TMA4315 Generalized Linear Models

Compulsory exercise 2: Logistic regression and Poisson regression

Håvard Fossdal* Rasmus Grødeland† Yawar Mahmood‡

11 October, 2024

Logistic Regression

In this section, we will analyse a dataset consisting of 113 of the world's highest mountains. Our aim is to model the probability of successfully ascending different mountains using logistic regression. We will be regressing on the height and topographic prominence of each mountain, both of which are included in the dataset for all 113 observations, where they are measured in meters.

We let y_i represent the number of successful ascents and n_i the total number of attempts at ascending the ith mountain. Denoting the probability of success by π_i, the logistic regression model can be stated as,

$$Y_i \sim \text{Bin}(n_i, \pi_i), \quad \eta_i = \text{logit}(\pi_i) = \ln\left(\frac{\pi_i}{1-\pi_i}\right), \qquad \text{for } i = 1, \ldots, 113,$$

where logit denotes the logistic link function and $\eta_i := \boldsymbol{x}_i^T \boldsymbol{\beta}$ denotes the linear predictor. Here $\boldsymbol{x}_i$ is a vector of covariates corresponding to the ith observation, and $\boldsymbol{\beta}$ a vector of regression coefficients which has to be estimated when fitting the model.

Log-likelihood as a Function of the Regression Coefficients

Assuming that Y_i conditioned on the regressors $\boldsymbol{x}_i$ is independently distributed, the log-likelihood function for the regression coefficients $\ell(\boldsymbol{\beta})$ can be written as the sum of the log-likelihood contributions from each mountain $\ell_i(\boldsymbol{\beta})$. Given that Y_i follows a binomial distribution, the log-likelihood function can then be stated as,

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{113} \ell_i(\boldsymbol{\beta}) = \sum_{i=1}^{113} \ln\left(\binom{n_i}{y_i} \pi_i^{y_i} (1-\pi_i)^{n_i - y_i}\right),$$

where $\boldsymbol{\beta}$ is a function of the success probability π_i through the logistic link. Using properties of logarithms, we can simplify the expression to yield

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{113} \left(y_i \ln(\pi_i) + (n_i - y_i)\ln(1-\pi_i)\right) + c_i,$$

where $c_i = \ln(n_i!) - \ln(y_i!) - \ln((n_i - y_i)!)$ is a constant with respect to $\boldsymbol{\beta}$. If we now solve for π_i as a function of $\boldsymbol{\beta}$ through the linear predictor and logic link function, we find that

$$\pi_i(\boldsymbol{\beta}) = \frac{\exp(\eta_i)}{1+\exp(\eta_i)} = \frac{\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^T\boldsymbol{\beta})}.$$

*Group 2, Department of Mathematical Sciences, haafo@stud.ntnu.no
†Group 2, Department of Mathematical Sciences, rasmug@stud.ntnu.no
‡Group 2, Department of Mathematical Sciences, yawarm@stud.ntnu.no

Substituting π_i into the log-likelihood function, we obtain the following relationship upon simplifying the resulting expression,

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{113} \left(y_i \boldsymbol{x}_i^T \boldsymbol{\beta} - n_i \ln\left(1 + \exp(\boldsymbol{x}_i^T \boldsymbol{\beta})\right)\right) + c_i.$$

Maximum Likelihood Estimation (MLE)

To estimate the parameters $\boldsymbol{\beta}$ via maximum likelihood, we compute the value of $\boldsymbol{\beta}$ that maximises the log-likelihood $\ell(\boldsymbol{\beta})$. In practice, we use a step-wise process characterised by the following steps:

1. **Computing the Score Function.** We start by computing the gradient of the log-likelihood function with respect to $\boldsymbol{\beta}$, known as the score $\boldsymbol{s}(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$. This measures the sensitivity of the log-likelihood to changes in the regression coefficients. If the log-likelihood is a concave function of the regression coefficients, as it is in the case of logistic regression, we know that any maximum of the function is in fact a global maximum of the log-likelihood.

2. **Equating the Score Function to Zero.** To find the maximum likelihood estimates, we solve $\boldsymbol{s}(\hat{\boldsymbol{\beta}}) = 0$ for $\hat{\boldsymbol{\beta}}$. This system may be nonlinear, which often makes closed-form solutions difficult to obtain. Given that $\ell(\boldsymbol{\beta})$ is strictly concave, $\hat{\boldsymbol{\beta}}$ would then be the unique regression coefficients that maximises the log-likelihood.

3. **Numerical Methods.** Since $\boldsymbol{s}(\hat{\boldsymbol{\beta}}) = 0$ is typically nonlinear, in practice, we often tend to numerical methods such as the *Newton-Raphson algorithm* when solving for $\hat{\boldsymbol{\beta}}$. The Newton-Raphson method iteratively updates the estimates by the following scheme,

$$\hat{\boldsymbol{\beta}}^{(n+1)} \leftarrow \hat{\boldsymbol{\beta}}^{(n)} + \boldsymbol{H}^{-1}\left(\hat{\boldsymbol{\beta}}^{(n)}\right) \boldsymbol{s}\left(\hat{\boldsymbol{\beta}}^{(n)}\right),$$

   where $\boldsymbol{H}(\boldsymbol{\beta}) = -\frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^T}$ is the Hessian matrix representing the observed Fisher information. This iterative process is continued until convergence, which in practice means that the difference between successive estimates is controlled below a predefined threshold.

4. **Fisher Scoring Algorithm.** An alternative to the Newton-Raphson method is the *Fisher scoring algorithm*, where the Hessian matrix $\boldsymbol{H}(\boldsymbol{\beta})$ is replaced by the expected Fisher information $\boldsymbol{F}(\boldsymbol{\beta}) = \mathrm{E}[\boldsymbol{H}(\boldsymbol{\beta})]$. This algorithm also convergens under certain regularity conditions and is typically favored, as the expected Fisher information is usually easier to compute compared to the observed Fisher information. In cases where we use the *canonical link*, e.g. the logistic link for binomial regression, the expected and observed Fisher information is equivalent.

Convergence Criteria for Numerical Algorithms and Initial Estimates

Both of the algorithms given above, i.e. the Newton-Raphson and Fisher Scoring algorithm, require initial estimates $\hat{\boldsymbol{\beta}}^{(0)}$ to work. This is typically the case for most numerical methods, and we can find these initial estimates based on prior knowledge or estimate them using methods such as least squares. The iterations are usually stopped when the updates satisfy a convergence criterion, e.g.

$$\left\| \hat{\boldsymbol{\beta}}^{(n+1)} - \hat{\boldsymbol{\beta}}^{(n)} \right\|_2 \leq \epsilon,$$

where ϵ is a small positive constant, ensuring that the estimates are sufficiently close to the true maximum likelihood estimators.

Interpretation of Model Parameters

We continue our discussion of the coefficient estimates by fitting a logistic regression model to the dataset, where the number of successful ascents are regressed on the height and topographical prominence of the mountains. A summary of the model fit is provided below.

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

By fitting a logistic regression model to the dataset, we are assuming that the relationship between the *log-odds* of a successful ascent and the model covariates is explained by the relation

$$\ln\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_{\text{intercept}} + \beta_{\text{height}} \cdot \text{height}_i + \beta_{\text{prominence}} \cdot \text{prominence}_i.$$

Keeping this in mind, we make the following immediate observations based on the model summary,

- The **Intercept** is 13.69, which reflects the baseline log-odds of a successful ascent when both height and prominence are zero (though it does not have a practical interpretation in this context since height and prominence cannot be zero).
- The coefficient for **height** is -0.001635. This means that for every additional meter of height, the log-odds of successfully ascending some mountain decreases by approximately 0.001635.
- The coefficient for **prominence** is -1.74×10^{-4}. This means that for every additional meter of prominence, the log-odds of successfully ascending some mountain decreases by approximately 1.74×10^{-4}.

To interpret the coefficients in terms of *odds*, we start by computing the exponentials of the estimated coefficients. For a one-unit increase in any of the covariates, we would expect the odds of success to change by a factor equal to the exponential of the corresponding coefficient. Keeping this in mind, we make the following remarks,

- For each additional meter of **height**, the odds of successfully ascending some mountain decreases by a factor of 0.9984, or about 0.16 %. This is consistent with the idea that higher mountains are harder to climb.
- For each additional meter of **prominence**, the odds of successfully ascending some mountain decreases by a factor of 0.9998, or 0.02 %. Although the effect of prominence is small compared to that of height, it suggests that a greater prominence decreases the likelihood of success.

Significance of Model Parameters

We will now assess the significance of the model parameters using two common methods, the *Wald test* and the *Likelihood Ratio Test (LRT)*.

The Wald test is automatically performed as part of the `summary()` function when fitting a model using `glm()`. The summary provides the z-values and p-values for each parameter, which allows us to assess whether the coefficients are significantly different from zero.

- **Height**. The p-value for height is extremely small $p < 2 \cdot 10^{-16}$, which indicates that the height is a highly significant predictor of the probability of successfully ascending mountains.
- **Prominence**. The p-value for prominence is also small $p = 1.33 \cdot 10^{-4}$, indicating that prominence is a significant predictor as well. Still, the Wald test seems to indicate that the height is more significant for the success probability as compared to the prominence.

In addition to the Wald test, we can also perform a LRT to assess whether the model as a whole, including both predictors, provides a significantly better fit than the *null model*, i.e. a model that only includes the intercept. The LRT compares the deviance of the full model (with predictors) to that of the null model (without predictors). From the summary printout we find that,

- **Deviance of the full model**. The residual deviance of the model is 414.68 on 110 degrees of freedom. This deviance reflects the model's goodness-of-fit, with lower values indicating a better fit.
- **Null deviance**. The null deviance is 715.29 on 112 degrees of freedom, which corresponds to a model that only includes an intercept (no predictors).

The difference between the null deviance and the residual deviance shows how much better the model with predictors (height and prominence) fits the data. Using the `Anova()` function from the `car` library, we can carry out a LRT of individual predictors. This is done by comparing a model that includes the given predictor to a model without it.

```
## Analysis of Deviance Table (Type II tests)
##
## Response: cbind(success, fail)
##            LR Chisq Df Pr(>Chisq)
## height      139.765  1  < 2.2e-16 ***
## prominence   14.618  1  0.0001316 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We find that,

- The **LRT for height** results in a χ^2-value of 139.765 on 1 degree of freedom, which corresponds to a p-value of $p < 2.2 \times 10^{-16}$. This extremely small p-value indicates that height is a highly significant predictor, and the inclusion of height in the model significantly improves the model fit.
- The **LRT for prominence** results in a χ^2-value of 14.618 on 1 degree of freedom, which corresponds to a p-value of $p = 1.316 \times 10^{-4}$. This p-value shows that prominence is also a significant predictor, though its impact is smaller compared to that of height. The LRT thus conforms with the Wald test in this regard.

Confidence Interval for the Height Parameter

To create a 95 % confidence interval for the height parameter, we can use the standard error of the height coefficient and the corresponding critical value from the normal distribution.[1] The confidence interval can then be computed by solving

$$\hat{\beta}_{\text{height}} \pm z_{0.975} \cdot \text{SE}(\left(\hat{\beta}_{\text{height}}\right),$$

where $\hat{\beta}_{\text{height}}$ is the estimated coefficient for height and $\text{SE}\left(\hat{\beta}_{\text{height}}\right)$ its standard error. If we let $\hat{\beta}_L$ and $\hat{\beta}_H$ denote the lower and upper limits of such an interval, we can express the 95 % confidence interval as follows,

$$\left(\hat{\beta}_L, \hat{\beta}_H\right) \approx (-0.00191, -0.00136).$$

This interval suggests that, with 95% confidence, the true effect of height on the log-odds of a successful ascent lies between -0.00191 and -0.00136. Meaning that for each additional meter of height, the log-odds of successfully reaching the summit decreases by a value within this range, with 95% confidence. Since the entire interval is negative and does not include zero, we can conclude that height has a **statistically significant negative effect** on the probability of a successful ascent. In other words, higher mountains are associated with a lower likelihood of successful ascents.

Confidence Interval for the Effect of Height on the Odds of Success

To calculate the confidence interval for the effect that height has on the odds, we exponentiate the lower and upper limits of the previously calculated confidence interval. The resulting interval,

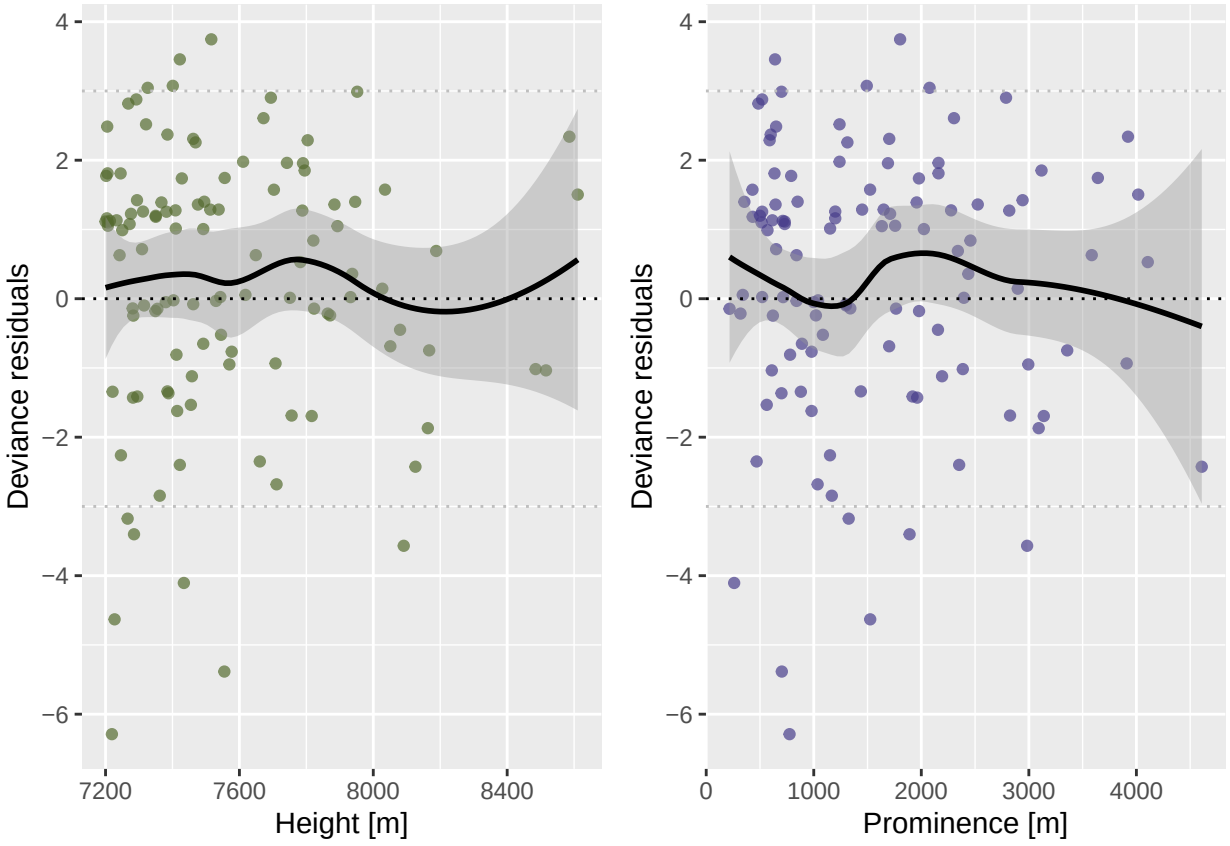
$$\left(\exp\left(\hat{\beta}_L\right), \exp\left(\hat{\beta}_H\right)\right) \approx (0.9981, 0.9986),$$

is a 95 % confidence interval for the multiplicative effect height has on the odds of a successful ascent. Meaning that for every additional meter of height, the odds of a successful ascent decreases by a factor between 0.9981 and 0.9981, with 95 % confidence. Since the entire confidence interval is less than one, we can conclude that height has a **statistically significant negative effect** on the odds of success. In practical terms, this implies that for each additional meter of height, the odds of a successful ascent decreases by approximately 0.14 % to 0.19 %. This supports the idea that taller mountains are more difficult to climb.

Visualising the Deviance Residuals as a Function of the Covariates

To further assess the model fit, we examine how the deviance residuals vary as a function of height and prominence. Deviance residuals provide a measure of how well the model predicts the observed values. If the model fits the data well, we expect the residuals to be scattered randomly around zero with a small magnitude. In a perfectly fitted model, also known as a *saturated model*, the observed responses would be perfectly predicted by the covariates, and the corresponding deviance residuals would equate to zero for all observations. Since our model is not saturated, residuals deviate from zero, but ideally, these deviations should not be excessively large, hence their magnitude should be relatively small. In the figure that follows, the residuals have been plotted against height (on the left) and prominence (on the right).

[1] For a 95 % confidence interval, the critical value is approximately 1.96.



In addition to the deviance residuals in the figure above, we have included some additional lines to ease the interpretation of the figure. The additional lines and the reasoning behind them are as follows,

- **Dotted lines at** 0. This indicates where the residuals would lie if the fit was perfect.
- **Dotted lines at** ± 3. This represents a common threshold for the residuals. Deviance residuals larger than 3 in magnitude suggest that the model may not be adequately capturing the data for these specific observations.[2] Residuals of such magnitude are typically found in the tails of a standard normal distribution, indicating that the model struggles to capture the relationship between the covariates and the outcome.
- **Solid lines.** These correspond to smoothed fits obtained by *locally weighted scatterplot smoothing (LOWESS)*, which show overall trends in the residuals. The **shaded area** around each LOWESS-line represents 95 % confidence intervals for these trends.

Interpretation of the Residual Plots

We make the following observations based on the residual plots,

- **Height (left plot)**. The deviance residuals fluctuate around zero, which is generally a good sign, indicating that the model fits the data reasonably well with respect to height. However, there is a slight upward trend for taller mountains (those above $8,000$ meters), suggesting that the model may slightly **underestimate** the probability of success for taller peaks. Most of the residuals are within the ± 3 range, indicating that the model performs adequately for most mountains with respect to height.

[2] Given that the model holds, the residuals are approximately standard normal, and so residuals with magnitude greater than 3 are in the tails of the standard normal distribution. Residuals such as these usually indicate strain in the model. Source: Ford, C. (2022). *Understanding Deviance Residuals.* University of Virginia Library

- **Prominence (right plot)**. The deviance residuals for prominence show slightly more spread than for height, with several residuals exceeding the ± 3 range. This suggests that the model might not fully capture the relationship between prominence and the probability of success. The residuals fluctuate somewhat around zero, but no clear systematic pattern emerges. The oscillations in the LOWESS smoother are minor, suggesting that the model's fit with respect to prominence is less precise than for height but still reasonable.

The overall pattern of the residuals indicates that the logistic regression model is a reasonable fit for the data, particularly for mountains with heights below $8,000$ meters and moderate prominences. However, certain taller mountains and those with extreme prominence show larger residuals, indicating the potential need for model refinement or the inclusion of additional predictors to better capture this part of the data.

Residual Deviance and Model Fit

In one of the previous sections, we briefly mentioned the residual deviance and null deviance given as part of the model summary. The residual deviance was found to be approximately equal to 414.68 on 110 degrees of freedom. Since the residual deviance follow a χ^2 distribution with 110 degrees of freedom, it is possible to find a p-value for the significance of regression, e.g. by using the `pchisq()` function. In this case, the p-value for regression, $p = 6.61 \times 10^{-37}$ is immensely small, indicating that the regression is highly significant. This conforms with the results from the previous sections, where both height and prominence were found to be significant predictors for the odds of success.

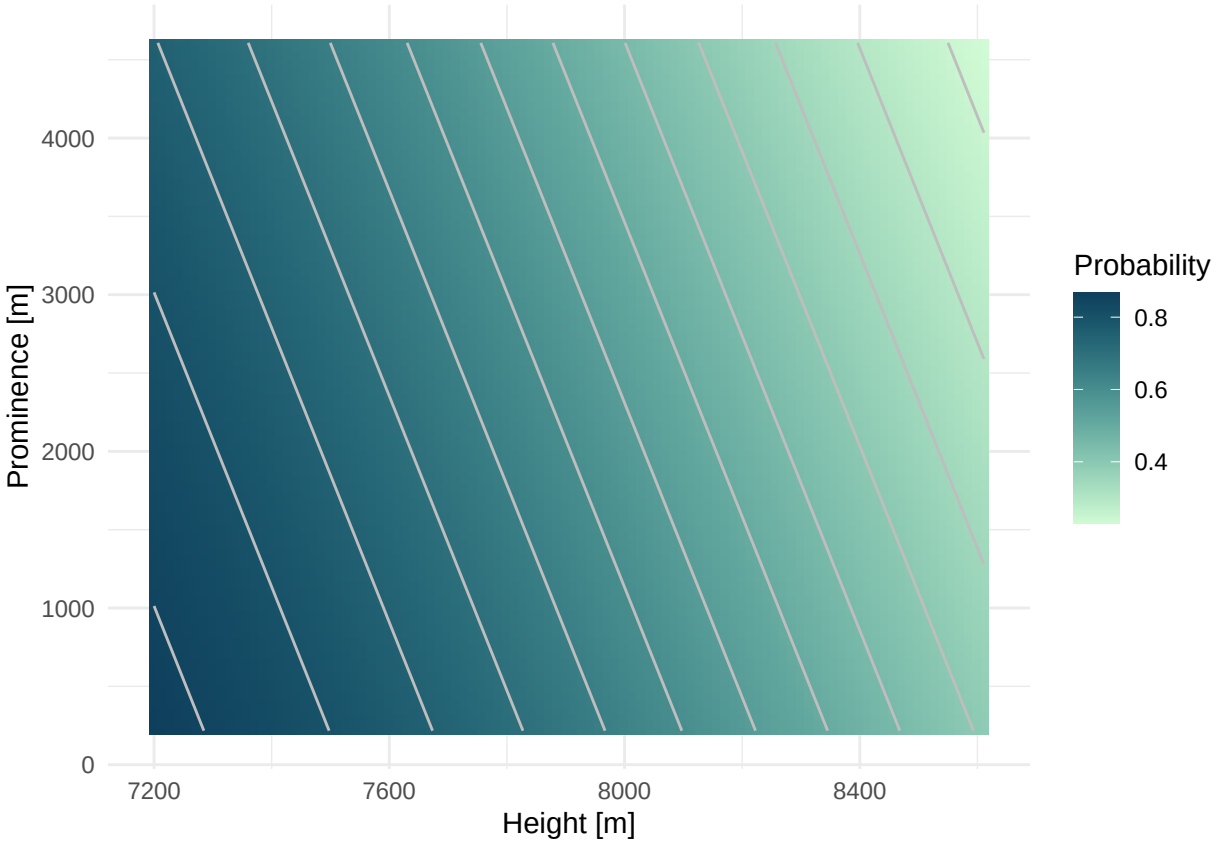
Furthermore, we found that the null deviance was approximately 715.29 on 112 degrees of freedom. Since the null deviance corresponds to a logistic regression model including only the intercept, we can use the null deviance as a measure of the total deviance in the dataset that can be explained by logistic regression. If we define the generalised coefficient of determination by the following relation,

$$R^2_{\text{GEN}} = 1 - \frac{\text{residual deviance}}{\text{null deviance}},$$

we get a statistic that indicates the "amount of deviance" in the dataset which is explained by the model, similarily to how the R^2 statistic indicates the degree to which a multiple linear regression model captures the available variability in a dataset. For our model, we find that $R^2_{\text{GEN}} = 42.03\ \%$, which indicates that the inclusion of predictors such as height and prominence explain a fair bit of the deviance present in the null model.

Estimating Probabilities as a Function of Height and Prominence

To visualize how the estimated probabilities of successful ascents vary as a function of height and prominence, we create a grid of height and prominence values within the data range, and then use the fitted logistic regression model to estimate the probabilities. The resulting plot is included below.



We make the following observations regarding the plot above,

- **Height**. The probability of a successful ascent **decreases** as the height increases, as shown by the shift from dark blue to light teal along the x-axis. This is expected since taller mountains are generally harder to climb, which matches the earlier results, where height was found to have a significant negative effect on the probability of success.
- **Prominence**. There is a similar effect with prominence, although it seems less pronounced compared to height. As the prominence increases (moving up on the y-axis), the probability decreases gradually, but the effect is smaller than for height. This matches with the previous results, where the effect of prominence was found to be statistically significant but smaller than the effect of height.
- **Contour Lines**. The contour lines highlight areas where the probability remains constant. We observe that the spacing between the contour lines is narrower towards the middle of the plot compared to the top right and bottom left of the plot. This indicates that the probability of success changes more rapidly in this region, i.e. for mountains with a height close to $8,000$ meters.[3] This is to be expected, since we are using a logistic link which samples data along a sigmoid.

Estimating the Probability of Success for Ascending Mount Everest and Chogolisa

We now want to estimate the probability of successfully ascending Mount Everest (height and prominence both equal to 8848 meters), and Chogolisa (height 7665 meters and prominence 1624 meters). In adition

[3]The partial pressure of oxygen in the atmosphere above 8000 m is known to be insufficient for sustaining human life over extended time periods. Mountaineers often refer to this region as the "death zone", so it makes physical sense that mountains above this level are substantially harder to ascend.

to the estimated probabilities of success, we want to construct confidence intervals for the predictions and comment on the validity of the estimates.

To construct the confidence intervals for the success probabilities, we first note that the maximum likelihood estimates $\hat{\boldsymbol{\beta}}$ have a know asymptotic distribution $\hat{\boldsymbol{\beta}} \overset{a}{\sim} \mathrm{N}\left(\boldsymbol{\beta}, \boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\right)$, where $\boldsymbol{F}$ denotes the expected Fisher information. When estimating the linear predictor $\eta_i := \boldsymbol{x}_i^T\boldsymbol{\beta}$ by means of maximum likelihood, the estimated predictor $\hat{\eta}_i$ becomes a linear combination of the asymptotic normal $\hat{\boldsymbol{\beta}}$, meaning that the estimated predictor $\hat{\eta}_i$ is itself asymptotic normal with mean and variance given by, $\hat{\eta}_i \overset{a}{\sim} \mathrm{N}\left(\boldsymbol{x}_i^T\boldsymbol{\beta}, \boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i\right)$. If we let $z_{1-\alpha/2}$ denote the critical value of the standard normal corresponding to a $1-\alpha$ confidence interval, a $1-\alpha$ confidence interval for η_i can then be computed as

$$\hat{\eta}_i \pm z_{1-\alpha/2}\sqrt{\boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i}.$$

Since the success probabilities are a function of the linear predictor η_i through the logistic link, corresponding confidence intervals for the estimated probabilities can be computed by a inverse transformation of the lower and upper limit of the above confidence interval. Furthermore, The inverse function of the logicistic link is known function of the form,

$$\pi_i = \frac{\exp(\eta_i)}{1+\exp(\eta_i)},$$

meaning that a $1-\alpha$ confidence interval for the probabilities $\hat{\pi}_i$ can be computed as

$$\left(\frac{\exp\left(\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}} - z_{1-\alpha/2}\sqrt{\boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i}\right)}{1+\exp\left(\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}} - z_{1-\alpha/2}\sqrt{\boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i}\right)}, \frac{\exp\left(\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}} + z_{1-\alpha/2}\sqrt{\boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i}\right)}{1+\exp\left(\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}} + z_{1-\alpha/2}\sqrt{\boldsymbol{x}_i^T\boldsymbol{F}^{-1}\left(\hat{\boldsymbol{\beta}}\right)\boldsymbol{x}_i}\right)}\right).$$

Mount Everest (Height = 8848, Prominence = 8848)

- **Estimated Probability of Success:** 0.0892 (approximately 8.92 %)
- 95 % **Confidence Interval for the Probability:** (0.0549, 0.1417)

The model predicts a relatively low probability of successfully ascending Mount Everest, which aligns with general expectations since Everest is one of the most challenging peaks to climb. The confidence interval suggests that, with 95 % confidence, the true probability of success lies between 5.49 % and 14.17 %. This result seems reasonable given the extreme height and prominence of Everest, both of which are known to make climbs particularly difficult.

Chogolisa (Height = 7665, Prominence = 1624)

- **Estimated Probability of Success:** 0.7043 (approximately 70.43 %)
- 95 % **Confidence Interval for the Probability:** (0.6832, 0.7245)

For Chogolisa, the model predicts a much higher probability of success, about 70.43 %, with a relatively narrow confidence interval. This higher probability makes sense in the context of the provided data, where there were 20 successes and 2 failures recorded on Chogolisa. Compared to Everest, Chogolisa has a lower height and prominence, both of which contribute to its higher success probability.

Is It Reasonable to Use This Model?

- **Mount Everest:** Using the model to estimate the probability of successfully ascending Mount Everest should be interpreted with caution. While the model captures some general trends related to height and prominence, it does not account for other critical factors specific to Mount Everest, such as extreme weather conditions, high traffic on the route, or the use of supplemental oxygen. Additionally, Everest is a statistical outlier in terms of both height and prominence, and the model may not generalize well to such extreme values.
- **Chogolisa:** For Chogolisa, the model's estimate seems more reasonable, as the data provided already includes outcomes from climbers attempting the ascent. The confidence interval is narrower, suggesting the model has more certainty in its prediction. Given Chogolisa's more moderate height and prominence compared to Everest, the model appears better suited for predicting success on mountains of this scale.

The logistic regression model provides reasonable estimates for the probability of success on both Mount Everest and Chogolisa, though it is important to recognize the limitations of the model when applied to extreme cases like Everest. For Chogolisa, the model seems to offer a more reliable prediction, but for more extreme peaks like Everest, additional factors beyond height and prominence likely play a crucial role in determining success.

Poisson Regression – Eliteserien 2023

We aim to build a Poisson regression model to predict football match outcomes in the 2023 Eliteserien season, using the number of goals scored by each team as the primary variable. The dataset contains match results where each row represents a match between a home team and an away team, along with their respective scores measured in the number of goals scored by each team. We only know the results from the first 191 matches, so the results from the remaining 49 matches have to estimated. We assume that the goals scored by the different teams are independent of each other, and account for the possibility of home team advantage by the introduction of a home advantage parameter β_{home}. If we denote the home team by A and the away team by B, the scores of each team is assumed to be Poisson distributed with mean λ, where

$$\ln(\lambda_{\text{A}}) = \beta_0 + \beta_{\text{home}} + \beta_{\text{A}} - \beta_{\text{B}} \quad \text{and} \quad \ln(\lambda_{\text{B}}) = \beta_0 - \beta_{\text{A}} + \beta_{\text{B}},$$

for some intercept β_0. We assume that both β_0 and β_{home} is equivalent for each team. The goal is to estimate the strength of each team based on the matches played so far, and ultimately, use this model to predict future match outcomes and the likelihood of a team becoming the champion.

Is the assumption of independence between the goals made by each team reasonable?

We want to assess whether the assumption of independence between the goals made by the home and away teams is reasonable. The Poisson regression model used to estimate the number of goals in a match assumes that the home and away goals are drawn from independent Poisson distributions. However, to verify this assumption, we conduct a *Pearson's χ^2 test* for independence between the goals scored by each team. This is done by cross tabulating the goals scored by the home and away teams during the 191 matches for which we have the scores, and then pasing the resulting contingency table to the in built `chisq.test()` function. The results are as follows,

- **Chi-squared statistic**: 20.98
- **Degrees of freedom (df)**: 36
- **p-value**: 0.9783

The null hypothesis H_0 for this test is that the home and away goals are independent. A large p-value (greater than the common significance level of 0.05) suggests that we do not have enough evidence to reject

the null hypothesis. In this case, the p-value is **0.9783**, which is well above 0.05. Therefore, we fail to reject the null hypothesis, implying that the goals scored by the home and away teams can be considered **independent**.

So what does this mean in practice?

This supports the assumption made in the Poisson regression model, where the home and away goals are modeled independently. We can now confidently proceed with using the Poisson regression model to predict outcomes for future matches in the Eliteserien, assuming the independence of home and away goals.

Preliminary team rankings

We now want to produce the preliminary ranking for the Eliteserien season as of October 1st, i.e. after the first 191 matches have been recorded. Points are awarded as follows,

- The winner of a match gets 3 points.
- The loser gets 0 points.
- In the case of a draw, both teams get 1 point each.

We also need to compute the goal difference for each team to break ties in case two teams have the same number of points. A printout of the preliminary ranking is included below, where we also include a column for the number of goals scored and the number of goals conceded.

```
## # A tibble: 16 x 5
##    team               total_points total_goals_scored total_goals_conceded
##    <chr>                     <dbl>              <int>                <int>
##  1 Bodø/Glimt                   53                 57                   21
##  2 Brann                        46                 42                   28
##  3 Molde                        44                 54                   28
##  4 Viking                       43                 46                   31
##  5 Rosenborg                    40                 40                   34
##  6 Fredrikstad                  40                 32                   29
##  7 KFUM                         33                 30                   29
##  8 HamKam                       29                 30                   29
##  9 Tromsø                       28                 28                   33
## 10 Strømsgodset                 28                 28                   37
## 11 Kristiansund                 26                 27                   36
## 12 Sarpsborg 08                 26                 33                   48
## 13 Haugesund                    23                 23                   37
## 14 Sandefjord Fotball           22                 31                   39
## 15 Odd                          22                 22                   40
## 16 Lillestrøm                   21                 26                   50
## # i 1 more variable: total_goal_difference <int>
```

The table reflects both the strengths and weaknesses of the teams, with Bodø/Glimt clearly leading the way in terms of points, goals scored, and defensive prowess. The mid-table is very competitive, with several teams closely matched in points. Lillestrøm is at the bottom of the table with 21 points, and their poor defensive record (50 goals conceded) highlights their struggles this season. The use of goal difference as a tiebreaker plays a crucial role in determining the standings, especially for teams that are tied on points.

Implementing Poisson Regression with Numerical Optimization

We now want to create a function that performs Poisson regression using numerical optimization to estimate the intercept, home advantage, and strength parameters for each team in the Eliteserien 2023 dataset. We will use the `optim()` function to find the maximum likelihood estimates for these parameters.

Steps:

1. Construct the design matrix X of size 384×18 (16 teams - 1 reference team + intercept and home advantage).
2. Define the negative log-likelihood function.
3. Optimize the parameters using `optim()` to find the maximum likelihood estimates.
4. Compare the ranking based on estimated strength parameters with the preliminary ranking.

Step 1: Creating the Design Matrix

We will create the design matrix X based on the information in the problem description. Each match contributes two rows to the matrix, one for the home team and one for the away team. We draw the effects of *Bodø/Glimt* into the intercept, the estimated strength parameters will therefore be in reference to the estimated strength parameter of *Bodø/Glimt*.

Step 2: Define the Negative Log-Likelihood Function

Next, we define the negative log-likelihood function, which will be minimized using `optim()` to find the Poisson regression parameters. The log-likelihood is based on the Poisson distribution with the log-link function:

$$\text{log-likelihood} = \sum (\text{goals} \cdot \log(\lambda) - \lambda)$$

where $\lambda = \exp(X \cdot \beta)$ is the Poisson mean, and β

Step 3: Perform Optimization with `optim()`

We use `optim()` to find the maximum likelihood estimates of the regression parameters. We initialize the parameters close to zeros and use the "BFGS" method for optimization. The function minimizes the negative log-likelihood defined in Step 2 and thus computes the strength coefficients for each team, i.e. λ_{Odd}, $\lambda_{\text{Fredrikstad}}$, ...

Step 4: Rank Teams Based on Estimated Strengths

Once we have the estimated parameters, we can rank the teams based on their strength parameters. The strength parameters indicate each team's relative strength in scoring goals, and teams with higher strength values are ranked higher. A printout of the ranking, as computed by `optim()`, is provided below alongside the computed strength parameters.

```
##                  team   strength
## 1          Bodø/Glimt  0.0000000
## 2               Molde -0.1353584
## 3              Viking -0.2518991
## 4               Brann -0.2922518
## 5           Rosenborg -0.4105967
## 6         Fredrikstad -0.4360857
## 7                KFUM -0.4415655
## 8              HamKam -0.4544676
## 9              Tromsø -0.5592119
## 10 Sandefjord Fotball -0.5999091
```

```
## 11       Kristiansund -0.6158672
## 12       Strømsgodset -0.6188305
## 13          Haugesund -0.6740761
## 14       Sarpsborg 08 -0.7130678
## 15                Odd -0.7457159
## 16         Lillestrøm -0.8171542
```

Correctness of the Numerical Algorithm

To assess whether the numerical optimization method using `optim()` converged to the correct estimates, we now compare the results given in the printout above to the estimators provided by the `glm()` function. The summary of the fitted model is provided below.

```
##
## Call:
## glm(formula = goals ~ -1 + X, family = "poisson")
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## XIntercept            0.23110    0.06433   3.592 0.000328 ***
## XHomeAdvantage        0.15532    0.08567   1.813 0.069851 .
## XOdd                 -0.74587    0.16551  -4.507 6.59e-06 ***
## XFredrikstad         -0.43610    0.16591  -2.629 0.008574 **
## XLillestrøm          -0.81707    0.16549  -4.937 7.92e-07 ***
## XMolde               -0.13541    0.16254  -0.833 0.404799
## XRosenborg           -0.41061    0.16580  -2.477 0.013265 *
## XTromsø              -0.55920    0.16603  -3.368 0.000757 ***
## XViking              -0.25181    0.16148  -1.559 0.118901
## XKFUM                -0.44152    0.16273  -2.713 0.006663 **
## XKristiansund        -0.61584    0.16401  -3.755 0.000173 ***
## XHamKam              -0.45457    0.16223  -2.802 0.005080 **
## XHaugesund           -0.67406    0.16203  -4.160 3.18e-05 ***
## XSarpsborg 08        -0.71304    0.16192  -4.404 1.06e-05 ***
## XStrømsgodset        -0.61881    0.16204  -3.819 0.000134 ***
## XBrann               -0.29222    0.16107  -1.814 0.069650 .
## XSandefjord Fotball -0.59992    0.16403  -3.657 0.000255 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 549.63  on 382  degrees of freedom
## Residual deviance: 426.61  on 365  degrees of freedom
## AIC: 1161.4
##
## Number of Fisher Scoring iterations: 5
```

In the summary for the fitted model, the estimated coefficient for e.g. *XViking* corresponds to the strength parameter for Viking and so on. As for the ranking provided my the manual optimization, the estimates from the GLM is to be understood relative to the strength parameter of Bodø/Glimt. We observe that differences in the strength parameters are quite small, most if not all parameters are equal up to the 4th decimal point. This indicates that the estimates provided by the numerical algorithm are likely correct and unbiased, at least in the sence of maximum likelihood.

Comparison with the Preliminary Ranking

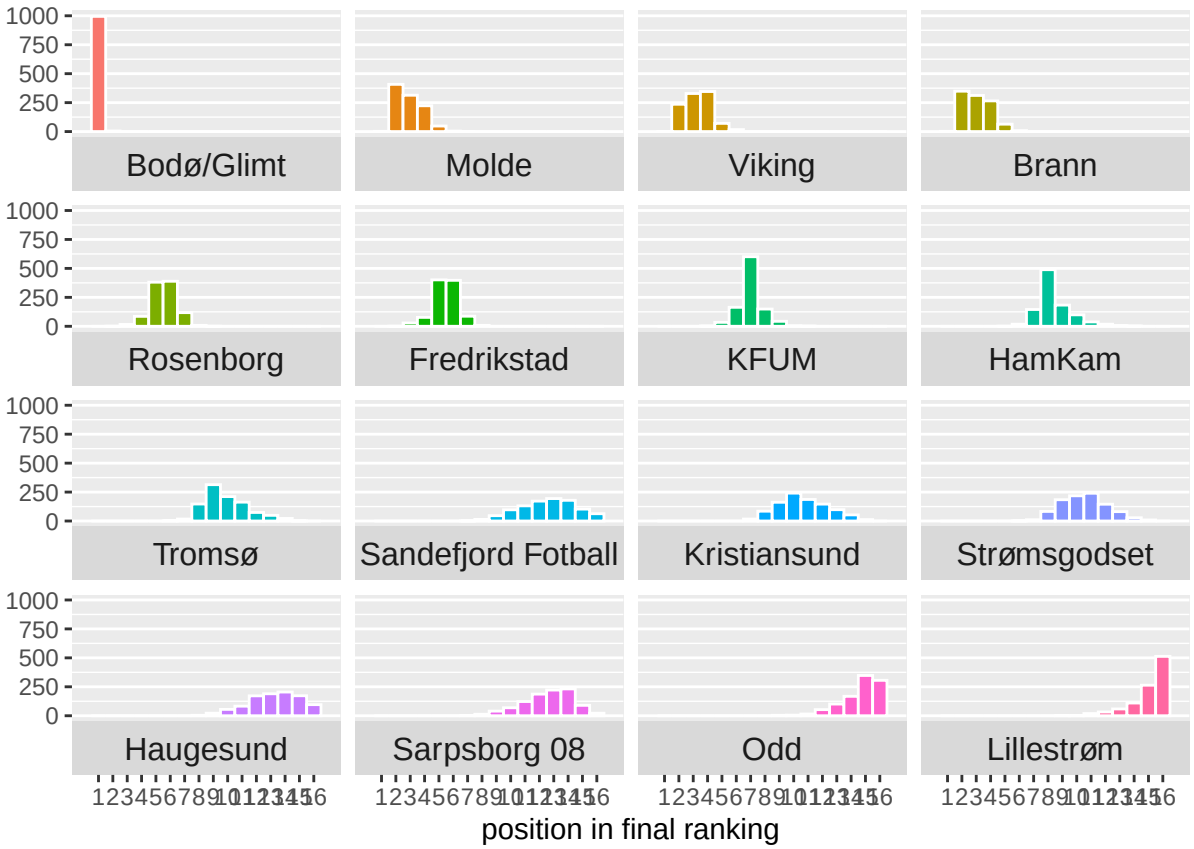
Now that we know that the computed rankings are likely correct in the sense of maximum likelihood, we want to compare the predicted ranking with the preliminary ranking. Since the predicted ranking is based on the strength parameter of each team, we would expect that high ranking teams score more goals as compared to the lower ranking teams. This is supported by the information given in the printout of the preliminary ranking, where we observe that best scoring teams, both in terms of points and amount of goals scored, is with a few exceptions predicted to be the strongest teams in the 2023 season.

Some teams also seem to have had a lucky start at the 2023 season, by being placed higher on the preliminary ranking than they are on the predicted ranking. Teams which might consider themselves lucky include Brann and Sandefjord Fotball, which are placed 2 and 4 places higher in the preliminary ranking, while teams such as Strømsgodset are placed 2 places lower in the preliminary ranking as compared to the predicted ranking. It seems reasonable that most teams occupy the same space on the predicted leader board as they do on the preliminary ranking, since most of the matches of this season has already been played. We might have seen more differences in the rankings given that the season had just started.

In both rankings, Bodø/Glimt seems to be dominating the other teams by quite a margin. Bodø/Glimt might therefore be the safest bet as to who will win Eliteserien in the 2023 season.

Simulating the Remaining Matches by Means of Monte Carlo

We are now going to simulate the reaming 49 matches by using the estimated strengths of each team to make 1000 realisations of the number of goals scored by each team during the remaining matches. The number of goals are then appended to the preliminary summary, to get the simulated number of goals for each team during the 2023 season. A histogram of the simulation, where the x-axis denotes the position of the team in the final ranking of the season, is provided below.



Observations and insights from the Simulations

Bodø/Glimt Dominates!

- The histogram for Bodø/Glimt shows a distinct peak at the top position, indicating that they are the most likely team to win the league in the majority of the simulated seasons. The distribution is highly skewed towards first place.
- The statistical summary shows that Bodø/Glimt has a mean rank of 1.005 with a very low variance (0.00498) and a win proportion of 99.5 %. This highlights their dominance, as they consistently finish in the top position across simulations.

Molde, Viking and Brann are not doing too bad - but lack consistency

Molde, Viking, and Brann also show a tendency to finish in higher-ranking positions, but their distributions are more spread out compared to Bodø/Glimt. - Molde, for example, has a mean rank of 2.923 with a variance of 0.878, suggesting that while they are competitive, they are less consistent compared to Bodø/Glimt. Viking and Brann similarly show competitive rankings but with greater variability.

Mid-Table - do not put your money here - huge variability!

- Teams such as Kristiansund, Sandefjord Fotball, and Strømsgodset display distributions that are centered around the middle to lower ranks. These distributions show significant variance, indicating that their final ranking is highly uncertain and can vary widely depending on game outcomes.
- For example, Kristiansund has a mean rank of 10.661 with a variance of 3.021, indicating significant variability in their performance.

Lower-Table - there is no hope

- Teams like Haugesund, Sarpsborg 08, Odd and Lillestrøm show a notable concentration towards the bottom of the rankings, indicating a higher probability of finishing among the last few positions. Their distributions suggest that they struggle to compete against stronger teams in the league.
- Lillestrøm, for instance, has a mean rank of 15.099 with a last place proportion of 51.3 %, indicating a significant likelihood of finishing at the bottom.

Uncertainty and Variability

- The variability in the distributions is an important indicator of the uncertainty inherent in football results. The histograms demonstrate that while some teams (such as Bodø/Glimt) have a high likelihood of finishing at the top, others are much more affected by randomness and fluctuations in match outcomes.

You can find all the statistics in the summary table printed below.

```
## # A tibble: 16 x 8
##    team         mean_rank variance_rank sd_rank ci_lower ci_upper win_proportion
##    <chr>            <dbl>         <dbl>   <dbl>    <dbl>    <dbl>          <dbl>
##  1 Bodø/Glimt        1.00       0.00498  0.0706        1        1          0.995
##  2 Brann             3.06       0.947    0.973         2        5          0.002
##  3 Fredrikstad       5.46       0.811    0.901         3        7          0
##  4 HamKam            8.43       1.51     1.23          7       12          0
##  5 Haugesund        13.2        3.16     1.78          9       16          0
```

```
##  6 KFUM              7.01       0.717    0.846         5        9          0
##  7 Kristiansund     10.7        3.02     1.74          8       14          0
##  8 Lillestrøm       15.1        1.51     1.23         12       16          0
##  9 Molde             2.92       0.878    0.937         2        5          0.002
## 10 Odd              14.7        1.76     1.32         12       16          0
## 11 Rosenborg         5.53       0.826    0.909         4        7          0
## 12 Sandefjord ~     12.6        3.68     1.92          9       16          0
## 13 Sarpsborg 08     12.6        2.97     1.72          9       15          0
## 14 Strømsgodset     10.5        2.56     1.60          8       14          0
## 15 Tromsø            9.88       2.35     1.53          8       13          0
## 16 Viking            3.30       0.933    0.966         2        5          0.001
## # i 1 more variable: last_place_proportion <dbl>
```

Code for Problem 1

```r
#------------#
# Problem 1b #
#------------#

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

# compute the odds
odds_ratios <- exp(coef(model))
odds_ratios

# compute the likelihood ratio test
lrt_results <- car::Anova(model, test = "LR")
lrt_results

# compute confidence intervals
height_coef <- coef(model)["height"]
height_se <- summary(model)$coefficients["height", "Std. Error"]
# ...for regression coefficients
ci_height <- c(height_coef - 1.96 * height_se, height_coef + 1.96 * height_se)
ci_height
# ...for effects on the odds
ci_odds_height <- exp(ci_height)
ci_odds_height



#------------#
# Problem 1c #
#------------#
```

```r
# extract deviance residuals
res <- residuals(model, type = 'deviance')
# create left residual plot
res_height <- ggplot(data = mount, mapping = aes(x = height, y = res)) +
  geom_point(alpha = 0.7, color = 'darkolivegreen') +
  geom_smooth(method = 'loess', color = 'black', linetype = 'solid') +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  geom_hline(yintercept = c(3, -3), linetype = 'dotted', color = 'grey') +
  labs(x = 'Height [m]', y = 'Deviance residuals')
# create right residual plot
res_prominence <- ggplot(data = mount, mapping = aes(x = prominence, y = res)) +
  geom_point(alpha = 0.7, color = 'darkslateblue') +
  geom_smooth(method = 'loess', color = 'black', linetype = 'solid') +
  geom_hline(yintercept = 0, linetype = 'dotted') +
  geom_hline(yintercept = c(3, -3), linetype = 'dotted', color = 'grey') +
  labs(x = 'Prominence [m]', y = 'Deviance residuals')
# plot figure
plot_grid(res_height, res_prominence, align = 'v')


# compute p-value for regression
p_value_regression <- pchisq(model$deviance, model$df.residual,
                             lower.tail = FALSE)

# extract the residual deviance and null deviance from the model
dev_model <- model$deviance
dev_null <- model$null.deviance
# compute generalized R-squared
R_squared_gen <- 1 - dev_model / dev_null
# print results
cat("Generalized R-squared: ", round(R_squared_gen, 4), "\n")

# compute and plot probability plot
height_range <- seq(min(mount$height), max(mount$height), length.out = 100)
prominence_range <- seq(min(mount$prominence), max(mount$prominence),
                        length.out = 100)

grid <- expand.grid(height = height_range, prominence = prominence_range)

grid$probability <- predict(model, newdata = grid, type = "response")

ggplot(grid, aes(x = height, y = prominence, z = probability)) +
  geom_raster(aes(fill = probability)) +
  geom_contour(colour = "grey") +
  scale_fill_gradientn(colours = rev(hcl.colors(n=10, palette = 'Dark Mint'))) +
  labs(x = "Height [m]", y = "Prominence [m]", fill = "Probability") +
  theme_minimal()



#------------#
# Problem 1d #
#------------#
```

```r
# Fisher information
vcov_matrix <- vcov(model)

# dataframe Everest
everest <- data.frame(height = 8848, prominence = 8848)

# compute predictions for Everest
eta_everest <- predict(model, newdata = everest, type = "link")
prob_everest <- predict(model, newdata = everest, type = "response")

# coefficients and standard error for Everest
x_everest <- c(1, everest$height, everest$prominence)
se_eta_everest <- sqrt(t(x_everest) %*% vcov_matrix %*% x_everest)

# confidence interval for linear predictor
ci_eta_everest <- c(eta_everest - 1.96 * se_eta_everest,
                    eta_everest + 1.96 * se_eta_everest)

# confidence interval for probability
ci_prob_everest <- exp(ci_eta_everest) / (1 + exp(ci_eta_everest))

# print results
cat("Estimated probability of success for Mount Everest:", round(prob_everest, 4), "\n")
cat("95% confidence interval for the probability:", round(ci_prob_everest, 4), "\n")

# dataframe Chogolisa
chogolisa <- data.frame(height = 7665, prominence = 1624)

# compute predictions for Chogolisa
eta_chogolisa <- predict(model, newdata = chogolisa, type = "link")
prob_chogolisa <- predict(model, newdata = chogolisa, type = "response")

# coefficients and standard error for Chogolisa
x_chogolisa <- c(1, chogolisa$height, chogolisa$prominence)
se_eta_chogolisa <- sqrt(t(x_chogolisa) %*% vcov_matrix %*% x_chogolisa)

# confidence interval for linear predictor
ci_eta_chogolisa <- c(eta_chogolisa - 1.96 * se_eta_chogolisa,
                      eta_chogolisa + 1.96 * se_eta_chogolisa)

# confidence interval for probability
ci_prob_chogolisa <- exp(ci_eta_chogolisa) / (1 + exp(ci_eta_chogolisa))

# print results
cat("Estimated probability of success for Chogolisa:", round(prob_chogolisa, 4), "\n")
cat("95% confidence interval for the probability:", round(ci_prob_chogolisa, 4), "\n")
```

Code for Problem 2

```r
#------------#
# Problem 2a #
```

```r
#------------#

# load the dataset
filepath <- "https://www.math.ntnu.no/emner/TMA4315/2024h/eliteserien2024.csv"
eliteserie <- read.csv(file = filepath)

# print the dataframe
head(eliteserie)

# remove matches that has not jet been played
eliteserie_clean <- eliteserie %>% filter(!is.na(yh) & !is.na(ya))

# contingency table
goal_table <- table(eliteserie_clean$yh, eliteserie_clean$ya)

# Pearson's chi_squared test
chi_test <- chisq.test(goal_table)
chi_test



#------------#
# Problem 2a #
#------------#

# compute preliminary rankings
team_stats <- eliteserie %>%
  mutate(home_goal_diff = yh - ya,
         away_goal_diff = ya - yh) %>%
  pivot_longer(cols = c(home, away),
               names_to = "location",
               values_to = "team") %>%
  mutate(points = case_when(
           location == "home" & yh > ya ~ 3,
           location == "away" & ya > yh ~ 3,
           yh == ya ~ 1,
           TRUE ~ 0),
         goals_scored = ifelse(location == "home", yh, ya),
         goals_conceded = ifelse(location == "home", ya, yh),
         goal_diff = goals_scored - goals_conceded) %>%
  group_by(team) %>%
  summarise(total_points = sum(points),
            total_goals_scored = sum(goals_scored, na.rm = TRUE),
            total_goals_conceded = sum(goals_conceded, na.rm = TRUE),
            total_goal_difference = sum(goal_diff, na.rm = TRUE)) %>%
  arrange(desc(total_points), desc(total_goal_difference), desc(total_goals_scored))

# show the dataframe
team_stats



#------------#
```

```r
# Problem 2c #
#------------#

# Home and away goals combined into one vector
goals <- c(eliteserie_clean$yh, eliteserie_clean$ya)

teams <- unique(c(eliteserie_clean$home, eliteserie_clean$away))
teams <- teams[teams != "Bodø/Glimt"]

n_games <- nrow(eliteserie_clean)
n_teams <- length(teams)
X <- matrix(0, nrow = 2 * n_games, ncol = n_teams + 2)  # 15 teams + intercept + home advantage
colnames(X) <- c("Intercept", "HomeAdvantage", teams)

for (i in 1:n_games) {
  home_team <- eliteserie_clean$home[i]
  away_team <- eliteserie_clean$away[i]

  X[i, "Intercept"] <- 1
  X[i, "HomeAdvantage"] <- 1
  if (home_team != "Bodø/Glimt") {
    X[i, home_team] <- 1
  }
  if (away_team != "Bodø/Glimt") {
    X[i, away_team] <- -1
  }

  X[n_games + i, "Intercept"] <- 1
  X[n_games + i, "HomeAdvantage"] <- 0
  if (home_team != "Bodø/Glimt") {
    X[n_games + i, home_team] <- -1
  }
  if (away_team != "Bodø/Glimt") {
    X[n_games + i, away_team] <- 1
  }
}

X_df <- as.data.frame(X)

X_df$goals <- goals

head(X_df)

# Define the negative log-likelihood function for Poisson regression
neg_log_likelihood <- function(beta, X, goals) {
  lambda <- exp(X %*% beta)

  # Avoid log(0) by adding a small constant to lambda
  lambda <- pmax(lambda, 1e-10)

  if (any(!is.finite(lambda))) {
    stop("Non-finite lambda values encountered!")
  }
```

```r
  log_likelihood <- sum(goals * log(lambda) - lambda)

  return(-log_likelihood)
}

neg_log_likelihood(rep(0.1, ncol(X)), X, goals)

estimate_parameters <- function(
    initial_estimate,  # Initial parameter values
    object_function,   # The function to minimize (negative log-likelihood)
    X,                 # Design matrix (passed as argument)
    responses,         # Response vector
    method = 'BFGS',   # Optimization method
    hessian = TRUE     # Return Hessian matrix for covariance estimation
) {

  # compute estimates
  result <- optim(
    par = initial_estimate,
    fn = object_function,
    X = X,             # (passed as argument)
    goals = responses, # (passed as argument)
    method = method,
    hessian = hessian
  )

  # check for convergence
  if (result$convergence != 0) {stop("Optimization did not converge!")}

  # estimated coefficients
  estimated_beta <- result$par
  names(estimated_beta) <- c("Intercept", "Home Advantage", teams)

  # data frame of coefficients
  beta_df <- as.data.frame(estimated_beta)
  beta_df$Parameter <- rownames(beta_df)
  rownames(beta_df) <- NULL
  beta_df <- beta_df[, c("Parameter", "estimated_beta")]

  return(beta_df)
}

initial_beta <- rep(0.1, ncol(X))

beta_df <- estimate_parameters(
  initial_estimate = initial_beta,
  object_function = neg_log_likelihood,
  X = X,
  responses = goals
)

print(beta_df)
```

```r
team_strengths <- beta_df[-c(1, 2), 2]

team_ranking <- data.frame(
  team = teams,
  strength = team_strengths
)

# Add Bodø/Glimt with a strength of 0 (reference team)
team_ranking <- rbind(team_ranking, data.frame(team = "Bodø/Glimt", strength = 0))

# Rank teams based on strength (higher strength = higher rank)
team_ranking <- team_ranking %>%
  arrange(desc(strength))

team_ranking

summary(glm(goals ~ -1 + X, family = "poisson"))



#------------#
# Problem 2d #
#------------#

n_games <- nrow(eliteserie_clean)
n_teams <- length(teams)

# remaining matches, which have to be simulated
unplayed_matches <- eliteserie[is.na(eliteserie$yh), c("home", "away")]

# relevant parameters
num_simulated_realisations <- 1000
num_remaining_matches <- nrow(unplayed_matches)

# design matrix for the remaining matches
X_simulation <- matrix(0, nrow = 2 * num_remaining_matches, ncol = n_teams + 2)
colnames(X_simulation) <- c("Intercept", "HomeAdvantage", teams)

for (i in 1:num_remaining_matches) {
  home_team <- unplayed_matches$home[i]
  away_team <- unplayed_matches$away[i]

  X_simulation[i, "Intercept"] <- 1
  X_simulation[i, "HomeAdvantage"] <- 1
  if (home_team != "Bodø/Glimt") {
    X_simulation[i, home_team] <- 1
  }
  if (away_team != "Bodø/Glimt") {
    X_simulation[i, away_team] <- -1
  }

  X_simulation[num_remaining_matches + i, "Intercept"] <- 1
  X_simulation[num_remaining_matches + i, "HomeAdvantage"] <- 0
```

```
  if (home_team != "Bodø/Glimt") {
    X_simulation[num_remaining_matches + i, home_team] <- -1
  }
  if (away_team != "Bodø/Glimt") {
    X_simulation[num_remaining_matches + i, away_team] <- 1
  }
}

# matrix to hold the results of the simulation
simulation_results <- matrix(
  0,
  nrow = n_teams + 1,
  ncol = num_simulated_realisations
)
rownames(simulation_results) <- team_ranking$team

# seed for reproducibility
set.seed(2)

# Monte Carlo algorithm
for(realisation in 1:num_simulated_realisations){

  # define a dataframe for the realisation
  df_realisation <- data.frame(
    unplayed_matches$home,
    unplayed_matches$away,
    matrix(0, nrow = num_remaining_matches, ncol = 1),
    matrix(0, nrow = num_remaining_matches, ncol = 1)
  )
  colnames(df_realisation) <- colnames(eliteserie)[-1]

  # simulate the remaining matches
  for(match in 1:num_remaining_matches){
    df_realisation$yh[match] <- rpois(
      n = 1,
      lambda = exp(t(X_simulation[match, ]) %*% beta_df$estimated_beta)
    )
    df_realisation$ya[match] <- rpois(
      n = 1,
      lambda = exp(t(X_simulation[match + num_remaining_matches,]) %*%
                     beta_df$estimated_beta)
    )
  }

  # add the simulated matches to the earlier matches and compute the
  # rankings for the realisation
  simulated_team_stats <- rbind(eliteserie_clean[,-1], df_realisation) %>%
  mutate(home_goal_diff = yh - ya,
         away_goal_diff = ya - yh) %>%
  pivot_longer(cols = c(home, away),
               names_to = "location",
               values_to = "team") %>%
  mutate(points = case_when(
```

```
            location == "home" & yh > ya ~ 3,
            location == "away" & ya > yh ~ 3,
            yh == ya ~ 1,
            TRUE ~ 0),
          goals_scored = ifelse(location == "home", yh, ya),
          goals_conceded = ifelse(location == "home", ya, yh),
          goal_diff = goals_scored - goals_conceded) %>%
  group_by(team) %>%
  summarise(total_points = sum(points),
            total_goals_scored = sum(goals_scored, na.rm = TRUE),
            total_goals_conceded = sum(goals_conceded, na.rm = TRUE),
            total_goal_difference = sum(goal_diff, na.rm = TRUE)) %>%
  arrange(desc(total_points), desc(total_goal_difference),
          desc(total_goals_scored))

  # record the simulated ranking
  for(team in 1:16){
    simulation_results[simulated_team_stats[[team, 1]], realisation] = team
  }
}

# make a nice plot of the rankings
simulation_plot <- ggplot(data = melt(simulation_results),
                          aes(x = value, fill = as.factor(Var1))) +
  facet_wrap( ~ as.factor(Var1), ncol = 4, strip.position = 'bottom') +
  guides(fill = 'none') +
  geom_histogram(binwidth = 1, center = 0, col="white") +
  theme(panel.grid.minor.x = element_blank(),
        panel.grid.major.x = element_blank()) +
  theme(strip.text = element_text(size = 12)) +
  scale_x_continuous(breaks = 1:16, limits = c(0,17)) +
  labs(x = 'position in final ranking', y = '')

simulation_plot

simulation_results_df <- as.data.frame(t(simulation_results))
colnames(simulation_results_df) <- rownames(simulation_results)

team_statistics <- simulation_results_df %>%
  pivot_longer(cols = everything(), names_to = "team", values_to = "rank") %>%
  group_by(team) %>%
  summarise(
    mean_rank = mean(rank),
    variance_rank = var(rank),
    sd_rank = sd(rank),
    ci_lower = quantile(rank, probs = 0.025),
    ci_upper = quantile(rank, probs = 0.975),
    win_proportion = mean(rank == 1),
    last_place_proportion = mean(rank == max(rank))
  )

print(team_statistics)
```