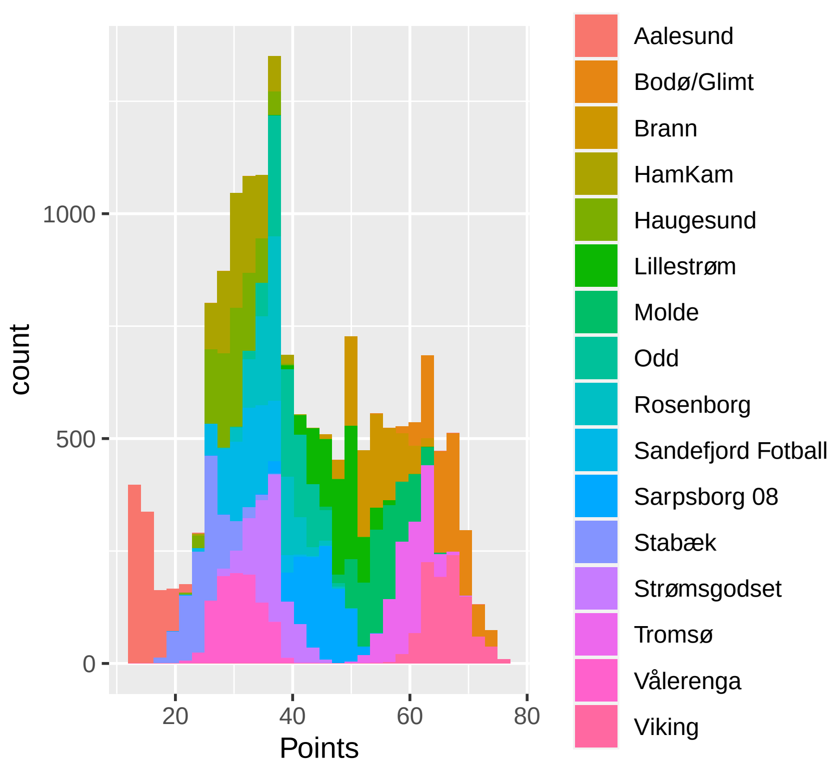


Figure 2: End of season point distribution after 1000 simulations, given the standings as of Oct. 1.

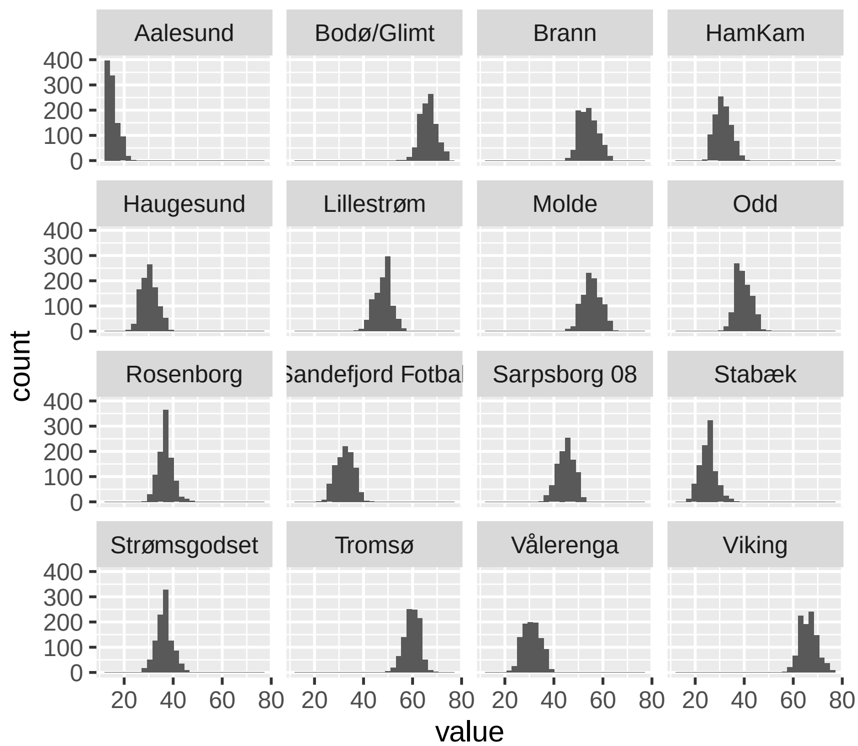


Figure 3: End of season point distribution after 1000 simulations for the individual teams, given the standings as of Oct. 1.