Part 1: Logistic regression (4 points)

Part 2: Poisson regression – Eliteserien 2023 (6 points)

TMA4315 Generalized Linear Models

Compulsory exercise 2: Logistic regression and Poisson regression

Deadline: Friday, October 11, 2024 at midnight

To be handed in on Blackboard.

Students may work on the assignment alone, or in groups of two or three, and are also encouraged to collaborate across groups. You do not have to use the same group as for the first exercise. Each group needs to hand in an [R Markdown] (http://rmarkdown.rstudio.com/ (http://rmarkdown.rstudio.com/)) document with answers to the questions as well as all source code. You find a template here: https://www.math.ntnu.no/emner/TMA4315/2018h/template_glm.Rmd (https://www.math.ntnu.no/emner/TMA4315/2018h/template_glm.Rmd). Hand in the file as a pdf file (`.pdf`) only, no zipped folders necessary. Include *all* code you have written for necessary functions in the end of your file (hint: use `eval = FALSE` in the chunk option) (only include the final version).

You shall deliver a document that answers all questions, using print-outs from `R` when asked for/when it seems necessary. It should include calculations, theory and fomulas you needed to answer the questions, and enough information so the teaching assistant can see that you have understood the theory (but do not write too much either, and exclude code not directly necessary for the answers). You will often be asked to interpret the parameters. This does not mean that you should just say what the numeric values are or state how you interpret the parameter in general, but rather explain what these values mean for the model in question.

Please write the names of the group members and the group number you have on Blackboard on top of your document!

In this exercise you shall first use the built-in `glm` -package to perform logistic regression. Then you must create a function that calculates the regression coefficients for a Poisson regression (you do **not** have to create a `myglm` -package, as only the estimates of $\beta\beta$ are needed, but a framework is provided for those who want to look at it).

Part 1: Logistic regression (4 points)

Use the built-in `R` -function `glm` for this part!

Wikipedia's *List of highest mountains* (https://en.wikipedia.org/wiki/List_of_highest_mountains_on_Earth (https://en.wikipedia.org/wiki/List_of_highest_mountains_on_Earth)) lists 118 of the world's highest mountains, along with some properties of each one, including the number of successful and failed attempts at reaching the summit as of 2004. In this problem, we will consider a data set consisting of the height (in meters), topographic prominence (also in meters), number of successful ascents and number of failed attempts for 113 of the mountains on the list. The mountains Mount Everest (height 8848, prominence 8848), Muztagh Ata, Ismoil Somoni Peak, and Jengish Chokusu/Tömür/Pk Pobeda are excluded from the dataset because of incomplete data. In addition the mountain Chogolisa (height 7665, prominence 1624) is removed from the data set to be used for prediction.

In the following, let y_iy_i be the number of successful ascents, and let n_in_i be the total number of attempts (sum of successful and failed) of the ith mountain. Use a binary regression with logit link to model the probability of success. That is,

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

where $\mathbf{x}_i$ is p dimensional column vector of covariates for observation i, and is the vector of regression parameters.

a) (1 point)

- Write down the log-likelihood function $\ell(\ )$ for this model (i.e., as a function of).
- Explain in a few words what we do in practice to arrive at maximum likelihood estimates for the parameters β.

b) (1 point)

Load the dataset:

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

Note that some entries differs from the Wikipedia list.

Fit the model, with height and prominence as predictors. Hint: To fit this model using the `glm` function, you write

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

- Interpret the model parameters (hint: odds).
- Discuss their significance (hint: Wald, LRT or score test).
- Create a 95 % confidence interval for the height parameter. Let $(\hat{\beta}_L, \hat{\beta}_H)$ denote the limits of your interval.
- Calculate the confidence interval $(\exp(\hat{\beta}_L), \exp(\hat{\beta}_H))$, and interpret.

c) (1 point)

For each of the covariates (`height` and `prominence`), plot the deviance residuals (hint: use the function `residuals` with argument `type = "deviance"`) as a function of the covariate. Describe what you see.
Then use the model deviance to assess the model fit.

Now you shall plot the estimated probabilities as a function of both height and prominence. The procedure will be similar to what we did in the interactive lecture in the second week of module 3. Only include estimated probabilities *inside* the data range. Comment briefly on the plot. (Hints for `ggplot` : Use `geom_raster` . The functions `geom_contour` and `scale_fill_gradientn(colours = terrain.colors(10))` can make this graph easier to interpret.)

d) (1 point)

The height and prominence of Mount Everest are both equal to 8848 meters. Based on the fitted model, estimate the probability of successfully ascending Mount Everest, and construct a confidence interval for the probability (hint: use the asympotic distribution of the MLE to find the asymptotic distribution of the linear predictor, then make a confidence interval for the linear predictor, finally transform this interval (i.e., the lower and upper limits of the interval) to get an interval for the probability of sucess). Write down the mathematical formula for this confidence interval.
Is it reasonable to use this model to estimate the probability of ascending Mount Everest? Why/why not?
Do the same with Chogolisa (height 7665 and prominence 1624). The data from Chogolisa gave 2 failures and 20 successes. Comment briefly on the result.

Part 2: Poisson regression – Eliteserien 2023 (6 points)

In Compulsory exercise 1 you made your own `mylm` package to handle Gaussian responses. In this exercise you will create a *function* (not a full package) to estimate regression parameters in a Poisson regression. The main difference from before is that it is now necessary to perform numerical optimization to find the parameter estimates. You can create the function in any way you want. However, you may of course make an `R` package with only one function if you want, and then a framework can be found here: https://www.math.ntnu.no/emner/TMA4315/2018h/myglm.R (https://www.math.ntnu.no/emner/TMA4315/2018h/myglm.R). You do *not* have to create print, summary, plot or anova this time, only the `myglm` function is necessary.

Introduction

If you choose to create and build a package, follow the instructions from Exercise 1, but now use the `myglm.R` file from the link above.

Here are some tips and suggestions that can make the implementation of the function easier:

1. Use the `R` function `optim` to find the maximum likelihood estimates for . Note that `optim` minimizes by default, but you can either use the negative log-likelihood, or read under *Details* in the documentation for `optim` . You must create an `R` function for the log-likelihood and provide it to `optim` (the function can be included in the same file as the code for the package). Note that you must choose the correct optimizing method `optim` should use. Hint: we did something similar in Module 3, interactive session in week 1. Note that you must choose another optimization method than the default Nelder-Mead (again, see the documentation for `optim`)
2. If you use `hessian = TRUE` in `optim` it will return the Hessian at the maximum likelihood estimate. Calculate the estimated covariance matrix based on this Hessian (but you will not actually need these values, you only need the estimates of , but it is cool to know).
3. You might want to implement a function that evaluates derivatives of the (log)likelihood and provide it to `optim` if you want to get more accurate results (this requires that you use an optimization method that uses the gradient, see documentation for `optim`).

Load the data set in R:

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

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

The data set consists of four columns:

- `home` : the name of the home team
- `away` : the name of the away team
- `yh` : the score of the home team
- `ya` : the score of the away team

The data set contains results from all played football matches of the 2023 (Norwegian) Eliteserien. Each row of the dataset corresponds to one played match, where the home team played on their home turf against the away team, who was visiting. Each team in the league faces all other teams twice during the season. Once as the home team, and once as the away team. Hence, a league contested by n teams will consist of $n(n-1)$ matches. In our case, $n = 16$, so there will be played 240 matches during the season.

As of the afternoon of the 1st of October, the day the data were collected, 191 matches have been played in this season, and each team have played between 23 and 23 matches. Note that the number of home matches and away matches vary between the teams as the season is not finished. *Do NOT update the dataset during the season when more games are played!* See the Wikipedia article on Tippeligaen 2023 (https://en.wikipedia.org/wiki/2023_Eliteserien) for information on the 2023 season.

Your task is to predict the winner of the season based on the scores so far, and to study the uncertainty of the ranking so far in the season. Specifically, for the 2023 Eliteserien, you should be able to give some answer to the question: *How likely or unlikely are Rosenborg to become champions?* You should begin by reading this (https://www.math.ntnu.no/emner/TMA4315/2018h/Lee1997.pdf) article, which gives more background. However, the model we use in this exercise is a simplification of the model in the article.

Our model is as follows: For each game, we assume that the score (the number of goals) of the home team is independent of the score of the away team. We assume that each team has a single parameter that measures its strength. We denote this strength parameter β_{A} for team A, and so on. (In the article by Lee (1997) they use two parameters, one offensive and one defensive. We use only one!)

For a match where the home team is A and the away team is B, the score (number of goals) for team A will have the Poisson distribution with mean λ, where

$$\ln \lambda = \beta_0 + \beta_{\mathrm{home}} + \beta_{\mathrm{A}} - \beta_{\mathrm{B}},$$

and the score (number of goals) for team B will have the Poisson distribution with mean λ, with

$$\ln \lambda = \beta_0 - \beta_{\mathrm{A}} + \beta_{\mathrm{B}},$$

where β_0 is our intercept and β_{home} is a home advantage parameter, both are taken to be the same for all teams. If two teams have equal strength, then $\beta_A = \beta_B$, then $\exp(\beta_0)$ will give the expected number of goals for the away team, and $\exp(\beta_0 + \beta_{\mathrm{home}})$ will give the expected numer of goals for the home team.

a) (1 point)

Is the assumption of independence between the goals made by the home and away teams reasonable?

Hint: See the attached article (Lee, 1997 (https://www.math.ntnu.no/emner/TMA4315/2017h/Lee1997.pdf)), and Wikipedia article on the chi-squared test (https://en.wikipedia.org/wiki/Pearson%27s_chi-squared_test). Useful concepts include *contingency tables* and *Pearson's* χ^2 *test*. The function `chisq.test` in `R` can be used, but you can also implement it yourselves to get a better understanding of the test.
Testing for independence in contingency tables will be covered in the lectures in Module 6 (the same week the deadline for this project is), so this is a teaser on the subject.

b) (1 point)

If a match has a winning team, the winner gets 3 points and the loser gets 0. If the match is a draw, both teams get 1 point each. Produce the preliminary ranking for the season as of October 1st (this will differ from the official rankings since the season is not finished!). Note that you need to calculate the goal differences as well, as some teams have the same number of points.

c) (2 points)

In this exercise you shall make a function that performs the regression for you (but no printing or anything else is required, no classes and packages are necessary to make!).

Using Poisson regression, estimate the intercept, the home advantage and the strength parameter for each team. Produce a ranking based on estimated strengths and compare with the ranking from b). Discuss.

Hints: The response vector – the scores – has length $192 \cdot 2 = 384$ and contains the last two columns of `tippeliga` (`yh` and `ya`). From the formulas above, we see that the linear predictors are of the form

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

Let A be the home team and B the away team. If the score Y is for A, then $x_{\mathrm{home}} = 1$, $x_{\mathrm{A}} = 1$, $x_{\mathrm{B}} = -1$ and $x_{\mathrm{C}} = 0$ for all other teams C. If the score Y is for B, then $x_{\mathrm{home}} = 0$, $x_{\mathrm{A}} = -1$, $x_{\mathrm{B}} = 1$ and $x_{\mathrm{C}} = 0$ for all other teams C. Thus we have $n + 1 + 1 = 18$ covariates (including the intercept).

Instead of constructing 18 covariate vectors, you may find it easier to construct the 384×18 design matrix `X`. Then you can specify the model as `formula = goals ~ -1+X`, where `goals` is the response vector, and `-1` since `R` automatically adds an additional intercept term if you do not explicitely prohibit it in this way.

There is a complication: The 16 team covariate vectors are linearly dependent (they sum to the zero vector), and thus there are infinitely many parameter vectors giving the same predictor – also for the predictor maximizing the likelihood. You may choose any of them, but to get `optim` to work, the simplest is to force one of the parameters, say $\beta_{\mathrm{BodoeGlimt}}$, to be zero, which is equivalent to removing the covariate $x_{\mathrm{BodoeGlimt}}$, thus reducing the number of team covariates from 16 to 15. In effect, the strength parameter of Bodø/Glimt serves as a reference having value zero, with stronger teams having greater strength parameter and weaker teams having less.

To make interpretation easier, make sure to name your covariates (use `colnames` for the columns of `X`) according to the names of the teams, and e.g. `HomeAdvantage` for the home advantage covariate and `Intercept` for the intercept.

You may want to compare your results with the results of the built-in function `glm`:

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

d) (2 points)

Finally, we want to investigate rankings by means of simulation instead of comparing estimated strength. To do this, we use the estimated strengths of each team and the intercept and the home advantage, and simulate $1\,000$ full seasons.

New: Choose whether you want to

1. simulate the full season, also the first 191 matches, or
2. simulate only the last 49 games, and use the known results from the 191 first games.

The following command will extract a dataframe with the unplayed matches in the 2023 season, which you may find useful.

```
Unplayed <- eliteserie[is.na(eliteserie$yh), c("home", "away")]
```

For each season, produce the final ranking, and then study the simulated distribution of the final rank/final number of points for each team. Compare with the preliminary ranking of the (real) 2023 season and discuss. Useful quantities here include mean, variance/standard deviation, confidence intervals and the proportion of seasons each team has won or lost. You should include at least one plot in this exercise!!

The distributions of the final rank for each team can be visualized using histograms. This is quite simple to do with `ggplot` (but you do not have to use it). Useful commands in `R` include the function `melt` from the `reshape2` library, and the command `facet_wrap(~ teams)` helps with displaying the histograms together. Other types of plots can also be helpful in visualizing the results.

Hints: The simulation may take some time, so you can do the simulation once and save the simulated results in a `txt` file or an `R` file (see `write.table` and `saveRDS`). Then you can load the simulated data into `R` when they are needed. If you want your simulations to be reproducable you may `set.seed(42)` (or any other favorite number of yours) before you perform the simulations.