TMA4315: Compulsory exercise 2 (Logistic regression & Poisson regression)

Group 6: Marcus Hjørund, Gard Gravdal

03.11.2023

Contents


```
# Libraries used in the project
library(ggplot2)
library(dplyr)
library(reshape2)
library(gmodels)
library(myglm)
```

Part 1

In the first part of the project we will look at the list of the 118 highest mountains in the world, along with some properties of the mountain, such as the number of succsessful and failed attempts to reach the summit as of 2004. We will see that this can be modeled as a binary regression problem with the logit link function to model the probability of succsess. We will consider a data set consisting of the heights (in meter), topographic prominence (in meters) in addition to the number of failed/succsessful attempts for 113 of the mountains.

We let y_i be the number of succsessful ascents and n_i be the number of total attempts. We then use a binary regression with logit link to model the probability of succsess:

$Y_i = \text{Bin}(n_i, \pi_i)$, $\eta_i = x_i^T \beta$ and $\log(\text{odds}) = \log(\frac{\pi_i}{1-\pi_i}) = \eta_i$ where $x_i \in \mathbb{R}^p$ is the vector of covariates for observation i, β is the vector of regression parameters and π_i is the probability of succsess for observation i.

a) log-likelihood and maximum likelihood

The likelihood function $L(\beta)$ is used to find the maximum likelihood estimate of parameters by maximizing the function. The log-likelihood function is the log of $L(\beta)$. Since the log-function is monotone, the log-likelihood function will have the same maximum. The log-likelihood function $\ell(\beta)$ for the model (individual, not grouped) is given as

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

To find the maximum likelihood estimates for the parameter β, we look at the values of β that maximizes the log-likelihood function $\ell(\beta)$. In few words, we make a key assumption that the maximum likelihood estimator $\hat{\beta}$ can be approximated by $\hat{\beta} \approx N_p(\beta, F^{-1}(\hat{\beta}))$ as $n \to \infty$, where $F(\beta) = \sum_{j=1}^{n} x_j x_j^T \pi_j (1 - \pi_j)$ is the Expected Fisher information matrix (individual). To find $\hat{\beta}$, we must typically use optimization techniques such as Newton-Raphson or Fisher scoring (which are the same in this case), solving iteratively

$$\beta^{(t+1)} = \beta^{(t)} + F^{-1}(\beta^{(t)}) s(\beta^{(t)})$$

until convergence, where $s(\beta^{(t)})$ is the score function equal to $s(\beta) = \sum_{i=1}^{n} x_i (y_i - \pi_i)$ in this case.

b) Modelling probability of success, confidence intervals

In the next section we will use the data set to fit the model explained in the introduction with *prominence* and *height* as predictors. In addition we will interpret the model and discuss the significance. Further, we will make a 95% confidence interval for the *height* $(\hat{\beta_L}, \hat{\beta_H})$ and calculate the confidence interval $(\exp(\hat{\beta_L}), \exp(\hat{\beta_H}))$.

We load the dataset.

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

We fit the model with height and prominence as predictors.

```
model1 <- glm(cbind(success, fail) ~ height + prominence, data = mount,
    family = "binomial")
model1
```

```
##
## Call:  glm(formula = cbind(success, fail) ~ height + prominence, family = "binomial",
##     data = mount)
##
## Coefficients:
## (Intercept)       height   prominence
##   13.685845    -0.001635    -0.000174
##
## Degrees of Freedom: 112 Total (i.e. Null);  110 Residual
## Null Deviance:       715.3
## Residual Deviance: 414.7     AIC: 686
```

Next, we look to interpret the model. We start by using the summary-function on our model.

```
summary1 <- summary(model1)
summary1
```

```
##
## 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
```

To interpret the model, we look at the coefficients of the logistic regression model. Each coefficient corresponds to a predictor variable and measures the change in log-odds of the response (success/failure of ascension in this case) for a one-unit change in predictor variable, holding all other predictor variables constant. The coefficient for the intercept provides the log-odds of success for the response when all other predictor variables are zero (or relative to a reference level, not the case here).

We interpret the coefficients by calculating the odds-ratio from the formula Odds $= e^{\beta_i}$ for coefficient i. The odds ratio represents how a one-unit change in predictor variable β_i affects the odds of the outcome variable. We see that the coefficient for the predictor *height* is -1.635e-03, the coefficient for prominence is -1.749e-04 and 1.369e+01 for the coefficient of the intercept. We then calculate the different odds:

```
OddsIntercept1 <- exp(summary1$coefficients[1])
OddsHeight1 <- exp(summary1$coefficients[2])
OddsProminence1 <- exp(summary1$coefficients[3])
cat("The odds for one-unit change intercept: \n", OddsIntercept1, "\n")
```

```
## The odds for one-unit change intercept:
##  878389.2
```

```
cat("The odds for one-unit change Height: \n", OddsHeight1, "\n")
```

```
## The odds for one-unit change Height:
##  0.9983659
```

```
cat("The odds for one-unit change Prominence: \n", OddsProminence1, "\n")
```

```
## The odds for one-unit change Prominence:
##  0.999826
```

We expect the odds ratio of the intercept to be very large (tending towards infinite) in this case, as it represents the odds of successful ascension when height and prominence is zero. We see that this is the case for our model with the odds for the coefficient of intercept at 878389.2.

The values of the odds ratios for height and prominence are 0.9983659 and 0.999826 respectively. An odds ratio greater than 1 (meaning positive coefficient) indicates that an increase in the predictor variable is associated with higher odds of the event. Likewise if the odds ratio is less than 1 (coefficient negative), the predictor variable is associated with lower odds of the event. In this case we see that both the coefficients of the predictor variables are small negative numbers and thus have odds ratios slightly smaller than 1. This is interpreted as when we change the height of the mountain by 1 meter, the odds of success changes by 0.9983659 in ratio. As a concrete example, we have the relationship "odds success 10m height" = "odds of success 9m height" times the odd ratio 0.9983659. Similarly a change in 1 meter prominence changes the odds of success by 0.999826 in ratio.

Further, we want to test the significance of the coefficients. The significance can be tested using a *wald-test*. For hypothesis $H_0 : \mathbf{C}\beta = \mathbf{d}$ vs. $\mathbf{C}\beta \neq \mathbf{d}$, the Wald-test is given as

$$w = (\mathbf{C}\hat{\beta} - \mathbf{d})^T [\mathbf{C}F^{-1}(\hat{\beta})\mathbf{C}^T]^{-1}(\mathbf{C}\hat{\beta} - \mathbf{d})$$

which in the case of one regression parameter simplifies to the hypothesis $H_0 : \beta_k = 0$ vs. $\beta_k \neq 0$ and

$$w = \left(\frac{\hat{\beta}_k - \beta_k}{\sqrt{a_{kk}(\hat{\beta})}} \right)^2 = Z_k^2 \sim \chi_1^2$$

where $a_{kk}(\hat{\beta})$ is diagonal element number k in the matrix $F^{-1}(\hat{\beta}) = Cov(\hat{\beta})$. In the code below we perform a significance test for the coefficients.

```
Cov.beta_h <- summary1$cov.unscaled
w1 = (model1$coefficients[1]/Cov.beta_h[1])^2
w2 = (model1$coefficients[2]/Cov.beta_h[2])^2
w3 = (model1$coefficients[3]/Cov.beta_h[3])^2
w = c(w1, w2, w3)
Pr_chi = pchisq(w, df = 1, lower.tail = FALSE)
cat("Coefficients:\n", names(model1$coefficients), "\n p-values from the chi^2-test:\n",
    Pr_chi)
```

```
## Coefficients:
##  (Intercept) height prominence
##  p-values from the chi^2-test:
##  1.259343e-33 1.621647e-27 2.298112e-15
```

From the result of the χ^2-test, we see that all the coefficients are significant given that all the p-values are close to zero, meaning we reject the null hypothesis.

Next we want to find a confidence interval for the coefficient of the covariate height. We know that in the asymptotic case, we have

$$Z_k = \frac{\hat{\beta}_k - \beta_k}{\sqrt{a_{kk}(\hat{\beta})}} \sim N(0,1)$$

We use this to make a confidence interval. From $P(-z_{\alpha/2} \leq Z_k \leq z_{\alpha/2}) = 1 - \alpha$ we can solve for $\hat{\beta}_k$ and obtain

$$P(\hat{\beta}_k - z_{\alpha/2}\sqrt{a_{kk}(\hat{\beta})} \leq \beta_k \leq \hat{\beta}_k + z_{\alpha/2}\sqrt{a_{kk}(\hat{\beta})} = 1 - \alpha$$

which gives us the confidence interval for β. In the next code section we calculate the 95% confidence interval (i.e. $\alpha = 0.05$).

```
# Confidence interval for covariate height
beta.height.L <- model1$coefficients[2] - qnorm(0.975) * sqrt(Cov.beta_h[2,
    2])
beta.height.H <- model1$coefficients[2] + qnorm(0.975) * sqrt(Cov.beta_h[2,
    2])
beta.height.CI <- c(beta.height.L, beta.height.H)
beta.height.CI
```

```
##       height       height
## -0.001913631 -0.001357205
```

```
cat("95% Confidence interval for covariate height: \n", beta.height.CI)
```

```
## 95% Confidence interval for covariate height:
##  -0.001913631 -0.001357205
```

From the confidence interval of height it can be interesting to look at the 95% confidence interval of the odds ratio for height, $(e^{\beta_L}, e^{\beta_H})$. This can be interpreted as the interval in which we can say with 95% confidence that the odds ratio for height is according to the model.

```
OddsRatio.H.CI <- c(exp(beta.height.L), exp(beta.height.H))
cat("95% Confidence interval for odds ratio of height: \n", OddsRatio.H.CI,
    "\n")
```

```
## 95% Confidence interval for odds ratio of height:
##  0.9980882 0.9986437
```

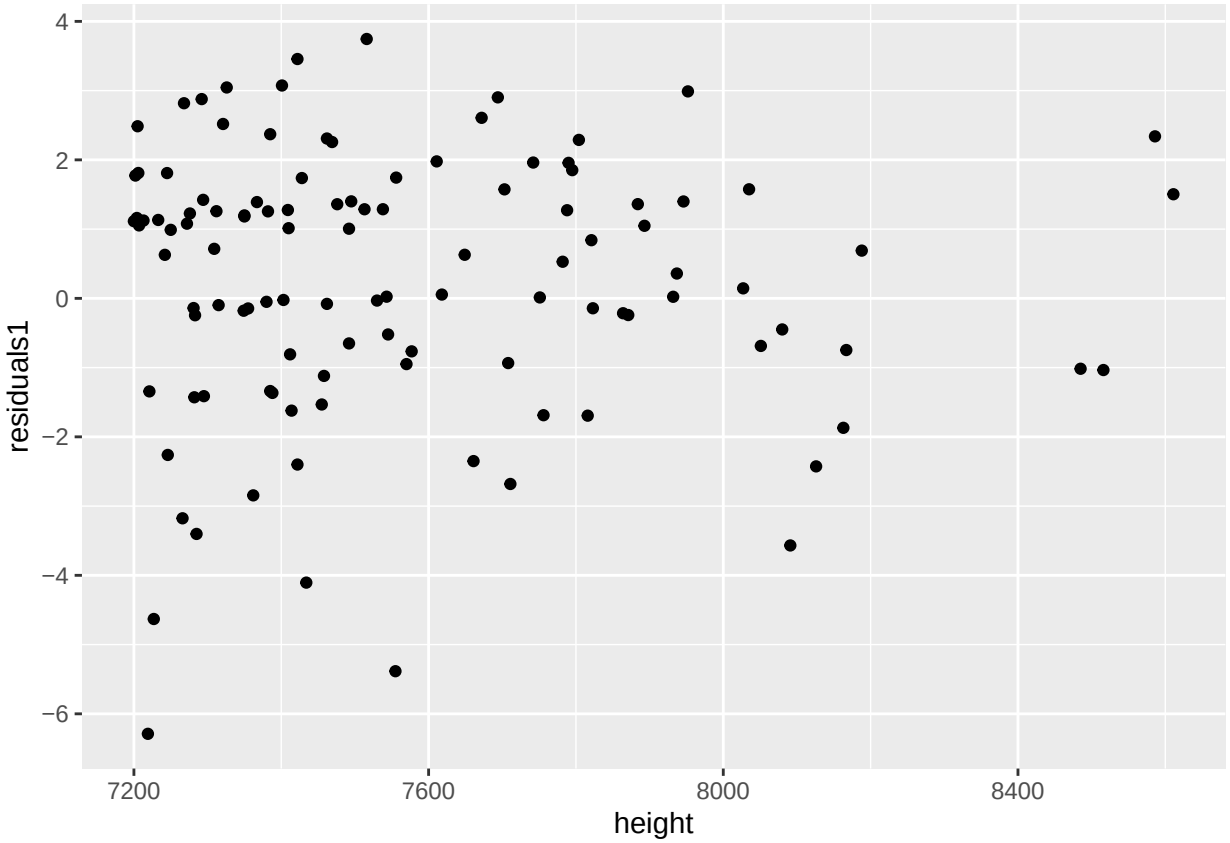
We see that the confidence interval for the odds ratio is given as (0.9980882, 0.9986437) for height. We can say that with 95% certainty that changing the height by 1 meter will change the odds by a ratio of somewhere in this interval according to the model.

c) Deviance, Probability(Height, Prominence)

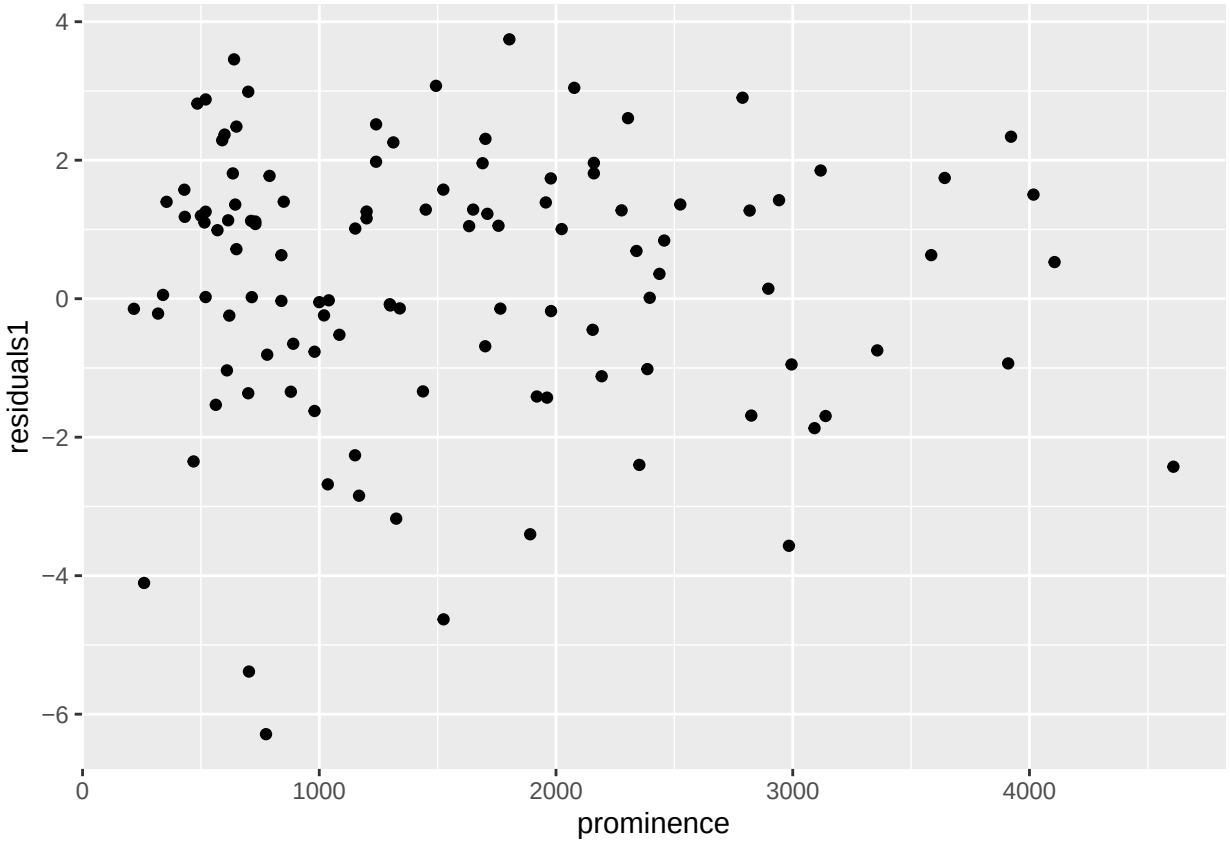
In the next section we will plot the deviance residuals as a function of the covariate for each covariate and interpret the result. After that we will use the model deviance to assess the fit. At the end of the section we will plot the estimated probabilities as a function of the covariates *height* and *prominence*.

The code below plots deviance residuals as a function of the covariates.

```
residuals1 <- residuals(model1, type = "deviance")
ggplot(mount, aes(x = height, y = residuals1)) + geom_point()
```




```
ggplot(mount, aes(x = prominence, y = residuals1)) + geom_point()
```



Before we start we should note that residual plots for logistic regression (and for GLM in general) is highly debated, and we should arguably not put much emphasis on the result.

Having said that, here is how we can interpret the plot. We know that the deviance residuals are a measure of the discrepancy between the observed response and the response predicted by our model. Positive deviance residuals suggests that the model underestimates the probability of success for a given observation. In other words, the model predicts a lower probability of an event occurring than what is observed. It can indicate that the model is too conservative or that there are unmodeled factors contributing to the response. Negative values indicate the opposite. The model then predicts a higher probability than what is observed. Near zero deviance residuals indicate that the model is close in agreement with the observed data.

Looking at both plots, it seems that the deviance residuals have an approximate mean of zero for all values of the covariate. There is no clear pattern in deviance residual value and covariate value for either covariate.

In the next section we use the model deviance to assess the fit of the model. The deviance is used to assess model fit and choice, and is based on the likelihood ratio test. The method is defined by putting the *saturated model* in place of the large model in the ratio test, while our candidate model is in place of the smaller model. The saturated model is an "imaginary model", where if we were to provide a perfect fit for our data, we could estimate each π_j by the observed frequency for each group. The saturated model would be the model where each group was given its own parameter. The interpretation of the deviance is that a small deviance indicates a good model, since then our candidate model is close to the saturated model with distance measured in deviance.

We investigate the (residual) deviance as well as the null deviance in the code below. The null deviance is the deviance of the model with no covariates and only the intercept present. If the difference between the null deviance and the residual deviance is large, it suggests that our model is good since our model explains more of the variance in the response than the null model and that the covariates are correlated with the response.

We want to investigate if the candidate model is equally as good as the saturated model. We use the fact

that the deviance is distributed as χ^2 with $G - p$ degrees of freedom, where G are the groups. We perform the hypothesis test below with the null hypothesis being that the candidate model is equally as good as the saturated model.

```
# Deviance of the model
D = deviance(model1)
NullD = summary1$null.deviance
cat("Residual deviance for our model:\n", D, "\n")
```

```
## Residual deviance for our model:
##  414.6842
```

```
cat("Null deviance for our model:\n", NullD, "\n")
```

```
## Null deviance for our model:
##  715.2889
```

```
# Testing significance of model using chi-squared test
df.residual1 = summary1$df.residual
pchisq(D, df.residual1, lower.tail = FALSE)
```

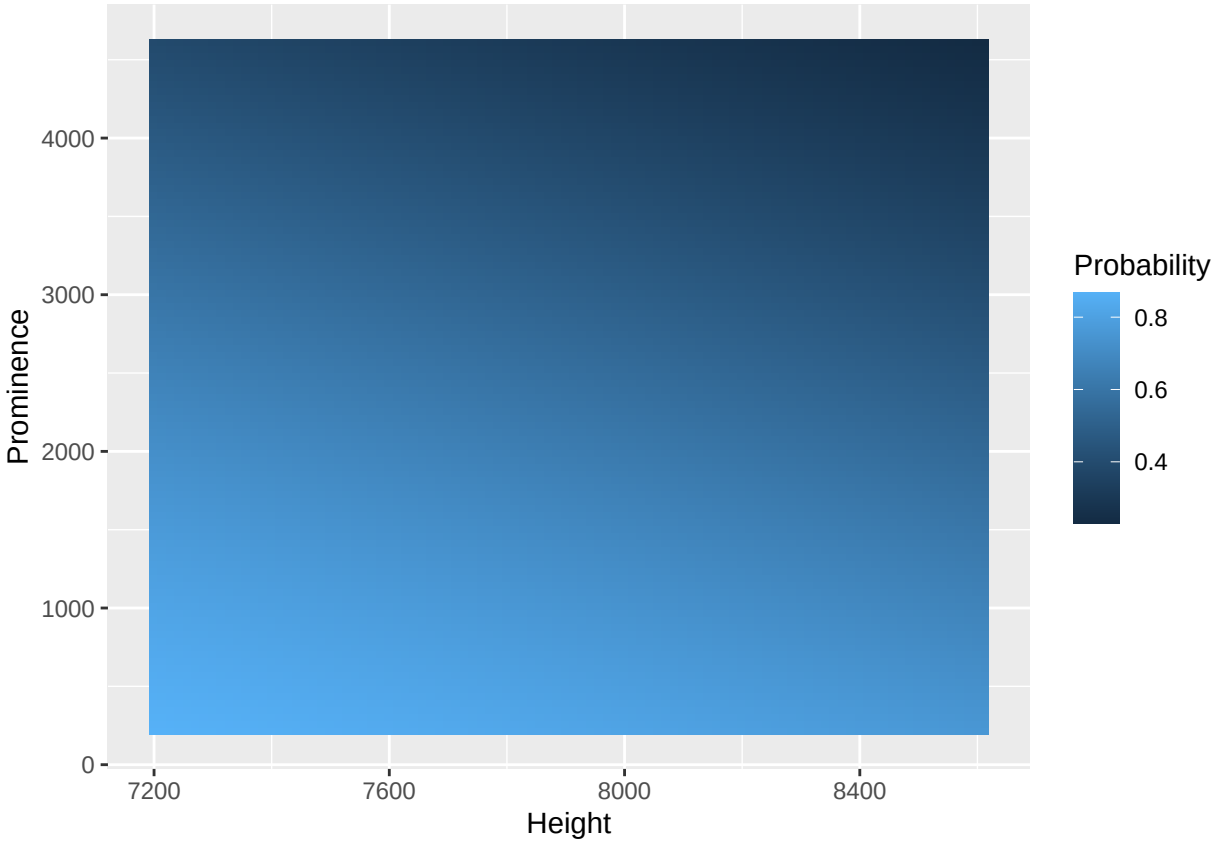
```
## [1] 6.61187e-37
```

We see that there is a relatively small difference between the null deviance and the residual deviance. This indicates that our model explains slightly more of the variance in the response than the null model, and that our covariates don't explain a lot of the variance in the response success/failure to ascend mountain.

We see that the hypothesis test is significant for our model as the p-value for the χ^2 test is very close to zero, meaning we reject the null hypothesis that our model is a good fit. Overall it seems like our model is a poor fit.

In the next section we plot the estimated probabilities as a function of both height and prominence.

```
datapoints <- 100
colHeight <- seq(min(mount$height), max(mount$height), length.out = datapoints)
colProm <- seq(min(mount$prominence), max(mount$prominence), length.out = datapoints)
prob_matrix <- matrix(1:10000, ncol = 100, nrow = 100)
prob_frame <- data.frame()
for (i in 1:datapoints) {
    for (j in 1:datapoints) {
        eta = c(1, colHeight[i], colProm[j]) %*% model1$coefficients
        prob_matrix[i, j] = plogis(eta)
        prob_frame <- rbind(prob_frame, c(colHeight[j], colProm[i], prob_matrix[i,
            j]))
    }
}
colnames(prob_frame) <- c("Height", "Prominence", "Probability")
ggplot(data = prob_frame, mapping = aes(x = Height, y = Prominence, z = Probability)) +
    geom_raster(aes(fill = Probability))
```



In the plot we see that the x-axis is Height and is scaled from about 7200 meters to about 8700 meters. The y-axis is Prominence and is scaled from 0 meters to about 4500 meters. From the plot we see that the bottom left corner at [7200,0], the probability is close to 1, while the probability is close to 0 at [8700,4500] in the top right corner. We see a pattern of decreasing probability of success as we move increasingly along the x- and y-axis, as we would expect intuitively since the difficulty presumably increases when the height and prominence increase. The plot gives an impression of a relatively simple relationship between the covariates and the probability of successfully ascending.

d) Concrete examples, Mount Everest and Chogolisa

Next we will find the estimated probability of successfully ascending Mount Everest according to our model. Mount Everest has both height and prominence equal to 8848 meters. Using the linear predictor $\eta = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$ and plugging in the values $x_1 = x_2 = 8848$, we find the log-odds by $\log(\frac{\pi}{1-\pi}) = \eta$. From this we find the probability of climbing Mount Everest by $\pi = \frac{e^\eta}{1+e^\eta}$. The inbuilt function *plogis()* calculates the final expression of π as a function of η.

```
# Probability of climbing Mount Everest according to model
x.ME = c(1, 8848, 8848)
eta.ME = x.ME %*% model1$coefficients
pi.ME = plogis(eta.ME)[1]
cat("Probability climbing Mount Everest successfully according to model:\n",
    pi.ME)
```

```
## Probability climbing Mount Everest successfully according to model:
##  0.08917783
```

We see that the probability of successfully climbing Mount Everest is approximately 8.9% according to our model.

Further, we would like to make an approximate confidence interval for the probability of climbing Mount Everest. To do this, we make use of the asymptotic distribution of the maximum likelihood estimator $\hat{\beta}$ to find the asymptotic distribution of the linear predictor $\hat{\eta}$. From before, we know that as $n \to \infty$, $\hat{\beta} \approx N_p(\beta, F^{-1}(\hat{\beta}))$, $Cov(\hat{\beta}) = F^{-1}(\hat{\beta})$ in the asymptotic case. Thus we can assume $\hat{\eta} \approx N$, since it is a linear combination of normally distributed variables. We find that $E[\hat{\eta}] = E[x^T\hat{\beta}] = x^T\beta$. Further, we find that $Var[\hat{\eta}] = x^T Cov(\hat{\beta})x = x^T F^{-1}(\hat{\beta})x$. Thus we have $\hat{\eta} \sim N(x^T\beta, x^T F^{-1}(\hat{\beta})x)$, and we can write

$$Z = \frac{x^T\hat{\beta} - x^T\beta}{(x^T F^{-1}(\hat{\beta})x)^{\frac{1}{2}}} \sim N(0,1)$$

We make use of this to construct a confidence interval for η with a z-test. From $P(-z_{\alpha/2} \leq Z \leq z_{\alpha/2}) = 1-\alpha$, we can solve for $\eta = x^T\beta$ and we find that $CI(\eta) = x^T\hat{\beta} \pm z_{\alpha/2}(x^T F^{-1}(\hat{\beta})x)^{1/2}$. Then the confidence interval for the probability is (π_L, π_H), where $\pi_k = \frac{e^{\eta_k}}{1+e^{\eta_k}}$, k = L,H. The result for Mount Everest is calculated in the code below.

```
# Confidence interval eta Mount Everest
SE.eta.ME <- sqrt(t(x.ME) %*% Cov.beta_h %*% x.ME)
eta.ME.L <- eta.ME - qnorm(0.975) * SE.eta.ME
eta.ME.H <- eta.ME + qnorm(0.975) * SE.eta.ME

# Confidence interval probability Mount Everest
pi.ME.CI <- c(plogis(eta.ME.L), plogis(eta.ME.H))
cat("Confidence interval for the probability of succsessfully
    climbing Mount Everest",
    pi.ME, "is:\n", pi.ME.CI)
```

```
## Confidence interval for the probability of succsessfully
##     climbing Mount Everest 0.08917783 is:
##  0.05486572 0.1417303
```

We might question if it is reasonable to use this model to estimate the probability of ascending Mount Everest. The result from the analysis of the deviance suggested that our model is poor since it did not pass the hypothesis test and in addition didn't have a relatively big difference between null deviance and residual deviance. The plot of probability as a function of height and prominence also seemed to show a very simplified relationship between the probability of successfully ascending and height/prominence. Intuitively, the result of about 8.9% probability of successfully ascending Mount Everest seems low. We can also argue intuitively that the difficulty of climbing a mountain should contain other factors than only overall height and prominence, such as type of terrain, average experience of climbers per mountain etc. Overall, it seems like our model does not capture a lot of the variability of the response.

Now we want to do the same analysis for the mountain Chogolisa, which stands at 7665m height and with prominence 1624m. The data for Chogolisa gave 2 failures and 20 successes.

```
# Same procedure for Chogolisa
x.CH = c(1, 7665, 1624)
eta.CH = x.CH %*% model1$coefficients
pi.CH = plogis(eta.CH)[1]
cat("Probability climbing Chogolisa successfully according to model:\n",
    pi.CH)
```

```
## Probability climbing Chogolisa successfully according to model:
##  0.7042924
```

```
# Confidence interval eta Chogolisa
SE.eta.CH <- sqrt(t(x.CH) %*% Cov.beta_h %*% x.CH)
eta.CH.L <- eta.CH - qnorm(0.975) * SE.eta.CH
eta.CH.H <- eta.CH + qnorm(0.975) * SE.eta.CH

# Confidence interval probability Chogolisa
pi.CH.CI <- c(plogis(eta.CH.L), plogis(eta.CH.H))
cat("Confidence interval for the probability of succsessfully
    climbing Chogolisa",
    pi.CH, "is:\n", pi.CH.CI)
```

```
## Confidence interval for the probability of succsessfully
##     climbing Chogolisa 0.7042924 is:
##  0.6832255 0.7245232
```

We see from the result that the probability of successfully climbing Chogolisa is 0.7042924 with 95% confidence interval of (0.6832255, 0.7245232) according to the model. Looking at the data, we see that the approximated probability of success is 20/22 or approximately 0.909 with a sample size of 22. This probability is clearly far outside our confidence interval, which suggests that our model is poor. This supports our previous assumption that the model is not reasonable.

Part 2

In the second part of the project we will look at a data set consisting of the games played in Eliteserien so far this season. So far the teams have played between 22 and 23 matches each, and in total 179 matches has been played. We will look to implement a model of the league by *Poisson regression*, where the teams *strengths* are the regression parameters we seek to estimate. The league consists of $n = 16$ teams, and there will be a total of $n(n-1) = 240$ games played at the end of the season. We will look to predict the winner of the season based on the scores so far in the season, as well as study the uncertainty of the ranking in the season so far. Finally, we want to know: How likely/unlikely are Rosenborg to become champions?

We model the games in the following way: We assume number of goals scored between home and away teams, *score*, are independent of each other in any given game. In the model we assume that each team has a single parameter that measures its *strength*, and it is denoted as β_{team} for a given team. We also assume that each game has (the same) home advantage, denoted by the home advantage parameter β_{home}. We then model the score (number of goals) for the home team with a Poisson distribution with mean λ, where

$$ln\lambda = \beta_0 + \beta_{home} + \beta_{HomeTeam} - \beta_{AwayTeam}$$

Similarly we model the score for the away team with a Poisson distribution with mean λ, and

$$ln\lambda = \beta_0 - \beta_{HomeTeam} + \beta_{AwayTeam}$$

In summary, the goals scored for the home team and the away team depends on the relative strength between the teams in addition to a home advantage for the team playing home. For two teams of equal strength, the expected score of the home team is $exp(\beta_0 + \beta_{home})$ and the expected score of the away team is $\exp(\beta_0)$.

a) Assumption of independence home/away goals

In order to do the Poisson regression for the Eliteserie data set, we assume that the scores of the home and away teams are independent, such that the distributions remain unchanged despite the home and away teams scoring goals. In this section we will check if this assumption is valid. To check whether this assumption is valid we will create a contingency table and test for independence using *Pearson's* χ^2 test.

The Pearson χ^2-goodness of fit statistic is given as

$$X_P^2 = \sum_{j=1}^{G} r_j^2 = \sum_{j=1}^{G} \frac{(y_j - n_j\hat{\pi_j})^2}{n_j\hat{\pi_j}(1-\hat{\pi_j})}$$

We take advantage of the fact that the Pearson χ^2 test is asymptotically equivalent to the deviance statistic, and thus from before we know that this is asymptotically χ^2_{G-p}. We test under the null hypothesis that the home score and away score are independent.

Table 1: Home vs. Away scores

	0	1	2	3	4	5
0	12	13	9	4	2	0
1	15	15	12	7	0	0
2	9	15	6	5	0	1
3	7	13	9	0	2	0
4	6	3	2	2	1	0
5	1	5	1	0	0	0
6	0	1	0	0	0	0
7	0	0	0	1	0	0

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

From the Pearson's χ^2 test we get a p-value of 0.5393. We find that we do not reject the null hypothesis for any reasonable choices of α such as $\alpha = 0.05$ since $p >> \alpha$. Thus we find that the Pearson's χ^2 suggests that the assumption of independence between home and away scores is valid.

b) Preliminary results

In the next section we set up the preliminary ranking of the season so far (as of October 1st). The ranking is decided by points. The winner of a game gets 3 points, a draw gives each time 1 point and a lost game gives 0 points. For two teams of equal points, the goal difference (total goals scored vs total goals conceded) between the teams decides, and the team with the highest goal difference is placed higher.

```
playedGames <- na.omit(eliteserie)
teamNames <- c(unique(eliteserie$home))
teamPoints <- matrix(rep(0, 32), nrow = 16, ncol = 2)
colnames(teamPoints) <- c("Scores", "GoalDifferences")
rownames(teamPoints) <- teamNames
for (i in 1:16) {
    dfHome <- subset(playedGames, home == teamNames[i])
```

```
    dfAway <- subset(playedGames, away == teamNames[i])
    # for loop for home data
    for (j in 1:length(dfHome$X)) {
        teamPoints[i, 2] <- teamPoints[i, 2] + (dfHome$yh[j] - dfHome$ya[j])
        if (dfHome$yh[j] > dfHome$ya[j]) {
            teamPoints[i, 1] <- teamPoints[i, 1] + 3
        } else if (dfHome$yh[j] == dfHome$ya[j]) {
            teamPoints[i, 1] <- teamPoints[i, 1] + 1
        }
    }
    # new for loop for away data
    for (j in 1:length(dfAway$X)) {
        teamPoints[i, 2] <- teamPoints[i, 2] + (dfAway$ya[j] - dfAway$yh[j])
        if (dfAway$ya[j] > dfAway$yh[j]) {
            teamPoints[i, 1] <- teamPoints[i, 1] + 3
        } else if (dfAway$ya[j] == dfAway$yh[j]) {
            teamPoints[i, 1] <- teamPoints[i, 1] + 1
        }
    }
}
teamPoints <- as.data.frame(teamPoints)
cat("Ranking of teams in Eliteserien as of 1. October:\n")
```

```
## Ranking of teams in Eliteserien as of 1. October:
```

```
teamPoints <- teamPoints %>%
    arrange(desc(Scores), desc(GoalDifferences))
teamPoints
```

```
##                     Scores GoalDifferences
## 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
```

The ranking of the teams as of 1. October given. *Scores* in the table should be read as amount of points earned. We see that Viking is first with 51 points, while Aalesund is last with 12 points. Unfortunately for Rosenborg supporters, we can already tell that a first place finish is unlikely regardless of model, given that they are already 22 points behind first place with 21/24 points left to play for (7/8 games remaining for each team, given that a full season is 30 games).

c) Poisson, modelling parameter strengths

In the next section we look to use Poisson regression to estimate the intercept β_0 and the home advantage parameter β_{home} as well as the strength parameters β_{team} for each team. We will produce a ranking based on estimated strength and compare it with the ranking from the previous section.

In the implementation we force the parameter $\beta_{\text{Sarpsborg08}}$ to be 0, thus effectively reducing the number of team covariates from 16 to 15. In practice, the coefficient $\beta_{\text{Sarpsborg08}}$ serves as a reference parameter. Stronger teams have a positive strength parameter while weaker teams have a negative strength parameter. Sarpsborg08 was chosen arbitrarily.

```
set.seed(42)
goals <- c(playedGames[, "yh"], playedGames[, "ya"])
X <- matrix(rep(0, length(goals) * 18), ncol = 18, nrow = 358)
colnames(X) <- c("Intercept", "HomeAdvantage", teamNames)
X[, "Intercept"] <- rep(1, 358)
j <- 1
for (i in 1:358) {
    # first 179 scores are for the home team
    if (i <= 179) {
        # all home teams receive the home team bonus
        X[, "HomeAdvantage"][i] <- 1
        X[, playedGames$home[i]][i] <- 1
        X[, playedGames$away[i]][i] <- -1
    } else {
        # using j as we start anew with the playedGames data for
        # the away matches
        X[, playedGames$away[j]][i] <- 1
        X[, playedGames$home[j]][i] <- -1
        j <- j + 1
    }
}
# Using Sarpsborg08 as our reference
colIdx <- which(colnames(X) == "Sarpsborg 08")
X_without_reference <- subset(X, select = -c(colIdx))
model1 <- myglm(goals ~ -1 + X_without_reference)
model2 <- glm(goals ~ -1 + X_without_reference, family = "poisson")
estimates <- model1$estimate
estimateMatrix <- matrix(estimates, nrow = 1, ncol = length(estimates))
colnames(estimateMatrix) <- colnames(X_without_reference)
rownames(estimateMatrix) <- c("estimates")
estimateMatrix <- estimateMatrix[, order(-estimates)]
cat("Estimated strengths of coefficients:\n")
```

```
## Estimated strengths of coefficients:
```

```
estimateMatrix
```

```
##      HomeAdvantage         Bodø/Glimt              Molde             Viking
##         0.35856043         0.34370760         0.24047170         0.22088994
##          Intercept              Brann             Tromsø         Lillestrøm
##         0.16645289         0.09540541         0.08345698         0.01500923
##                Odd       Strømsgodset Sandefjord Fotball          Rosenborg
```

```
##       -0.09909623       -0.13540444       -0.16520213       -0.16666624
##         Vålerenga         Haugesund             Stabæk            HamKam
##       -0.17662506       -0.22851737       -0.28923247       -0.29285376
##         Aalesund
##       -0.57561495
```

Before comparing the official ranking with the ranking in strength for our model, we need to keep several things in mind. The comparison might not be completely fair, since some teams might have easier games remaining compared to other teams "on paper". As an example, maybe Bodø/Glimt have been "unlucky" in the model in case they have played the leaders Viking twice already, but have yet to play the assumed weakest team Aalesund twice yet. Their strength parameter would likely be higher than of the opposite case since they are more likely to win/score more goals against the weaker team. This could be something interesting to keep in mind for the final part of the project where we will simulate the remainder of the season.

When we compare the strength parameter to the official ranking as of October 1st, we see several interesting things. In general, the ranking based on strength agrees well with the current preliminary rankings, but with some deviations. 7 of the 16 teams are placed in the same position in both rankings, while 9 of the teams differ. For the teams that the two rankings disagree on, we see that they are still close in proximity. The biggest difference is for HamKam, which the model predicts a to be the 15th strongest team (i.e. the second weakest team), while the official rankings have them in 11th place. A reason for the difference could be that HamKam had some big losses, or that their won games are won by few goals on average. This is also reflected in the goal difference, where we can see that HamKam at -15 is quite a bit worse than their rivaling teams close on the table. Only Stabæk (-16, 15th in official ranking) and Aalesund (-36, 16th in the official ranking) have a worse goal difference. An interesting observation is that Bodø/Glimt is the highest ranked team in strength by quite a margin with coefficient value of 0.34388392.

Interestingly, the home advantage parameter at 0.35851491 is relatively large compared to the team covariates and the intercept. If we look at the case where a draw is most likely, i.e. when the expected goals for the home team is equal to the expected goals for the away team, we find that $\frac{1}{2}\beta_{home} = \beta_{AwayTeam} - \beta_{HomeTeam}$. This means that when the difference in strength between the away team and the home team (away team stronger) is close to 0.1792575, a draw is more likely. According to the model, an example close to this is if Sarpsborg08 (assumed 7th strongest team) played Vålerenga (12th strongest) at Vålerengas home stadium. The strength of Sarpsborg08 is neutralized by the home advantage of Vålerenga according to the model, giving close to a completely even game.

We see that the intercept of the model is 0.16638409. We can use this to calculate the expected number of goals for the home and away team if two teams of equal strength play according to the model. Expected goals for home team is given by $exp(\beta_0 + \beta_{home}x_{home})$ and the expected number of away goals is given by $exp(\beta_0)$. We calculate the result below.

```
beta_null <- estimateMatrix["Intercept"][[1]]
beta_home <- estimateMatrix["HomeAdvantage"][[1]]
xG_Home <- exp(beta_null + beta_home)[[1]]
xG_Away <- exp(beta_null)[[1]]
cat("Expected number of goals for team playing at home:\n", xG_Home,
    "\n")
```

```
## Expected number of goals for team playing at home:
##  1.690481
```

```
cat("Expected number of goals for team playing away:\n", xG_Away, "\n")
```

```
## Expected number of goals for team playing away:
##  1.181108
```

We see that the expected number of goals for the home team for two teams of equal strengths is equal to 1.690288 while the away team is expected to score 1.181027 goals. Intuitively we can interpret this as the home team having a higher chance of winning the game, but drawing chances are definitely there also for a team playing away. Away team winning should be relatively rare, but might happen from time to time.

d) Simulating results and analysing model

In this final section of the project, we look to investigate the rankings predicted by our model when we simulate the remaining 61 games. We use the estimated strengths and coefficients for the intercept and home advantage to simulate the remaining games 1000 times.

```
unplayedGames <- eliteserie[is.na(eliteserie$yh), c("home", "away")]
nUnplayedGames <- nrow(eliteserie) - nrow(playedGames)
X_unplayed <- matrix(rep(0, nUnplayedGames * 2), ncol = 18, nrow = nUnplayedGames *
    2)
colnames(X_unplayed) <- c("Intercept", "HomeAdvantage", teamNames)
X_unplayed[, "Intercept"] <- rep(1, nUnplayedGames * 2)
j <- 1
for (i in 1:(nUnplayedGames * 2)) {
    # first scores are for the home team
    if (i <= nUnplayedGames) {
        # all home teams receive the home team bonus
        X_unplayed[i, "HomeAdvantage"] <- 1
        X_unplayed[i, unplayedGames$home[i]] <- 1
        X_unplayed[i, unplayedGames$away[i]] <- -1
    } else {
        # using j as we start anew with the playedGames data for
        # the away matches
        X_unplayed[i, unplayedGames$away[j]] <- 1
        X_unplayed[i, unplayedGames$home[j]] <- -1
        j <- j + 1
    }
}
X_unplayed <- subset(X_unplayed, select = -c(colIdx))
estimatedLambda <- exp(X_unplayed %*% estimates)
rankings <- matrix(rep(0, 16 * 1000), nrow = 1000, ncol = 16)
winners <- matrix(rep(character(0), 1000), nrow = 1000, ncol = 1)
colnames(rankings) <- teamNames
for (k in 1:1000) {
    points <- as.matrix(teamPoints)
    yh <- rpois(nUnplayedGames, estimatedLambda[1:nUnplayedGames])
    ya <- rpois(nUnplayedGames, estimatedLambda[-c(1:nUnplayedGames)])
    unplayedGames["yh"] <- yh
    unplayedGames["ya"] <- ya
    for (i in 1:16) {
        dfHome <- subset(unplayedGames, home == teamNames[i])
        dfAway <- subset(unplayedGames, away == teamNames[i])
        # for loop for home data
        for (j in 1:nrow(dfHome)) {
            points[teamNames[i], 2] <- points[i, 2] + (dfHome$yh[j] -
                dfHome$ya[j])
            if (dfHome$yh[j] > dfHome$ya[j]) {
                points[teamNames[i], 1] <- points[teamNames[i], 1] +
```

```
                3
            } else if (dfHome$yh[j] == dfHome$ya[j]) {
                points[teamNames[i], 1] <- points[teamNames[i], 1] +
                  1
            }
        }
        # new for loop for away data
        for (j in 1:nrow(dfAway)) {
            points[i, 2] <- points[i, 2] + (dfAway$ya[j] - dfAway$yh[j])
            if (dfAway$ya[j] > dfAway$yh[j]) {
                points[teamNames[i], 1] <- points[teamNames[i], 1] +
                  3
            } else if (dfAway$ya[j] == dfAway$yh[j]) {
                points[teamNames[i], 1] <- points[teamNames[i], 1] +
                  1
            }
        }
    }
    points <- as.data.frame(points)
    points <- points %>%
        arrange(desc(Scores), desc(GoalDifferences))
    points["Rank"] <- 1:(nrow(points))
    winners[k] <- rownames(points)[1]
    for (l in 1:16) {
        rankings[k, l] <- points[, "Rank"][which(rownames(points) ==
            colnames(rankings)[l])]
    }
}
rankings <- as.data.frame(rankings)
```

Now that we have the rankings for the 1000 runs, we look to do some analysis of the data. What is the mean position of each team? How big is the variance of the position of each team? Which team ended up winning Eliteserien the most? These are some of the questions we look to answer.

First we will look at where each time placed in the runs. We count the amount of placements in each position for each team.

```
teamPlacementCounts <- matrix(0, nrow = 16, ncol = length(teamNames))
colnames(teamPlacementCounts) <- teamNames

# Iterate through each team and placement
for (team in teamNames) {
    for (placement in 1:16) {
        # Count the number of times the team achieved the placement
        count <- sum(rankings[, team] == placement)

        # Store the count in the matrix
        teamPlacementCounts[placement, team] <- count
    }
}
teamPlacementCounts
```

```
##       Rosenborg Aalesund HamKam Sarpsborg 08 Stabæk Tromsø Brann Lillestrøm
##  [1,]         0        0      0            0      0     19     2          0
```

```
##  [2,]         0        0      0            0       0    109     8           0
##  [3,]         0        0      0            0       0    592   119           2
##  [4,]         0        0      0            6       0    227   309          29
##  [5,]         1        0      0           26       0     51   456         135
##  [6,]         5        0      0          262       0      2   101         535
##  [7,]        47        0      0          573       0      0     5         247
##  [8,]       192        0     22           95       1      0     0          46
##  [9,]       285        0     38           32       2      0     0           6
## [10,]       226        0     98            5       4      0     0           0
## [11,]       148        0    173            1      15      0     0           0
## [12,]        67        0    211            0      43      0     0           0
## [13,]        24        0    210            0      84      0     0           0
## [14,]         4        0    186            0     174      0     0           0
## [15,]         1        7     62            0     671      0     0           0
## [16,]         0      993      0            0       6      0     0           0
##       Viking Bodø/Glimt Haugesund Molde Odd Sandefjord Fotball Strømsgodset
##  [1,]    518        461         0     0   0                  0           0
##  [2,]    399        475         0     9   0                  0           0
##  [3,]     73         51         0   163   0                  0           0
##  [4,]      9         13         0   407   0                  0           0
##  [5,]      1          0         0   330   0                  0           0
##  [6,]      0          0         0    79  12                  0           4
##  [7,]      0          0         0    12  84                  4          28
##  [8,]      0          0         3     0 447                 20         161
##  [9,]      0          0        15     0 255                 75         256
## [10,]      0          0        48     0 133                142         264
## [11,]      0          0        95     0  48                198         155
## [12,]      0          0       175     0  18                207          82
## [13,]      0          0       242     0   3                178          33
## [14,]      0          0       287     0   0                134          14
## [15,]      0          0       135     0   0                 41           3
## [16,]      0          0         0     0   0                  1           0
##       Vålerenga
##  [1,]         0
##  [2,]         0
##  [3,]         0
##  [4,]         0
##  [5,]         0
##  [6,]         0
##  [7,]         0
##  [8,]        13
##  [9,]        36
## [10,]        80
## [11,]       167
## [12,]       197
## [13,]       226
## [14,]       201
## [15,]        80
## [16,]         0
```

So who will be crowned champion at the end of the season? According to our model, Viking is the team that wins the most with 511 first place finishes for the 1000 runs, meaning they win 51.1% of the time. Bodø/Glimt follows closely with 463 first place finishes. Our model predicts a close race between these two teams. This result is in line with what we should expect, given that Viking started at first place with 7/8

games left while being the 3rd strongest team, and Bodø/Glimt was comfortably the predicted strongest team while starting in second place, only 2 points of Viking in first. Interestingly, the simulation indicates a small winning chance for Tromsø, with 26 first place finishes. These are the only 3 teams able to finish first in the 1000 runs.

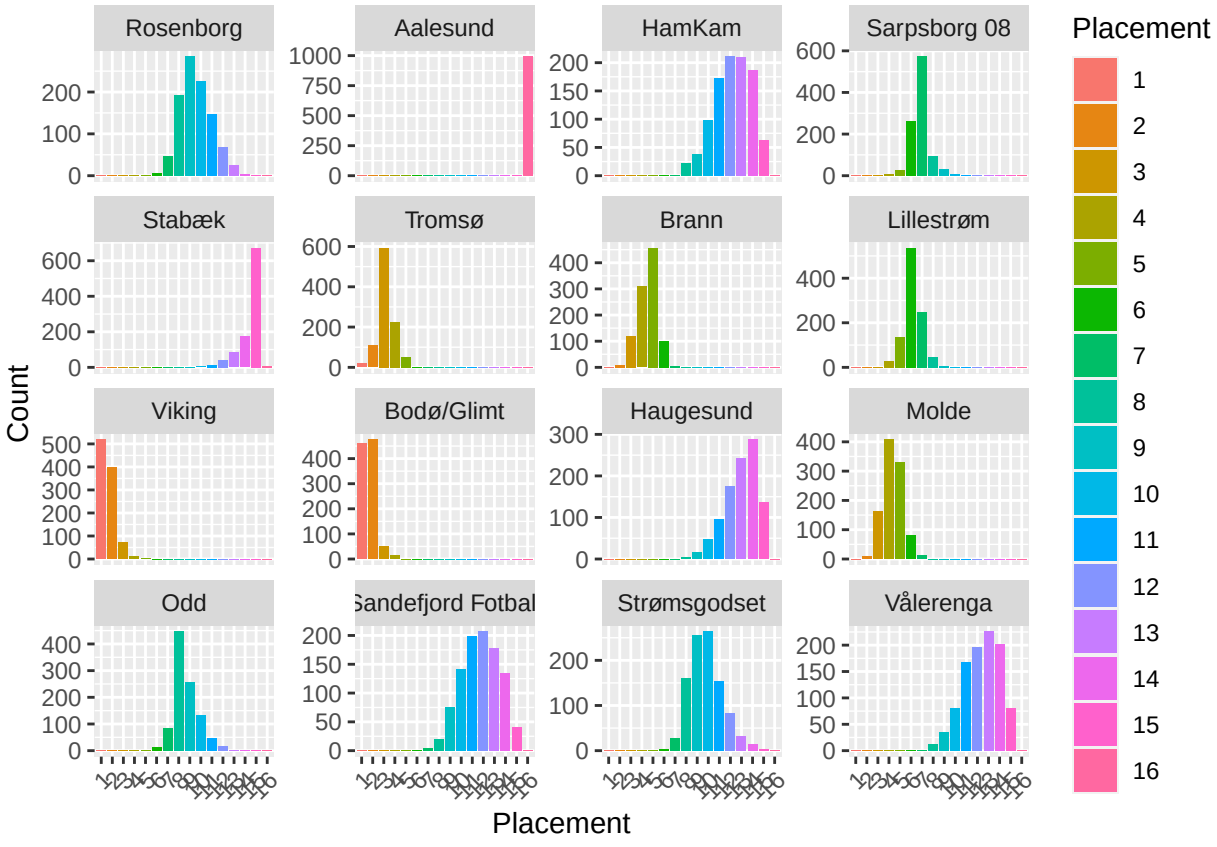
We see that according to our model, Aalesund has a firm grip on last place with 992 last place finishes. Only 8 times did they not finish last, when they managed to climb one spot to 15th place. Aalesund started out in last place and is the predicted weakest team. This result should not come as a surprise given the model coefficients and table. The 8 times Aalesund managed to escape last place, Stabæk finished last in all of them.

For Rosenborg fans, we are sorry to report that our model didn't predict them as winners a single time. According to our model, the best they can hope for is 6th place, where they were 7 times. Most likely is 8th place (210 finished), 9th place (288 finishes) or 10th place (238 finishes).

In the next code section we make a histogram that gives a good visualization of the team placement counts just discussed.

```
# Reshape the matrix into long format
teamPlacementCountsMelt <- melt(teamPlacementCounts, varnames = c("Placement",
    "Team"), value.name = "Count")

# Create histograms for each team using ggplot2
ggplot(data = teamPlacementCountsMelt, aes(x = factor(Placement), y = Count,
    fill = factor(Placement))) + geom_bar(stat = "identity", position = "dodge") +
    facet_wrap(~Team, scales = "free_y", ncol = 4) + labs(x = "Placement",
    y = "Count", fill = "Placement") + theme(axis.text.x = element_text(angle = 45,
    hjust = 1))
```



Next we want to look at the mean placement for each time. The result is calculated in the code below.

```
means <- colMeans(rankings)
means
```

```
##          Rosenborg           Aalesund             HamKam       Sarpsborg 08
##              9.540             15.993             12.197              6.846
##             Stabæk             Tromsø              Brann        Lillestrøm
##             14.436              3.188              4.532              6.158
##             Viking         Bodø/Glimt          Haugesund              Molde
##              1.576              1.616             12.973              4.343
##                Odd Sandefjord Fotball       Strømsgodset          Vålerenga
##              8.644             11.766              9.811             12.381
```

We see the favorites Viking and Bodø/Glimt with mean finish 1.581 and 1.626 respectively. Rosenborg has a mean finish of 9.486. Aalesund has the worst mean of 15.992 (16th is last place).

Finally we want to investigate the uncertainty the model has for the different teams. We also look at the 95% confidence intervals for each team.

```
variances <- sapply(rankings, var)
variances
```

```
##          Rosenborg           Aalesund             HamKam       Sarpsborg 08
##        2.092492492        0.006957958        2.734925926        0.680964965
##             Stabæk             Tromsø              Brann        Lillestrøm
##        1.046950951        0.599255255        0.791767768        0.729765766
##             Viking         Bodø/Glimt          Haugesund              Molde
##        0.456680681        0.416960961        2.130401401        0.836187187
##                Odd Sandefjord Fotball       Strømsgodset          Vålerenga
##        1.300564565        2.908152152        2.261540541        2.626465465
```

```
CI <- sapply(rankings, ci)
CI
```

```
##             Rosenborg     Aalesund      HamKam Sarpsborg 08      Stabæk
## Estimate   9.54000000 15.993000000 12.19700000   6.84600000 14.43600000
## CI lower   9.45023510 15.987823747 12.09437637   6.79479210 14.37250526
## CI upper   9.62976490 15.998176253 12.29962363   6.89720790 14.49949474
## Std. Error 0.04574377  0.002637794  0.05229652   0.02609531  0.03235662
##                Tromsø      Brann Lillestrøm     Viking Bodø/Glimt   Haugesund
## Estimate   3.18800000 4.53200000 6.15800000 1.57600000 1.61600000 12.97300000
## CI lower   3.13996249 4.47678291 6.10498896 1.53406459 1.57592973 12.88242563
## CI upper   3.23603751 4.58721709 6.21101104 1.61793541 1.65607027 13.06357437
## Std. Error 0.02447969 0.02813837 0.02701418 0.02137009 0.02041962  0.04615627
##                Molde        Odd Sandefjord Fotball Strømsgodset   Vålerenga
## Estimate   4.3430000 8.64400000        11.76600000   9.81100000 12.38100000
## CI lower   4.2862552 8.57323141        11.66017625   9.71767956 12.28043186
## CI upper   4.3997448 8.71476859        11.87182375   9.90432044 12.48156814
## Std. Error 0.0289169 0.03606334         0.05392729   0.04755566  0.05124905
```

Interestingly we see that the teams with the highest uncertainty finishes around midtable. The biggest uncertainty is for Sandefjord, which had mean finish of 11.864 and has variance 2.964468468. The team with

the least variance is Aalesund, which had a variance of 0.007943944. This should not be surprising of course, since poor Aalesund finished last 99.2% of the time according to our model. The two best teams Viking and Bodø/Glimt have the second and third least variance. Clearly there is a pattern of smaller variance at the very top and bottom of the table, increasing towards the middle.

We can use the example of 95% confidence interval for Rosenborg, which is (9.39881863, 9.57318137). We see that the 95% confidence interval seems to confident on a short interval compared to the result we got from the simulations. The pattern of small intervals is recurring for the other teams as well. The confidence intervals being to small/confident could indicate weakness in the prediction.

Code used in the myglm package

```
# Select Build, Build and reload to build and lode into the
# R-session.

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")
    best_par <- NULL
    best_loglik <- -Inf
    set.seed(42)
    for (i in 1:100) {
        initial_par <- runif(ncol(X), min = -1, max = 1)
        result <- optim(initial_par, loglik_poi, gr = score_poi, method = "BFGS",
            args = list(X, y))
        if (result$value > best_loglik) {
            best_par <- result$par
            best_loglik <- result$value
        }
    }
    par <- best_par
    params <- optim(par, loglik_poi, gr = score_poi, method = "BFGS",
        args = list(X, y))

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

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

    # Set class name. This is very important!
    class(est) <- "myglm"

    # Return the object with all results
    return(est)
}
```

```r
# You can put these functions somewhere else, but it is practical
# to keep them with the other functions you create optim requires
# that the first argument of the function optim shall optimize is
# the parameters over which minimization is to take place (par) it
# is common to include all other necessary arguments in a list
# called 'args', but this can be done other ways as well
loglik_poi <- function(par, args = list()) {
    # args[[1] is the data matrix X args[[2]] is the response goals
    X <- args[[1]]  # Ensure X is treated as a numeric matrix
    goals <- args[[2]]  # Response variable
    logl <- sum(goals * X %*% par) - sum(exp(X %*% par))
    return(-logl)
    # the Poisson log-likelihood
}

score_poi <- function(par, args = list()) {
    X <- args[[1]]  # Ensure X is treated as a numeric matrix
    goals <- args[[2]]  # Response variable
    score <- t(X) %*% (goals - exp(X %*% par))
    return(-score)
}
```