TMA4315: Compulsory exercise 2: Logistic regression and Poisson regression

Group 14: Adrian Bruland, Mathias Opland

26.10.2018

Contents

Part 1

a)

For logistic regression with grouped data, the log-likelihood function is

$$\ell(\beta) = \ln(L(\beta)) = \sum_{i=1}^{G}(y_i \mathbf{x}_i^T\beta - \mathbf{x}_i^T\beta \ln(1 + \exp(\mathbf{x}_i^T\beta))$$

where $\mathbf{x}_i$ is the p-vector whose elements are 1 followed by the k covariate values for observation i and G is the number of groups. As usual, β is the vector of regression parameters to be found.

In practice, in order to get the maximum likelihood estimates, we plug the observations $(y_i, \mathbf{x}_i)$ for $i = 1, ..., n$, into the derivative of ℓ with respect to β, called the score function

$$s(\beta) = \sum_{i=1}^{n} \frac{\partial \ell_i(\beta)}{\partial \beta} = \sum_{i=1}^{n} \mathbf{x}_i(y_i - n_i\pi_i) = \sum_{i=1}^{n} \mathbf{x}_i\Big(y_i - \mathbf{x}_i\beta\frac{\exp(\mathbf{x}_i\beta)}{1+\exp(\mathbf{x}_i\beta)}\Big)$$

.

This is a non-linear system of equation - set it equal to 0 and solve numerically, by the Newton-Raphson method or the Fisher scoring method.

b)

Fit the model

```
##
## Call:
## glm(formula = cbind(success, fail) ~ height + prominence, family = "binomial",
##     data = mount)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.2886  -0.8086   0.6893   1.4226   3.7456
##
## 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
```

The regression parameters β_i are interpretted respectively as the "log odds" (natural logarithm of the odds) when height and prominence are zero, the log of the factor by which the odds will be multiplied if height increases by one unit, and the analagous log-factor for prominence. Here $\beta_1 = \beta_{\text{height}}$ and $\beta_2 = \beta_{\text{prominence}}$ are negative, so an increase in height or prominence will, according to the model, yield a decrease in success odds.

The exponential function with the coeffients as input gives

```
##  (Intercept)       height   prominence
## 8.783892e+05 9.983659e-01 9.998260e-01
```

demonstrating that the odds will decrease as height and prominence increase, since the odds will be mulitplied by a factor between 0 and 1.

Significance: since the Wald test statistic is the square of the z-test statistic, we can look at the glm summary and look at the p-values, which are below 0.03112, so the covariates are both significant.

Confidence Intervals:

Choosing the standard significance level of $\alpha = .05$, the confidence level is 95%, and the respective confidence interval is

$$P\left(\hat{\beta}_1 - z_{\alpha/2}\sqrt{a_{jj}(\hat{\boldsymbol{\beta}})} \leq \beta_1 \leq \hat{\beta}_1 + z_{\alpha/2}\sqrt{a_{jj}(\hat{\boldsymbol{\beta}})}\right) = 0.95$$

Calculate the confidence interval for height covariate β_1 (lower and upper shown here):

```
##                [,1]         [,2]
## height -0.001913631 -0.001357205
```

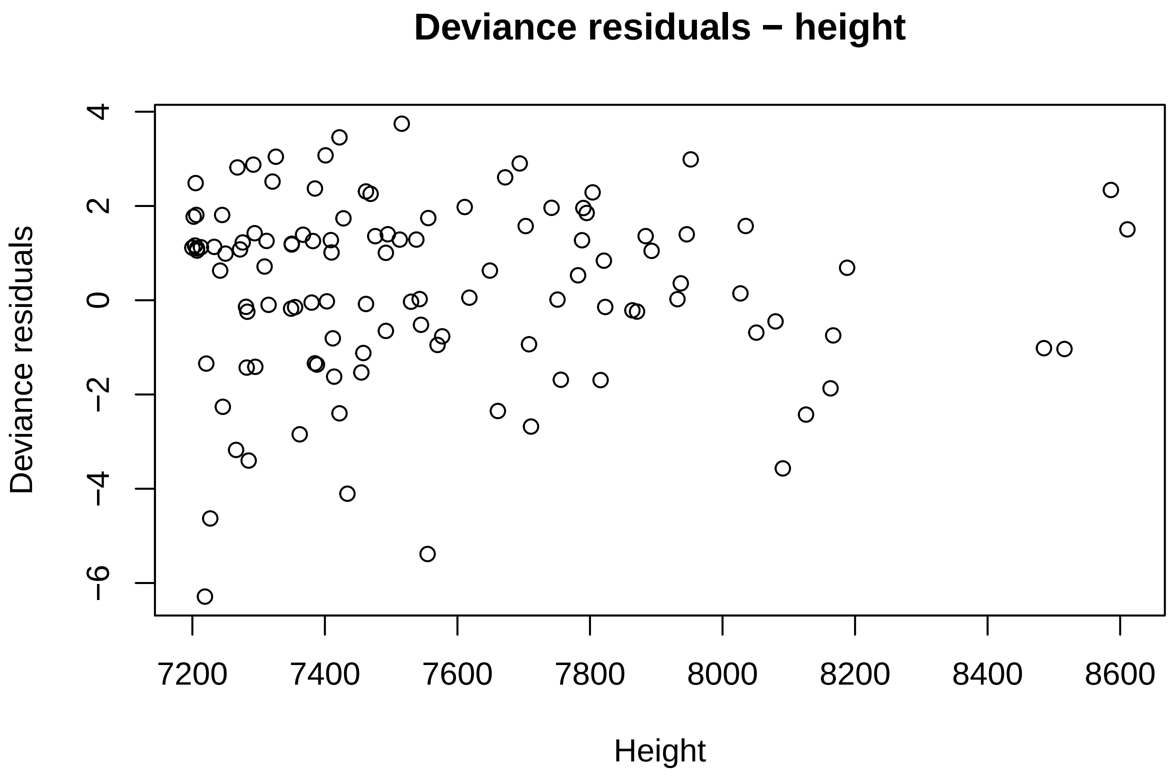
Find the confidence level for $\exp(\beta_1)$:

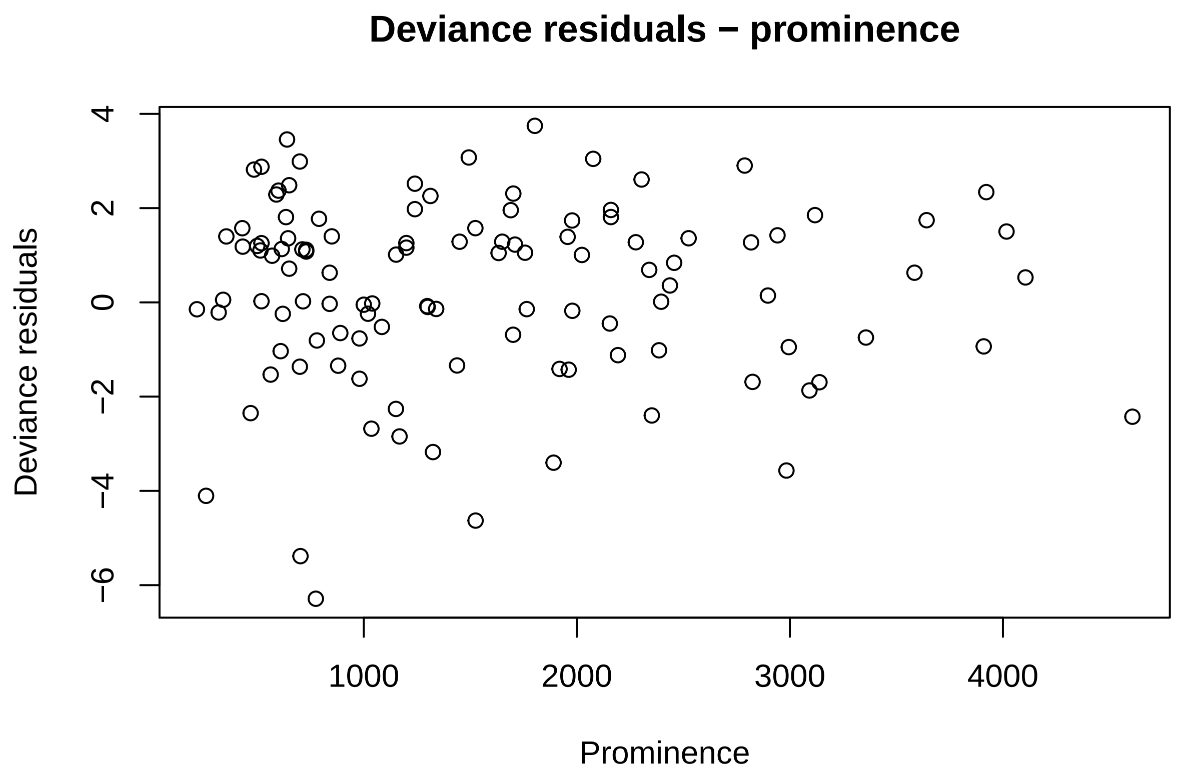
```
##             [,1]      [,2]
## height 0.9980882 0.9986437
```

There is a 95% probability that increasing the height covariate by one unit will change the odds of successfully scaling a mountain by a factor in the $\exp(\beta_1)$-interval given above, which will constitute a decrease, althought not by much. One more meter of height isn't that much of a challenge, but a much higher mountain will prove substantially more difficult.

c)

Plot the deviance residuals as a function of the two covariates, separately.





Both plots seem to show some clustering above the zero deviance residual line, particularly for smaller values in the top left corner. This suggests that for the mountains in the data set whose height/prominence is lower, the odds of success is greater than what the logit model tells us. However, there are a few outliers whose observed success rate is a lot lower than that predicted by the model, explaining why the model failed to predict the high odds of success for the other low-height or low-prominence mountains. The variance looks to be higher for the mountians lower height and/or prominence.

For the logit model, the deviance test is

$$D = 2\sum_{j=1}^{G}\left[y_j \ln(\frac{y_j}{n_j\hat{\pi}_j}) + (n_j - y_j)\ln\left(\frac{n_j - y_j}{n_j - n_j\hat{\pi}_j}\right)\right]$$

which is asymptotically distributed as χ^2_{G-p}. Use deviance test to assess model fit:

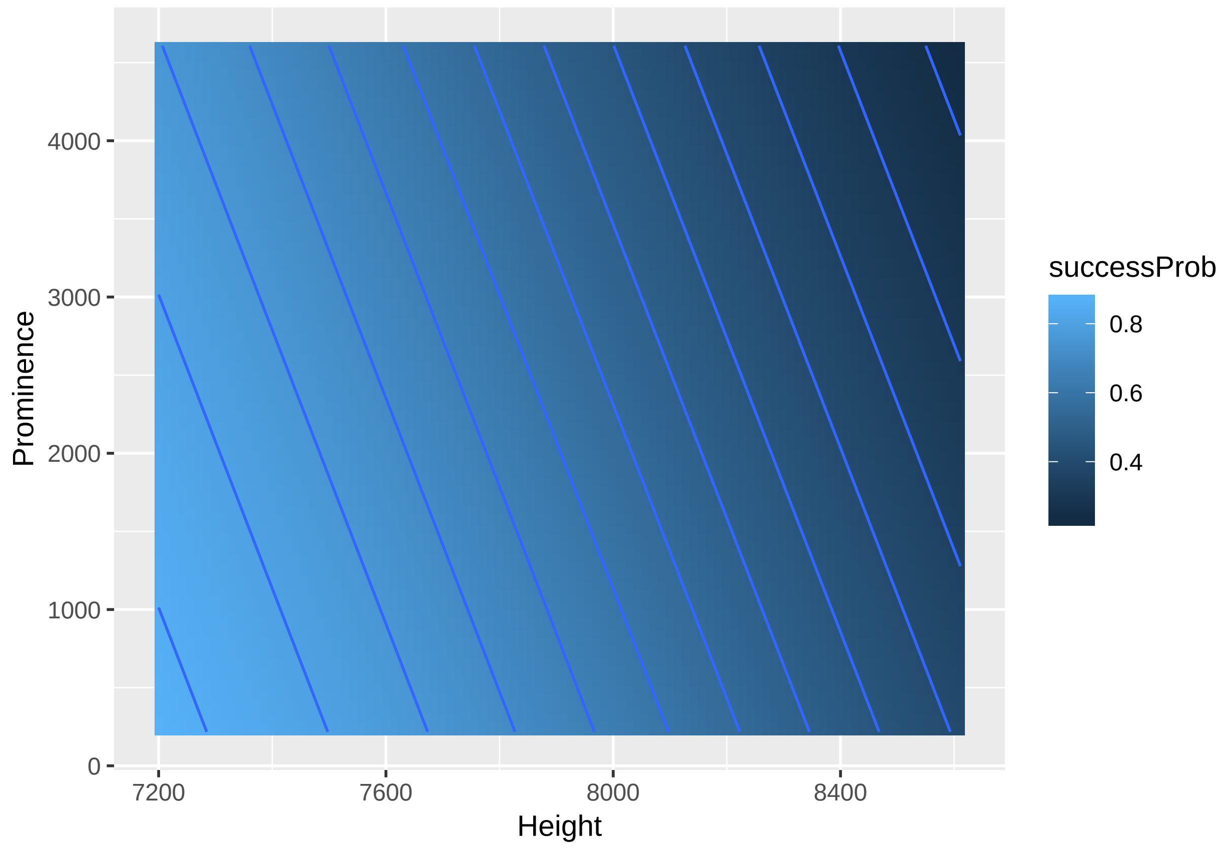
```
## [1] 414.6842
```

Find the p-value, by evaluating $1 - \chi^2_{G-p}$ for D:

```
## [1] 0
```

With a p-value of zero, we conclude that the chosen logit is not a good fit.

Next, we plot the estimated probabilities of successfully climbing a mountain in the covariate range of the logit model



As expected, the logit model graph shows a decrease in success probability with increasing height and prominence. The contour lines show that one unit of height can be traded off for about 3 units of prominence while keeping success probability stable. This means that height has a more adverse effect on success probability than prominence. This may be explained by the fact that a substantially higher altitude tends to give more adverse weather conditions, lack of oxygen, stronger winds and so forth.

d)

Mount Everest: According to the model, the probability of getting to the top of Mount Everest is

```
##             [,1]
## [1,] 0.08917783
```

We know that $\hat{\beta}$ has a normal distribution asymptotically

$$\hat{\beta} \sim N_p(\beta, F^{-1}(\hat{\beta}))$$

where F is the Expected Fisher information matrix. The estimate of the linear predictor for mount everest is $\hat{\eta} = x^T\hat{\beta}$. Since x_*^T is just a constant vector of covariate values, $\hat{\eta}_*$ is also normally distributed. The expected value and variance of $\hat{\eta}$ are given as

$$E(\hat{\eta}) = x^T E(\hat{\beta}) = x^T\beta = \eta$$

$$Var(\hat{\eta}) = x^T Var(\hat{\beta})x = x^T F^{-1}(\hat{\beta})x$$

The $(1-\alpha)\%$-confidence interval for η is

$$[\hat{\eta}_l, \hat{\eta}_u] = \left[\hat{\eta} - 1.96\sqrt{x^T F^{-1}(\hat{\beta})x}, \hat{\eta} + 1.96\sqrt{x^T F^{-1}(\hat{\beta})x}\right].$$

Transform this interval into an interval for the likelihood for successfully scaling Mount Everest:

$$[\hat{\pi}_l, \hat{\pi}_u] = \left[\frac{\exp\hat{\eta}_l}{1+\exp\hat{\eta}_l}, \frac{\exp\hat{\eta}_u}{1+\exp\hat{\eta}_u}\right]$$

We couldn't get the confidence interval code right, but we guess that the model doesn't fit Mount Everest too well, since it's well outside the dataset in terms of height. As the height increases, the probability of success will eventually get to zero and below. A negative probability makes no sense, so the model is clearly limited in scope.

Part 2

a)

According to the χ^2-test performed by the author of (https://www.math.ntnu.no/emner/TMA4315/2017h/Lee1997.pdf), there is no evidence against the assumption of independence between goals scored by the home and the away team. We have of course other data, and can perform our own χ^2-test to evaluate if the assumption of independence is reasonable or not.

```
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2018h/eliteserien2018"
eliteserie <- read.table(file = filepath, header = TRUE, colClasses = c("character",
    "character", "numeric", "numeric"))
contigency_data <- eliteserie
for (i in 1:length(contigency_data$home)) {
    if (contigency_data[i, ]$yh > 4) {
        contigency_data[i, ]$yh <- 4
    }
    if (contigency_data[i, ]$ya > 4) {
        contigency_data[i, ]$ya <- 4
    }
}
contigency_table <- table(contigency_data$ya, contigency_data$yh)
rownames(contigency_table) <- c("0", "1", "2", "3", "4+")
colnames(contigency_table) <- c("0", "1", "2", "3", "4+")
contigency_table
```

```
##
##        0  1  2  3 4+
##   0    8 19 10 13  8
##   1   18 26 14 10  7
##   2    3 15 13  7  3
##   3    1  5  4  2  1
##   4+   1  3  1  0  0
```

```
chisq.test(contigency_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  contigency_table
## X-squared = 14.156, df = 16, p-value = 0.5871
```

As we can see in our χ^2-test, the p-value (0.5871) is high, which supports the H_0-hypoteses that the goals scored by the home and the away team is independent. We would have to consider some dependency if the p-value had been under our siginficance value at 0.05. We have clustered goals of 4 or higher in the same row/column as the χ^2-test will be more reliable if there is more than 5 occurrences in each row and column. Thus we can proceed our regression with the assumption of indepencende between goals scored.

b)

```r
# This function computes the ranking and the goal difference, and
# returns a result table (not ordered)
calculate_points <- function(data_eliteserie) {
    result_table <- data.frame(team <- unique(data_eliteserie$home),
        position <- seq(1, length(team), by = 1), goal_for <- rep(0,
            length(team)), goal_against <- rep(0, length(team)), goal_score <- rep(0,
            length(team)), points <- rep(0, length(team)))
    colnames(result_table) <- c("Team", "Position", "GF", "GA", "GD",
        "Points")
    for (i in 1:length(data_eliteserie$home)) {
        if (data_eliteserie$yh[i] > data_eliteserie$ya[i]) {
            index = which(result_table$Team == data_eliteserie$home[i])
            result_table[index, ]$Points = result_table[index, ]$Points +
                3
        } else if (data_eliteserie$yh[i] < data_eliteserie$ya[i]) {
            index = which(result_table$Team == data_eliteserie$away[i])
            result_table[index, ]$Points = result_table[index, ]$Points +
                3
        } else {
            result_table[which(result_table$Team == data_eliteserie$home[i]),
                ]$Points = result_table[which(result_table$Team == data_eliteserie$home[i]),
                ]$Points + 1
            result_table[which(result_table$Team == data_eliteserie$away[i]),
                ]$Points = result_table[which(result_table$Team == data_eliteserie$away[i]),
                ]$Points + 1
        }
        result_table[which(result_table$Team == data_eliteserie$home[i]),
            ]$GF = result_table[which(result_table$Team == data_eliteserie$home[i]),
            ]$GF + data_eliteserie$yh[i]
        result_table[which(result_table$Team == data_eliteserie$home[i]),
            ]$GA = result_table[which(result_table$Team == data_eliteserie$home[i]),
            ]$GA + data_eliteserie$ya[i]
        result_table[which(result_table$Team == data_eliteserie$away[i]),
            ]$GF = result_table[which(result_table$Team == data_eliteserie$away[i]),
            ]$GF + data_eliteserie$ya[i]
        result_table[which(result_table$Team == data_eliteserie$away[i]),
            ]$GA = result_table[which(result_table$Team == data_eliteserie$away[i]),
            ]$GA + data_eliteserie$yh[i]
    }
    result_table$GD <- result_table$GF - result_table$GA
    ordered_table <- result_table[order(-result_table$Points, -result_table$GD),
        ]
    i <- 1
    for (team in ordered_table$Team) {
        result_table[which(result_table$Team == team), ]$Position <- i
        i <- i + 1
    }
    return(result_table)
}
result_table <- calculate_points(eliteserie)
result_table <- result_table[order(-result_table$Points, -result_table$GD),
    ]
print(result_table)
```

```
##                 Team Position GF GA  GD Points
## 10          Rosenborg        1 43 20  23     52
## 11              Brann        2 36 23  13     48
## 1               Molde        3 48 30  18     43
## 12          Haugesund        4 36 28   8     41
## 8          Ranheim_TF        5 38 40  -2     38
## 13         Vaalerenga        6 35 37  -2     36
## 3                 Odd        7 35 29   6     34
## 14            Tromsoe        8 35 33   2     33
## 6         Sarpsborg08        9 39 34   5     32
## 7        Kristiansund       10 32 35  -3     31
## 4          BodoeGlimt       11 28 30  -2     27
## 2       Stroemsgodset       12 38 38   0     26
## 9         Lillestroem       13 26 37 -11     25
## 16            Stabaek       14 29 43 -14     23
## 5               Start       15 24 42 -18     23
## 15 Sandefjord_Fotball       16 24 47 -23     15
```

Here is a the table ordered after points, where GF is goales scored, GA is goals against and GD is the goal balance. We can see that Rosenborg is leading with 4 points to Brann and 9 to Molde. The 4 point gap to Brann is not huge, considering there are 6 games left to play, and when Rosenborg and Brann played last time (where Rosenborg had the home advantage), Brann won. They have an unplayed match where Brann has the home advantage, and that match can be deciding.

c)

```
# This code generates the design matrix, as well as computing the
# strength parameters with our own function.
library(myglm)
goals <- c(eliteserie$yh, eliteserie$ya)
X <- matrix(data = 0, nrow = 384, ncol = 17)
colnames(X) <- c("Intercept", "HomeAdvantage", unique(eliteserie$home)[-4])
for (i in 1:length(eliteserie$home)) {
    X[i, 1] = 1
    X[i, 2] = 1
    home_index <- which(colnames(X) == eliteserie$home[i])
    away_index <- which(colnames(X) == eliteserie$away[i])
    X[i, home_index] <- 1
    X[i, away_index] <- -1
}
for (i in 1:length(eliteserie$away)) {
    X[i + length(eliteserie$home), 1] = 1
    home_index <- which(colnames(X) == eliteserie$home[i])
    away_index <- which(colnames(X) == eliteserie$away[i])
    X[i + length(eliteserie$home), home_index] <- -1
    X[i + length(eliteserie$home), away_index] <- 1
```

```r
}
strength_param <- myglm(goals ~ -1 + X)
strength_param <- strength_param[order(-strength_param)]
names(strength_param) <- substring(names(strength_param), 2)
strength_param
```

```
##      HomeAdvantage          Rosenborg              Molde
##        0.402062206        0.367125310        0.279399199
##              Brann          Haugesund          Intercept
##        0.225775206        0.141301460        0.100321807
##                Odd        Sarpsborg08            Tromsoe
##        0.100120614        0.097677349        0.060581141
##      Stroemsgodset         Vaalerenga       Kristiansund
##        0.049792126        0.014730410        0.012552907
##         Ranheim_TF         Lillestroem            Stabaek
##        0.008502727       -0.132621109       -0.147940567
##              Start Sandefjord_Fotball
##       -0.225757649       -0.291683130
```

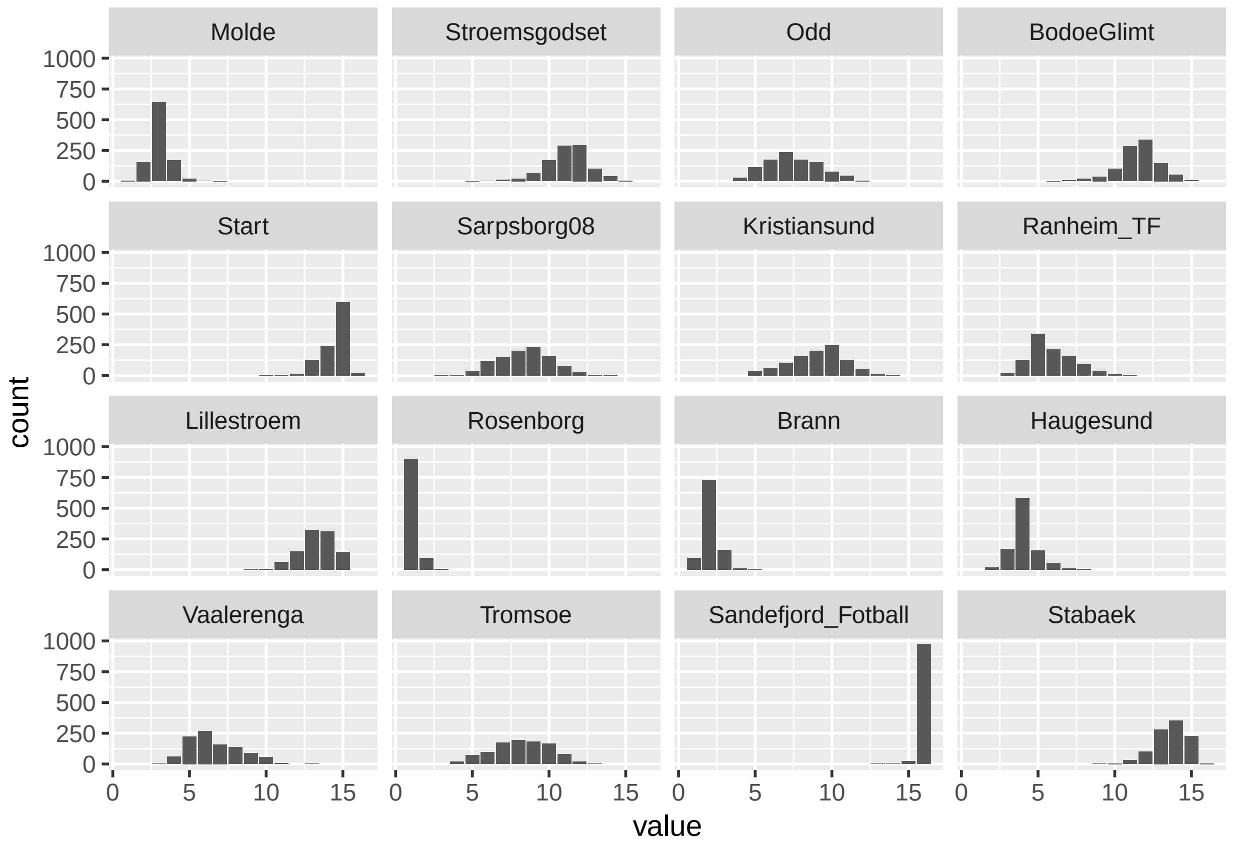
Here we see the estimated strength-parameters, with the stength parameter of BodoeGlimt set to zero. The expected number of goals for a home team A is thus $E[A] = exp(\beta_0 + \beta_{home} + \beta_A - \beta_B)$ and the away team B has the expected value $E[B] = exp(\beta_0 + \beta_B - \beta_A)$. We see that the intercept is some sort of "default" strength for BodoeGlimt, and if a BodoeGlimt play away against a team with higher strength parameter than the intercept, they would have a expected number of goals less than 1 (the sum of the betas less than zero). Rewriting as $E[A] = e^{\beta_0}e^{\beta_{home}}e^{\beta_A}/e^{\beta_B}$ we can more easily see that the expected number of goals for a team is e in the power of the intercept, times e in the power of the home parameter (if a home game for the team), times the e to the power of the strength parameter of the team, divided by e to the power of the strength parameter of the opponent. An interesting point is that Molde has a higher strength parameter than Brann, even though Brann has 5 more points so far in the season. That can be a result of Molde meeting better opponents than Brann so far, and the fact that Molde won both the matches between the teams, with 5-1 home and 0-4 away. Ranheim TF had a much lower strength parameter than their position in the table suggests. Looking at their played games, one can see that they are very unstable, loosing to teams that are considered worse, and winning against better teams. Thus it's hard to analyse their strength, and give it a number. The regression has probably considered some of their wins against better teams like Molde luck, which would explain the pessimistic strength estimate compared to their position at the table.

d)

```r
# This code simulates n seasons, and write the ranking for each
# season to a txt-file.
library("reshape2")
set.seed(42)
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2018h/unplayed2018"
eliteserie_unplayed <- read.table(file = filepath, header = TRUE, colClasses = c("character",
    "character"))
simulate_season_end <- function(data_unplayed_matches, strength_param) {
    for (i in 1:length(data_unplayed_matches$home)) {
        intercept <- strength_param["Intercept"]
        home_advantage <- strength_param["HomeAdvantage"]
        strength_hometeam <- strength_param[data_unplayed_matches$home[i]]
        strength_awayteam <- strength_param[data_unplayed_matches$away[i]]
        if (data_unplayed_matches$home[i] == "BodoeGlimt") {
            strength_hometeam <- 0
        } else if (data_unplayed_matches$away[i] == "BodoeGlimt") {
            strength_awayteam <- 0
        }
        data_unplayed_matches$yh[i] <- rpois(1, exp(strength_hometeam +
            intercept + home_advantage - strength_awayteam))
        data_unplayed_matches$ya[i] <- rpois(1, exp(strength_awayteam +
            intercept - strength_hometeam))
    }
    return(data_unplayed_matches)
}

Points <- c()
for (i in 1:1000) {
    simulated_results <- simulate_season_end(eliteserie_unplayed, strength_param)
    eliteserie_finished <- rbind(eliteserie, simulated_results)
    standings <- calculate_points(eliteserie_finished)
    write.table(standings, paste("Standings\\Standings_", toString(i),
        ".txt", sep = ""), sep = "\t", quote = FALSE)
}
```

```
# This code loads n season simulations, and plots the result
# distribution as bar diagram.
library("ggplot2")
library("reshape2")
n <- 1000
teams <- unique(eliteserie$home)
Positions <- matrix(0, nrow = length(teams), ncol = n)
Points <- matrix(0, nrow = length(teams), ncol = n)
rownames(Positions) <- c(teams)
rownames(Points) <- c(teams)
for (i in 1:n) {
    standings <- read.table(file = paste("Standings\\Standings_", toString(i),
        ".txt", sep = ""), header = TRUE, colClasses = c("numeric", "character",
        "numeric", "numeric", "numeric", "numeric"))
    Positions[, i] <- standings$Position
    Points[, i] <- standings$Points
}
Positions <- melt(Positions, id.vars = c("Teams"))
p <- ggplot(data = data.frame(Positions), aes(x = value)) + geom_bar() +
    facet_wrap(~Var1)
p
```



From this bar diagrams over the different simulations, we can see that Rosenborg wins around 900/1000 simulations, whereas Brann wins the remaining around 100 simulations. Molde wins about 2 or 3 of the simulations, but Rosenbord and Brann are mainly our contestants for the win. The omission of explanatory variables such as Rosenborg are playing Europe League in addition to Eliteserien, and their players will be more exhausted, can cause for a lower variability in the data than we see in our simulations. There are much more variability in the middel of the table, which suggests that the two top teams (Rosenborg and Brann) are a lot better than the others, and Sandefjord being the far worst (8 points behind the next team on the table so far). Another point to make is that this regression only set a general value on each teams strength, but following football results one knows that some teams perform better against other specific teams and vice versa. This regression only estimates the general strength of Rosenborg compared to all the other teams, but for a more realistic simulation one would have to estimate the strength difference between each team spesifically. And of course take in to account other factors as injuries and such.

```r
# This code computes mean and standard deviation of the simulations
result_statistics <- data.frame(Team = rownames(Points), Position = seq(1,
    length(Points[, 1]), by = 1), Mean = rowMeans(Points), row.names = NULL)

for (i in 1:nrow(Points)) {
    result_statistics$StdDev[i] <- sqrt(sum((Points[i, ] - result_statistics$Mean[[i]])^2/(length(Point
        ]) - 1)))
}
result_statistics <- result_statistics[order(-result_statistics$Mean),
    ]
result_statistics$Position <- seq(1, length(result_statistics$Team),
    by = 1)
result_statistics
```

```
##                   Team Position   Mean   StdDev
## 10           Rosenborg        1 64.588 2.845761
## 11               Brann        2 58.814 3.171666
## 1                Molde        3 53.887 3.136570
## 12           Haugesund        4 50.239 2.857980
## 8           Ranheim_TF        5 46.089 3.080677
## 13          Vaalerenga        6 44.477 3.050701
## 3                  Odd        7 42.338 2.972641
## 14             Tromsoe        8 41.268 3.022468
## 6          Sarpsborg08        9 40.696 2.966544
## 7         Kristiansund       10 39.994 3.098639
## 2        Stroemsgodset       11 35.304 3.106321
## 4           BodoeGlimt       12 34.802 3.030508
## 9          Lillestroem       13 30.629 2.951961
## 16             Stabaek       14 29.785 3.061396
## 5                Start       15 27.855 2.703939
## 15 Sandefjord_Fotball       16 19.565 2.637941
```

Looking at the means, there is a considerable difference in points between Rosenborg and Brann, at 5.774 points. However the standard deviations for the teams will suggest that if Rosenborg being close to the lower limit and Brann close to the upper limit, would give Brann more points than Rosenborg. If one also take into acount that they are playing against each other in the final part of the season, not only giving the winner three points, but also ensures that the looser gets zero points in that game, we have plausible scenarios where Brann wins Eliteserien. As shown in our simulations, they only win around 100/1000, but the match between the two teams will be crucial in the first place battle.

```r
# Code from myglm-package
myglm <- function(formula, data = list(), 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")

    # Add code here to calculate coefficients, residuals, fitted values,
    # etc...  and store the results in the list est
    est <- list(terms = terms, model = mf)

    par <- rep(0, ncol(X))
    betahat <- optim(par = par, fn = loglik_poi, x = X, y = y, method = "BFGS")$par
    names(betahat) = colnames(X)
    est$beta <- betahat

    # Store call and formula used
    est$call <- match.call()
    est$formula <- formula

    # Set class name. This is very important!
    class(est) <- "myglm"
    # Return the object with all results
    return(est$beta)
}

loglik_poi <- function(par, x, y) {
    beta <- par
    mu <- beta %*% t(x)
    LL <- sum(y * mu - exp(mu))
    return(-LL)
}
```