TMA4315: Compulsory exercise 2

Group 13: Magnus Liland, Jakob Gerhard Martinussen and Emma Skarstein

26.10.2018

Part 1: Logistic Regression

In this exercise we will use binary regression with a logit link to model the probability of successfully climbing a mountain given it's height and prominence as covariates. We assume the number of people who successfully climbs mountain i Y_i, is binomial distributed, $Y_i \sim \text{Bin}(n_i, \pi_i)$, where n_i is the number of people attempting to climb the mountain, and π_i is the probability for success. That is, a model on the form,

1. Model for response: $Y_i \sim \text{Bin}(n_i, \pi_i) \quad \text{for } i = 1, \ldots, 113$
2. Linear predictor: $\eta_i = \mathbf{x}_i^T \boldsymbol{\beta}$
3. Link function: $\eta_i = \ln\left(\frac{\pi_i}{1-\pi_i}\right)$

(a) Log-likelihood function for the parameters

The likelihood for this model is

$$L(\beta) = \prod_{i=1}^{n} f(y_i; \beta) = \prod_{i=1}^{n} \binom{n_i}{y_i} \pi_i^{y_i} (1-\pi_i)^{n_i - y_i}.$$

So the log-likelihood is given

$$l(\beta) = \log L(\beta) = \sum_{i=1}^{n} \left(\log \binom{n_i}{y_i} + y_i \log(\pi_i) + (n_i - y_i) \log(1 - pi_i) \right)$$

$$= \sum_{i=1}^{n} \left(\log \binom{n_i}{y_i} + y_i \log\left(\frac{\exp(\beta x_i^T)}{1+\exp(\beta x_i^T)}\right) + (n_i - y_i) \log\left(\frac{1}{1+\exp(\beta x_i^T)}\right) \right).$$

To arrive at the maximum likelihood estimators for β, one would find the maximum of the log-likelihood by numerical optimization. In practice that would mean to find the score function, $S(\beta)$, by differentiating $l(\beta)$, and solving

$$S(\beta) = \frac{\partial l(\beta)}{\partial \beta} = 0.$$

This is solved by Newton-Raphson or Fisher Scoring algorithm, which for an exponential distribution with the canonical link (like this model) are the same. It is an iterative method on the form

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

where S is the score function, and F is the expected Fisher information matrix. The expected Fisher information matrix is found either by differentiating the score function and taking the expectation, or calculating the covariance matrix of S,

$$F(\beta) = E\left(-\frac{\partial l(\beta)}{\partial \beta \; \partial \beta^T} \right) = Cov(S(\beta)).$$

(b) Calculate and interpret coefficients

We fit the model, with height and prominence as predictors.

```
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2018h/mountains"
mount <- read.table(file = filepath, header = TRUE, col.names = c("height",
    "prominence", "fail", "success"))
model <- glm(cbind(success, fail) ~ height + prominence, data = mount,
    family = "binomial")
summary(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
```

Interpret the parameters

To interpret the coefficients we consider the odds, as

$$\eta_i = \ln\left(\frac{\pi_i}{1-\pi_i}\right) \implies \frac{\pi_i}{1-\pi_i} = \exp(\beta x_i^T) = \exp(\beta_0)\exp(x_{i,\text{height}}\beta_{\text{height}})\exp(x_{i,\text{prominence}}\beta_{\text{prominence}})$$

So if $x_{i,\text{height}}$ changes to $x_{i,\text{height}}+1$ the odds change with a factor $\exp(\beta_{\text{height}})$ as

$$\begin{aligned}\text{odds}_{\text{new}} &= \exp(\beta_0)\exp(x_{i,\text{height}}\beta_{\text{height}})\exp((x_{i,\text{prominence}}+1)\beta_{\text{prominence}}) \\ &= \exp(\beta_0)\exp(x_{i,\text{height}}\beta_{\text{height}})\exp(x_{i,\text{prominence}}\beta_{\text{prominence}})\exp(\beta_{\text{height}}) = \text{odds}_{\text{old}}\exp(\beta_{\text{height}}).\end{aligned}$$

Similarly if $x_{i,\text{prominence}}$ changes to $x_{i,\text{prominence}}+1$ the odds change with a factor $\exp(\beta_{\text{prominence}})$. Calculating these factors we get

```r
exp(model$coefficients[2])
```

```
##    height
## 0.9983659
```

```r
exp(model$coefficients[3])
```

```
## prominence
##   0.999826
```

Both coefficients are negative, meaning the exponential transform of the coefficients are less than 0. This means that the odds decrease. This means the probability of successfully climbing a mountain decreases with increasing values of height and prominence. The factor from a change in height is larger than from a change in prominence.

Discuss the significance of the parameters

By conducting a Wald test we can discuss the significance of the parameters. The null hypothesis is that the true parameter value is zero, and asymptotically we have that

$$\frac{\hat{\beta}_i - 0}{SD(\hat{\beta}_i)} \sim N(0,1).$$

We calculate the p-value for each of the parameters.

```r
standard_error <- sqrt(diag(vcov(model)))
pvalue_coeff1 <- 2 * pnorm(abs((model$coefficients[1] - 0)/standard_error[1]),
    lower.tail = FALSE)
pvalue_coeff2 <- 2 * pnorm(abs((model$coefficients[2] - 0)/standard_error[2]),
    lower.tail = FALSE)
pvalue_coeff3 <- 2 * pnorm(abs((model$coefficients[3] - 0)/standard_error[3]),
    lower.tail = FALSE)
pvalue_coeff1
```

```
## (Intercept)
## 7.47721e-38
```

```r
pvalue_coeff2
```

```
##       height
## 1.031229e-30
```

```r
pvalue_coeff3
```

```
## prominence
## 0.00013302
```

We see that all the coefficients have a small p-value, and we would conclude that all the parameters are significant with any reasonable significance level.

Confidence interval for the parameters

Using the fact that the parameters are asymptotically normally distributed we can construct an 95% confidence interval by

$$[\hat{\beta}_i - z_{0.025}SD(\hat{\beta}_i), \hat{\beta}_i + z_{0.025}SD(\hat{\beta}_i)],$$

where $z_{0.025}$ is the 2.5% lower quantile of the standard normal distribution. For the three parameters in our model we get:

```
confint_coeff1 <- c(model$coefficient[1] - qnorm(0.975) * standard_error[1],
    model$coefficient[1] + qnorm(0.975) * standard_error[1])
confint_coeff2 <- c(model$coefficient[2] - qnorm(0.975) * standard_error[2],
    model$coefficient[2] + qnorm(0.975) * standard_error[2])
confint_coeff3 <- c(model$coefficient[3] - qnorm(0.975) * standard_error[3],
    model$coefficient[3] + qnorm(0.975) * standard_error[3])
confint_coeff1
```

```
## (Intercept) (Intercept)
##    11.60015    15.77154
```

```
confint_coeff2
```

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

```
confint_coeff3
```

```
##    prominence    prominence
## -2.632283e-04 -8.473354e-05
```

We can see that the confidence interval for the coefficients β_{height} and $\beta_{\text{prominence}}$ does not include numbers above 1, implying that we can with a confidence level of 95% claim that the effect of both height and prominence have a negative effect of the odds.

Confidence interval for exponential transform

We calculate the confidence interval for the exponential transform of the parameters by simply taking the exponential transform of the upper and lower limit. This transform does not have an intuitive interpretation for the intercepts, like for the two coefficients for height and prominence - the multiplicative effect on the odds. We therefore only consider these confidence intervals.

```
exp_confint_coeff2 <- exp(confint_coeff2)
exp_confint_coeff3 <- exp(confint_coeff3)
exp_confint_coeff2
```

```
##    height    height
## 0.9980882 0.9986437
```

```
exp_confint_coeff3
```

```
## prominence prominence
##  0.9997368  0.9999153
```

We see that the confidence interval for both parameters are less than zero, indicating that both parameters with a confidence level of 95% have a decreasing effect on the odds.

(c) Deviance residual plot and probability plot

Deviance residual plot

We have that the deviance is given by

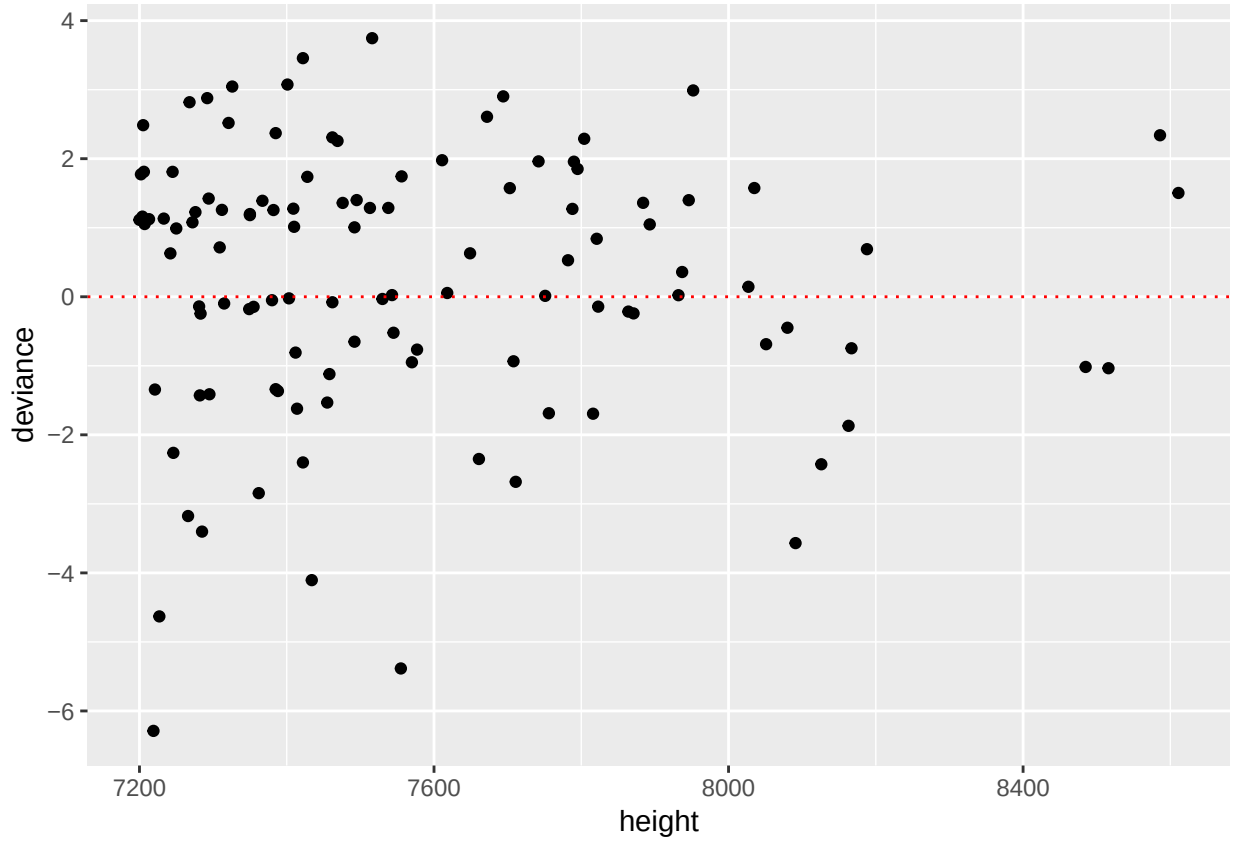
$$D = -2(l(\hat{\beta}) - l(\tilde{\beta})) = -2(l(\hat{\pi}) - l(\tilde{\pi})),$$

where $\tilde{\pi}$ denote the probability for the saturated model. This probability is simply $\tilde{\pi}_i = y_i/n_i$, such that the estimated number of successes is equal to the actual number for each mountain. The deviance for mountain i is

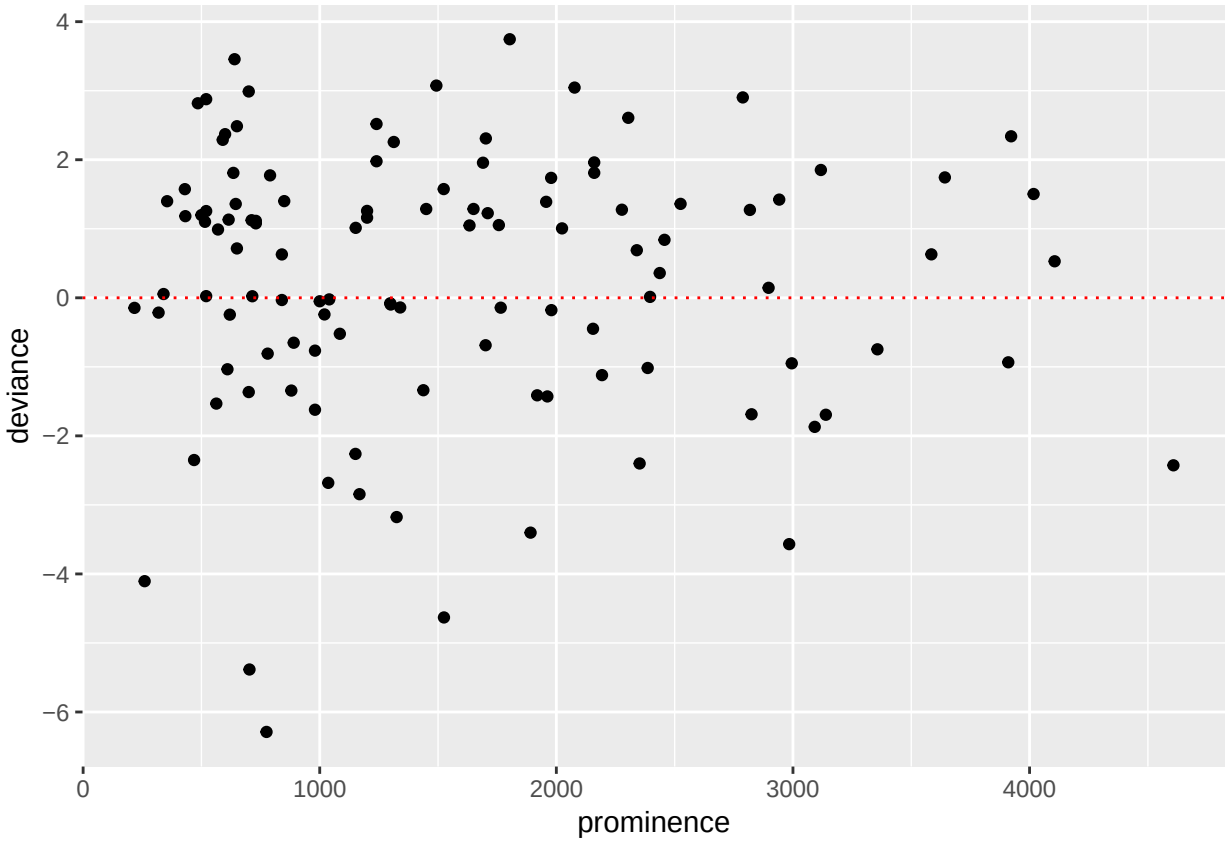
$$D_i = \text{sign}(y_i - \hat{y}_i) \cdot (-2)(l(\hat{\pi}_i) - l(\tilde{\pi}_i))$$
$$= \text{sign}(y_i - \hat{\pi}_i n_i) \cdot (-2)(n_i(\log(\hat{\pi}_i) - \log(y_i/n_i) + (n_i - y_i)(\log(1 - \hat{\pi}_i) - \log(1 - (\frac{y_i}{n_i})))).$$

We calculate the deviance of each mountain with the built in R-function, and plot the deviance against height and prominence:

```
deviance_vector <- residuals(model, type = "deviance")
gg_frame <- data.frame(height <- mount$height, prominence <- mount$prominence,
    deviance <- deviance_vector)
ggplot(data = gg_frame, aes(x = height, y = deviance)) + geom_point() +
    geom_hline(yintercept = 0, linetype = "dotted", colour = "red")
```




```
ggplot(data = gg_frame, aes(x = prominence, y = deviance)) + geom_point() +
    geom_hline(yintercept = 0, linetype = "dotted", colour = "red")
```



It seems that the residuals are reasonably randomly distributed around 0 for both "low" mountains and high mountains. However, it seems that the model underestimates the probability of successes for some of the lowest mountains, as there are some large residuals for these. The residuals for the somewhat larger mountains seems to in general be closer to zero. As a consequence of the "easy" mountains, it seems that the model often overestimates the probability of success for the more "average" mountains, as there seems to in general be more negative residuals than positive.

For the prominence we see pretty much the same pattern as for the height. The residuals seem to be reasonably randomly distributed around zero, expect for a few outliers for small prominence. It seems that the model also underestimates the probability of success in this case. For larger prominence the residuals are in general smaller. As with the previous plot it seems to in general be more extreme negative residuals than positive.

Estimated probabilities as function of the covariates

We proceed by plotting the estimated probability as a function of the covariates height and prominence. We have that

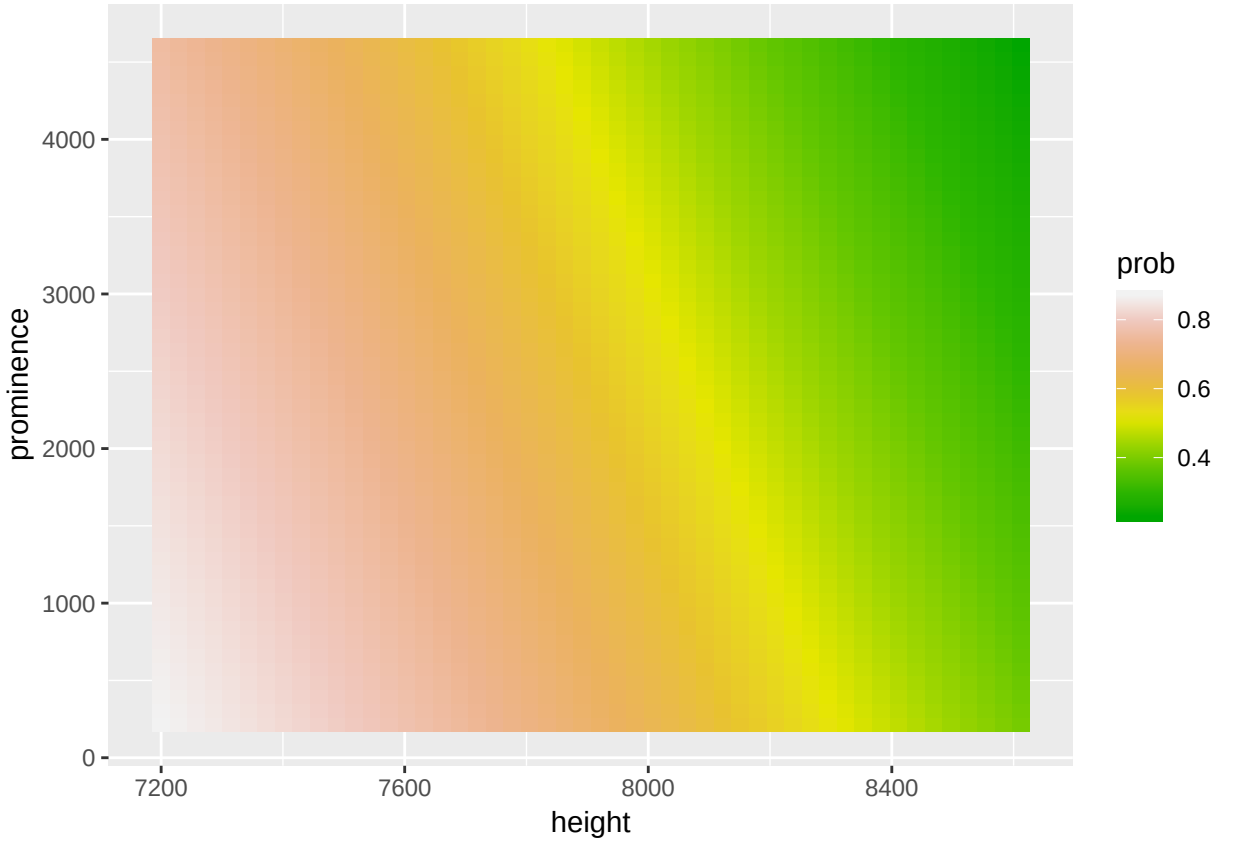
$$\hat{\pi}_i = \frac{\exp(\hat{\beta} x_i^T)}{1 + \exp(\hat{\beta} x_i^T)}.$$

We calculate this value for a set of heights and prominences from the minimum value to the maximum value in the data set for both. We plot the corresponding estimated probabilities against the covariates.

```
N = 50
height_vector <- seq(min(mount$height), max(mount$height), length.out = N)
```

```r
prominence_vector <- seq(min(mount$prominence), max(mount$prominence),
    length.out = N)

gg_frame <- expand.grid(height = height_vector, prominence = prominence_vector)
gg_frame$prob <- vector("numeric", length(gg_frame$height))
for (i in 1:length(gg_frame$height)) {
    nu_temp <- as.numeric(model$coefficients[1] + model$coefficients[2] *
        gg_frame$height[i] + model$coefficients[3] * gg_frame$prominence[i])
    gg_frame$prob[i] <- exp(nu_temp)/(1 + exp(nu_temp))
}
ggplot(gg_frame, aes(height, prominence, z = prob, fill = prob)) + geom_raster() +
    scale_fill_gradientn(colours = terrain.colors(10))  #+ geom_contour()
```



We see that the probability for success is close to 1 for a low mountain with low prominence, while the probability is smaller for a high mountain with a high prominence. The probability decreases faster with increasing height than with increasing prominence, but is clearly decreasing in both.

(d) Prediction of success probability for Everest and Chogolisa

Mount Everest

We have that $x_{\text{everest, height}} = 8848$ and $x_{\text{everest, prominence}} = 8848$. We calculate the corresponding probability of success by the formula above. For the confidence interval we use a similar approach as for the coefficients, using the fact that the coefficients are asymptotically normal, meaning that the linear predictor is asymptotically normal

$$\hat{\eta}_e = \hat{\beta}x_e^T = \hat{\beta}_0 + x_{e,\text{height}}\hat{\beta}_{\text{height}} + x_{e,\text{prominence}}\hat{\beta}_{\text{prominence}} \implies \hat{\eta}_e \sim N(\mu_{\eta_e} = \beta x_e^T, \sigma^2_{\eta_e} = x_e Cov(\beta) x_e^T).$$

We construct a confidence interval for the linear predictor with the same approach as employed earlier, namely

$$[\hat{\eta}_e - z_{0.025}\sigma_{\eta_e}, \hat{\eta}_e + z_{0.025}\sigma_{\eta_e}],$$

Finally, to get a 95% confidence interval for the probability we transform the interval above by the equation for probability

$$\left[\frac{\exp(\hat{\eta}_e - z_{0.025}\sigma_{\eta_e})}{1+\exp(\hat{\eta}_e - z_{0.025}\sigma_{\eta_e})}, \frac{\exp(\hat{\eta}_e + z_{0.025}\sigma_{\eta_e})}{1+\exp(\hat{\eta}_e + z_{0.025}\sigma_{\eta_e})}\right]$$

```
height_everest <- 8848
prominence_everest <- 8848
nu_everest <- as.numeric(model$coefficients[1] + model$coefficients[2] *
    height_everest + model$coefficients[3] * prominence_everest)
pi_everest <- exp(nu_everest)/(1 + exp(nu_everest))
pi_everest
```

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

```
sd_nu_everest <- sqrt(t(c(1, height_everest, prominence_everest)) %*%
    vcov(model) %*% c(1, height_everest, prominence_everest))
confint_nu_everest <- c(nu_everest - qnorm(0.975) * sd_nu_everest, nu_everest +
    qnorm(0.975) * sd_nu_everest)
confint_pi_everest <- exp(confint_nu_everest)/(1 + exp(confint_nu_everest))
confint_pi_everest
```

```
## [1] 0.05486572 0.14173033
```

```
max(mount$height)
```

```
## [1] 8611
```

```
mean(mount$height)
```

```
## [1] 7568.283
```

```
max(mount$prominence)
```

```
## [1] 4608
```

We see that the estimated probability of successfully climbing mount Everest is quite low, around 9%. The 95% confidence interval is rather large, from just over 5% to 14%. We note that Mount Everest is rather extreme compared to the other mountains in the data set, being approx. 400 meters higher than the highest in the data set (more than 1000 above the average), and almost twice as large prominence as the largest in the data set (almost 6 times the average prominence). This means the model does not necessarily describe Mount Everest as good as the other mountains.

Chogolisa

We have that $x_{\text{chogolisa, height}} = 7665$ and $x_{\text{chogolisa, prominence}} = 1624$. We calculate the estimated probability for success, and corresponding 95% confidence interval, in the same way as for Mount Everest. In addition we calculate the estimated $y_{\text{chogolisa}}$ by $\hat{y}_{\text{chogolisa}} = \hat{\pi}_{\text{chogolosa}} n_{\text{chogolisa}}$, and the confidence interval.

```
height_chogolisa <- 7665
prominence_chogolisa <- 1624
nu_chogolisa <- as.numeric(model$coefficients[1] + model$coefficients[2] *
    height_chogolisa + model$coefficients[3] * prominence_chogolisa)
pi_chogolisa <- exp(nu_chogolisa)/(1 + exp(nu_chogolisa))
pi_chogolisa
```

```
## [1] 0.7042924
```

```
sd_nu_chogolisa <- sqrt(t(c(1, height_chogolisa, prominence_chogolisa)) %*%
    vcov(model) %*% c(1, height_chogolisa, prominence_chogolisa))
confint_nu_chogolisa <- c(nu_chogolisa - qnorm(0.975) * sd_nu_chogolisa,
    nu_chogolisa + qnorm(0.975) * sd_nu_chogolisa)
confint_pi_chogolisa <- exp(confint_nu_chogolisa)/(1 + exp(confint_nu_chogolisa))
y_chogolisa_hat <- pi_chogolisa * 22
confint_y_chogolisa_hat <- confint_pi_chogolisa * 22
```

We see that the estimated number of people successfully climbing the mountain is somewhat smaller than the true number. This is consistent with our observation that the model have a tendency to underestimate the probability for mountains with small prominence, which is the case here.

Part 2: Poisson regression – Eliteserien 2018

In this exercise we want to answer the following question for the 2018 Eliteserien: *How likely or unlikely are Rosenborg to become champions?* Our data consists of the results of all played football matches of the 2018 Eliteserien, with one row per match, containing the home-team: `tippeliga$home`, the away-team: `tippeliga$away`, and the number of goals scored by each team, `tippeliga$yh` and `tippeliga$ya`. Using this data, we assume that the score of the home team is independent of the away team and that each team has a single parameter measuring their strength. These strength parameters are denoted by β, such that $\beta_{Tromsoe}$ is the strength parameter of Tromsoe, β_{Molde} is the strength parameter for Molde, etc. Then, for a match between two teams A and B where A is the home team, the score for team A will have a Poisson distribution with mean λ where $\ln \lambda = \beta_0 + \beta_{home} + \beta_A - \beta_B$, and team B's score will have a Poisson distribution with mean λ where $\ln \lambda = \beta_0 - \beta_A + \beta_B$. Here, β_0 is our intercept and β_{home} is a home advantage parameter.

a) Testing independence between home and away team with contingency test

We wish to test the assumption of independence between the goals made by the home and away team. For testing this independence, we use construct a *contingency table* and preform a *Pearson's χ^2 test*.

We construct the contingency table in *R*, defining the groups (where the groups are the number of goals scored) so that at least 80% of the cells in the table have count 5 or more, and none of the cells have count 0 (these are assumptions for the Pearson's chi-squared test to be valid).

```
temp_ya <- tippeliga$ya
temp_yh <- tippeliga$yh
temp_ya[temp_ya > 3] <- rep(3, length(temp_ya[temp_ya > 3]))
temp_yh[temp_yh > 3] <- rep(3, length(temp_yh[temp_yh > 3]))
goals_table <- table(temp_yh, temp_ya)
goals_table
```

```
##        temp_ya
## temp_yh  0  1  2  3
##       0  8 18  3  2
```

```
##       1 19 26 15  8
##       2 10 14 13  5
##       3 21 17 10  3
```

The column names of this table denote the number of goals scored by the home team, while the row names denote the number of goals scored by the away team. Then the element (0,0) in the table denotes the number of games where both teams scored zero goals (eight games had this result), and the element (2,3) denotes the number of games where the away team scored 2 goals and the away team scored 3 goals or more (ten games had this result), as examples.

Let H_i be the event that the home team scores i goals and similarly A_j be the event that the away team scores j goals, and let $p_i = P(H_i)$, $q_j = P(A_j)$ and $p_{ij} = P(H_i, A_j)$. Under our assumption that H_i and A_j are independent, $p_{ij} = p_i \cdot q_j$. So for each pair (i, j), we compare np_iq_i to the actual observed number of games with the combination (i, j), which we denote k_{ij}. This is what is done in the Pearsons χ^2 test. Our test statistic is the sum over all rows and columns of the table of $\frac{(k_{ij}-np_iq_i)}{np_{ij}}$. If this test statistic is larger that $\chi^2_{\alpha,(4-1)(4-1)}$, we reject H_0 at the α level of significance. These calculations could be done manually, or we can use the R function `chisq.test()`.

```
chisq.test(goals_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  goals_table
## X-squared = 12.101, df = 9, p-value = 0.2077
```

We see that our observed test statistic has a value of 12.1007628, while $\chi^2_{0.1,9} = 14.6836566$, along with a p-value of 0.2076882, which means that the assumption that the score of the home team and the score of the away team are independent seems to be quite reasonable.

b) Team rankings

After a match, the winning team gets 3 points and the loser gets 0, and if it is a draw both teams get 1 point each. We want to calculate the ranking for each team, as well as the goal differences.

```
tippeliga_table <- data.frame(team = unique(tippeliga$home), points = vector("numeric",
    length(unique(tippeliga$home))), "goal_diff" <- vector("numeric",
    length(unique(tippeliga$home))))
colnames(tippeliga_table) <- c("team", "points", "goal_diff")
for (i in 1:length(tippeliga_table$team)) {
    tippeliga_table$points[i] <- (sum(3 * (tippeliga$yh[tippeliga$home ==
        tippeliga_table$team[i]] > tippeliga$ya[tippeliga$home == tippeliga_table$team[i]])) +
        sum(3 * (tippeliga$ya[tippeliga$away == tippeliga_table$team[i]] >
            tippeliga$yh[tippeliga$away == tippeliga_table$team[i]])) +
        sum(1 * (tippeliga$ya[tippeliga$away == tippeliga_table$team[i]] ==
            tippeliga$yh[tippeliga$away == tippeliga_table$team[i]])) +
        sum(1 * (tippeliga$yh[tippeliga$home == tippeliga_table$team[i]] ==
            tippeliga$ya[tippeliga$home == tippeliga_table$team[i]])))


    tippeliga_table$goal_diff[i] <- (sum(tippeliga$yh[tippeliga$home ==
        tippeliga_table$team[i]]) + sum(tippeliga$ya[tippeliga$away ==
        tippeliga_table$team[i]]) - sum(tippeliga$ya[tippeliga$home ==
        tippeliga_table$team[i]]) - sum(tippeliga$yh[tippeliga$away ==
        tippeliga_table$team[i]]))
```

```
}

tippeliga_table <- tippeliga_table[order(-tippeliga_table$points, -tippeliga_table$goal_diff),
    ]
rownames(tippeliga_table) <- 1:length(tippeliga_table$team)
tippeliga_table
```

```
##                  team points goal_diff
## 1           Rosenborg     52        23
## 2               Brann     48        13
## 3               Molde     43        18
## 4           Haugesund     41         8
## 5          Ranheim_TF     38        -2
## 6          Vaalerenga     36        -2
## 7                 Odd     34         6
## 8             Tromsoe     33         2
## 9          Sarpsborg08     32         5
## 10       Kristiansund     31        -3
## 11         BodoeGlimt     27        -2
## 12      Stroemsgodset     26         0
## 13        Lillestroem     25       -11
## 14            Stabaek     23       -14
## 15              Start     23       -18
## 16 Sandefjord_Fotball     15       -23
```

c) Preforming Poisson regression

We now implement the actual Poisson regression to estimate the intercept, home advantage and strength parameter for each team.

Our linear predictors are of the form

$$\ln \mathrm{E}(Y) = \beta_0 + \beta_{\text{home}} x_{\text{home}} + \beta_{\text{BodoeGlimt}} x_{\text{BodoeGlimt}} + \cdots + \beta_{\text{Vaalerenga}} x_{\text{Vaalerenga}}.$$

Here, x_{home} is equal to 1 if the score Y is for the home team, and 0 if it is for the away team. If A is the home team and B is the away team, then if the score Y is for team A, then $x_A = 1$, $x_B = -1$ and the covariates for all other teams are 0.

We begin by constructing the design matrix X.

```
# Design matrix y = X beta
X <- matrix(0, nrow <- 384, ncol <- 18)
X[, 1] <- rep(1, 384)
colnames(X) <- c("Intercept", "HomeAdvantage", as.character(tippeliga_table$team))
for (i in 1:length(tippeliga$home)) {
    X[i, 2] <- 1
    X[i, colnames(X) == tippeliga$home[i]] <- 1
    X[i, colnames(X) == tippeliga$away[i]] <- -1
}

for (i in 1:length(tippeliga$home)) {
    X[i + length(tippeliga$home), 2] <- 0
    X[i + length(tippeliga$home), colnames(X) == tippeliga$home[i]] <- -1
    X[i + length(tippeliga$home), colnames(X) == tippeliga$away[i]] <- 1
```

```
}

Y <- c(tippeliga$yh, tippeliga$ya)
```

Next we implement the loglikelihood function, in order to estimate the parameters. The likelihood function is

$$L(\beta) = \prod_{i=1}^{n} \frac{\lambda_i^{y_i}}{y_i!} e^{-\lambda_i},$$

yielding the loglikelihood

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

where $\lambda_i = e^{\eta_i} = e^{x_i^T \beta}$.

```
loglik <- function(beta_optim, Y, X, beta_SF) {
    beta <- c(beta_optim, beta_SF)
    sum <- 0
    for (i in 1:length(Y)) {
        lambda_i <- exp(X[i, ] %*% beta)
        sum <- sum + Y[i] * log(lambda_i) - lambda_i - log(factorial(Y[i]))
    }
    return(-sum)
}
```

By now using `optim()`in *R*, we obtain the estimates for the intercept, home advantage, and strength parameters for every team.

```
optim_ret <- optim(par = (1:(dim(X)[2] - 1)), loglik, X = X, Y = Y, beta_SF = 1,
    method = "BFGS")
beta <- c(optim_ret$par, 1)
names(beta) <- colnames(X)
```

All the parameter estimates are:

```
##           Intercept      HomeAdvantage           Rosenborg
##           0.1003017           0.4020372           1.6589440
##               Brann               Molde           Haugesund
##           1.5176464           1.5712022           1.4331034
##          Ranheim_TF         Vaalerenga                 Odd
##           1.3003227           1.3064532           1.3919165
##             Tromsoe        Sarpsborg08       Kristiansund
##           1.3523221           1.3895071           1.3043574
##          BodoeGlimt      Stroemsgodset         Lillestroem
##           1.2919565           1.3416185           1.1591283
##             Stabaek               Start Sandefjord_Fotball
##           1.1438431           1.0660611           1.0000000
```

Based on these estimates we also make a ranking for all the teams:

```
t(t(sort(tail(beta, n = 16), decreasing = TRUE)))
```

```
##                       [,1]
## Rosenborg         1.658944
## Molde             1.571202
## Brann             1.517646
## Haugesund         1.433103
```

```
## Odd                1.391917
## Sarpsborg08        1.389507
## Tromsoe            1.352322
## Stroemsgodset      1.341618
## Vaalerenga         1.306453
## Kristiansund       1.304357
## Ranheim_TF         1.300323
## BodoeGlimt         1.291957
## Lillestroem        1.159128
## Stabaek            1.143843
## Start              1.066061
## Sandefjord_Fotball 1.000000
```

Comparing this ranking to the one produced in b), we note that the top and bottom ranking are the same, but that there is some variation among the middle rankings. The first ranking is based only on the results up until October 1st, while the second ranking is based on the estimated strengths of the different teams. The reason that they are slightly different may be due to the matching of the teams in the last part of the year, that is, the teams that do worse on the first ranking than the second one are maybe matched up against a lot of "bad" teams in the last part of the year, and thus their ranking is estimated to be better in the ranking in 2c.

d) Simulation

We now want to investigate the results of future matches using our estimated strengths. We simulate the remainding matches in this years "Eliteserie". This we do by constructing the design matrix X_{up} of these matches, and compute the corresponding λ for each team for each match by

$$\hat{\lambda}_{up} = X_{up}\hat{\beta}.$$

The simulations is to draw $N = 1000$ realizations of the scores based on a Poisson distribution with λ as specified above.

```
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2018h/unplayed2018"
unplayed <- read.table(file = filepath, header = TRUE, colClasses = c("character",
    "character"))

X_up <- matrix(0, nrow <- 2 * length(unplayed$home), ncol <- 18)
X_up[, 1] <- rep(1, 2 * length(unplayed$home))
colnames(X_up) <- c("Intercept", "HomeAdvantage", as.character(tippeliga_table$team))
for (i in 1:length(unplayed$home)) {
    X_up[i, 2] <- 1
    X_up[i, colnames(X_up) == unplayed$home[i]] <- 1
    X_up[i, colnames(X_up) == unplayed$away[i]] <- -1
}

for (i in 1:length(unplayed$home)) {
    X_up[i + length(unplayed$home), 2] <- 0
    X_up[i + length(unplayed$home), colnames(X_up) == unplayed$home[i]] <- -1
    X_up[i + length(unplayed$home), colnames(X_up) == unplayed$away[i]] <- 1
}

lambda <- exp(X_up %*% beta)
n_matches <- length(lambda)
N = 100  #simulations
# TODO: N=1000
```

```r
position_simulation <- matrix(0, nrow <- N, ncol <- 16)
colnames(position_simulation) <- tippeliga_table$team
rownames(position_simulation) <- 1:N


for (j in 1:N) {
    tippeliga_table_sim <- tippeliga_table
    unplayed$yh <- rpois(n_matches/2, lambda = head(lambda, n = (n_matches/2)))
    unplayed$ya <- rpois(n_matches/2, lambda = tail(lambda, n = (n_matches/2)))



    for (i in 1:length(tippeliga_table_sim$team)) {
        tippeliga_table_sim$points[i] <- (tippeliga_table_sim$points[i] +
            sum(3 * (unplayed$yh[unplayed$home == tippeliga_table_sim$team[i]] >
                unplayed$ya[unplayed$home == tippeliga_table_sim$team[i]])) +
            sum(3 * (unplayed$ya[unplayed$away == tippeliga_table_sim$team[i]] >
                unplayed$yh[unplayed$away == tippeliga_table_sim$team[i]])) +
            sum(1 * (unplayed$ya[unplayed$away == tippeliga_table_sim$team[i]] ==
                unplayed$yh[unplayed$away == tippeliga_table_sim$team[i]])) +
            sum(1 * (unplayed$yh[unplayed$home == tippeliga_table_sim$team[i]] ==
                unplayed$ya[unplayed$home == tippeliga_table_sim$team[i]])))


        tippeliga_table_sim$goal_diff[i] <- (tippeliga_table_sim$goal_diff[i] +
            sum(unplayed$yh[unplayed$home == tippeliga_table_sim$team[i]]) +
            sum(unplayed$ya[unplayed$away == tippeliga_table_sim$team[i]]) -
            sum(unplayed$ya[unplayed$home == tippeliga_table_sim$team[i]]) -
            sum(unplayed$yh[unplayed$away == tippeliga_table_sim$team[i]]))
    }

    tippeliga_table_sim <- tippeliga_table_sim[order(-tippeliga_table_sim$points,
        -tippeliga_table_sim$goal_diff), ]
    rownames(tippeliga_table_sim) <- 1:length(tippeliga_table$team)

    for (k in 1:length(tippeliga_table_sim$team)) {
        position_simulation[j, tippeliga_table_sim[k, ]$team == colnames(position_simulation)] <- k
    }
}

table_sim <- colMeans(position_simulation)
table_sim <- t(t(table_sim))
colnames(table_sim) <- "Mean position"
table_sim <- data.frame(table_sim)
table_sim$old.position <- 1:16
lowest <- vector("numeric", 16)
highest <- vector("numeric", 16)
for (i in 1:16) {
    lowest[i] <- max(position_simulation[, i])
    highest[i] <- min(position_simulation[, i])
}
table_sim$highest.position <- highest
table_sim$lowest.position <- lowest
```

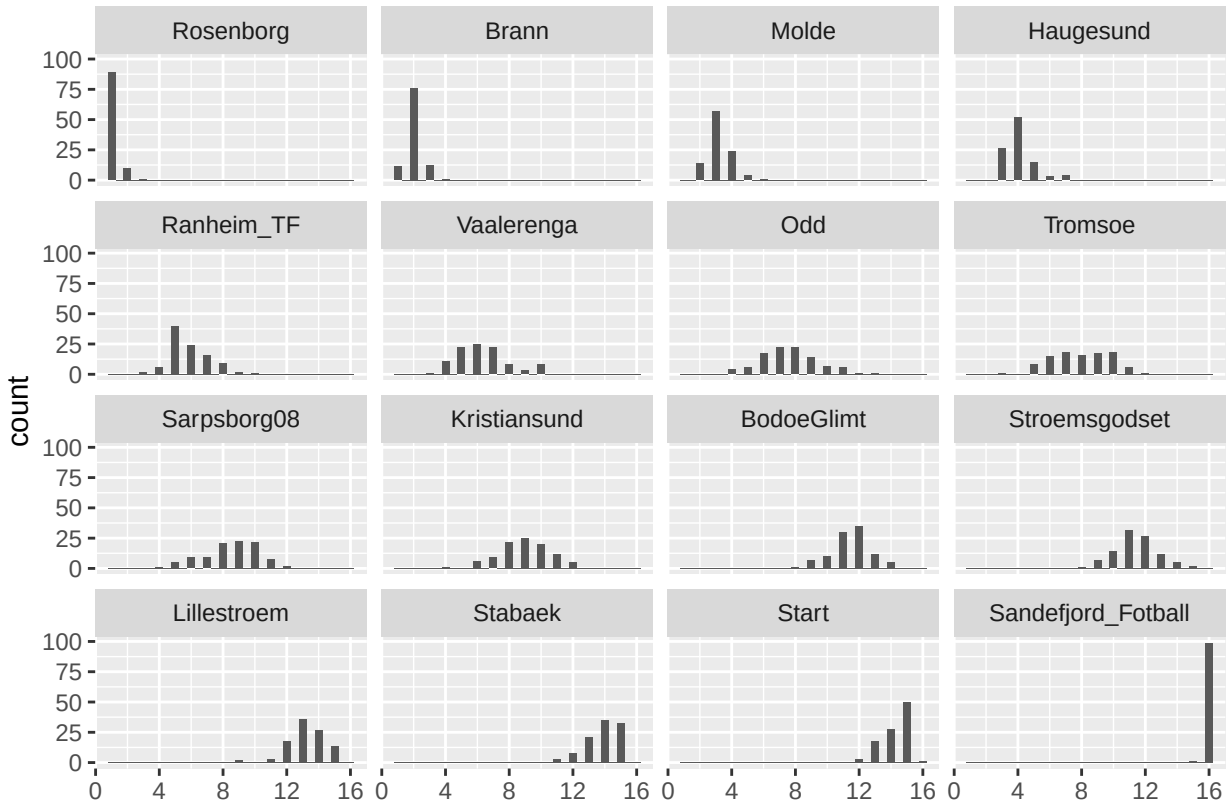
```
table_sim <- table_sim[order(table_sim$Mean.position), , drop = FALSE]
table_sim$team <- rownames(table_sim)
table_sim <- table_sim[, c(5, 1, 2, 3, 4)]
rownames(table_sim) <- 1:16
table_sim
```

```
##                  team Mean.position old.position highest.position
## 1           Rosenborg          1.12            1                1
## 2               Brann          2.03            2                1
## 3               Molde          3.21            3                2
## 4           Haugesund          4.07            4                3
## 5          Ranheim_TF          5.86            5                3
## 6          Vaalerenga          6.32            6                3
## 7                 Odd          7.65            7                4
## 8             Tromsoe          7.98            8                3
## 9         Sarpsborg08          8.53            9                4
## 10       Kristiansund          8.96           10                4
## 11      Stroemsgodset         11.43           12                8
## 12         BodoeGlimt         11.47           11                8
## 13        Lillestroem         13.23           13                9
## 14            Stabaek         13.87           14               11
## 15              Start         14.28           15               12
## 16 Sandefjord_Fotball         15.99           16               15
##    lowest.position
## 1                3
## 2                4
## 3                6
## 4                7
## 5               10
## 6               10
## 7               13
## 8               12
## 9               12
## 10              12
## 11              15
## 12              14
## 13              15
## 14              15
## 15              16
## 16              16
```

The result of the $N = 1000$ simulations is displayed in the matrix `table_sim`, where the column `old.position` indicate where the team was placed as of October 1st, while the table is sorted by the mean placement in the simulation. It is clear Rosenborg end up as the winner in most of the simulations, while Sandefjord ends up as the loser in most. The remainding teams have a somewhat larger spread in their placement, but most teams finish at the same position as they had at October 1st, with some exceptions.

To get a clearer view of the spread in the placement, we consider the histogram of each teams placement.

```
position_simulation <- melt(data.frame(position_simulation))
ggplot(data = position_simulation) + geom_histogram(aes(position_simulation$value),
    binwidth = 0.5) + facet_wrap(~variable) + xlab("")
```



It is clear from the histograms that the teams at the top and bottom of the table have a clear tendency to stay where they are, with large spikes around their October 1st position, while the simulated position of the teams at the middle is more spread. This is consistent with our observations concering the strength parameters, β, as there are a lot of teams around the middle with very similar strength, while the teams at the top and at the bottom stand out with significantly higher and lower strength respectivley.