MTHM506/COMM511 - Statistical Data Modelling

Topic 1 - Maximum Likelihood Estimation

Preliminaries

In this session, we will continue with Topic 1 and introduce Maximum Likelihood Estimation. These notes refer to Topics 1.1-1.6 (and 1.10) from the lecture notes. All practical aspects in this session will be done using the `fish` dataframe. This can be found in `datasets.RData` on the course ELE page and can be loaded into `R` using the `load()` function.

```
# Loading datasets required
load('datasets.RData')
```

For the practical parts of this section, we need the `plot3D` package which will help us produce 3D plots. We use the `install.packages()` function to download and install the package and use the `library()` function to load them into the `R` library.

```
# Installing required packages
install.packages("plot3D")

# Loading required packages into the library
library(plot3D)
```

Introduction

In the last session, we saw that Linear Modelling (LM) is a good and flexible framework, (e.g. transform your variables, non-linear relationships in the covariates) however it didn't quite do the job we were hoping it would do.

We were looking at the example of the `fish` dataframe where we had the number of fish escaping a net from a total number of fish available of varying lengths

```
# First 12 rows
head(fish, 12)
   length escape total
1      21      1     1
2      23      1     1
3      25      6     6
4      26      6     6
5      27      9     9
6      28      7     8
7      29      3     3
8      30      8    11
9      31      6     7
10     32      2     6
11     33      6    21
12     34      5    23
```

We began by fitting a series of LMs, and then we realised that we need to take a step back and stop looking at everything through the lens of LM framework/Normal distributions and instead respect the nature of the data. In particular, we mentioned the Binomial distribution when modelling the `fish` dataframe.

Recall the proprieties of the binomial distribution

$$Y \sim Bin(N, \pi)$$

where N is the number of trials, y is the number of "successes" and π is the probability of success.

So using the `fish` dataframe, we want to build a Binomial model as follows

$$Y_i \sim Bin(N_i, \pi_i) \quad Y_i \text{ indep.}$$

where N_i is the number of available fish, y_i is the number of fish that escape and π_i is the probability of escape. We see from the data that the probability of escaping decreases for larger fish, so the question is how to we need to find a way to let π_i be a function of fish length to allow the probability to change based on the fish length. In LM we equated the mean and the linear function of the covariate like

$$\pi_i = \beta_0 + \beta_1 x_i$$

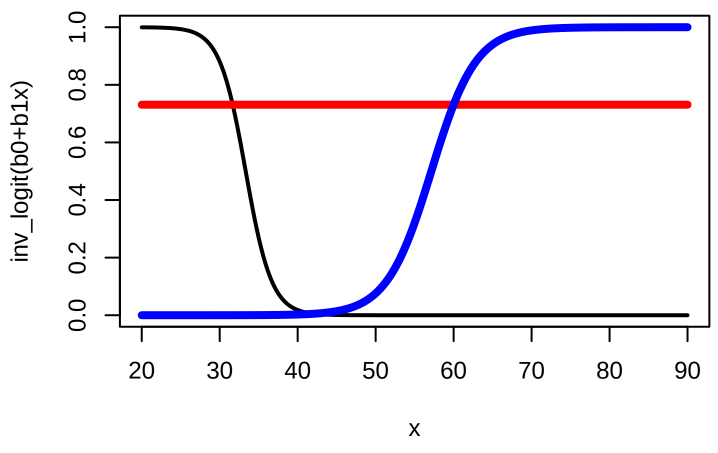
where x_i is fish length (in cm). Note that, if β_1 is positive the probability of escape increases and if β_1 is negative the probability of escape decreases in fish length.

Is this suitable way to model π_i? Remember π is a probability bounded between zero and one, so it would be difficult to get β_0 and β_1 that would you guarantee a π_i between zero and one for ALL values of x. We need to find a way to link π_i to $\beta_0 + \beta_1 x_i$ that constrains π_i. We saw a function in the last session that could help us do that… the logit function. We model the probability of escaping using a *logistic* relationship:

$$\pi_i = \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}} \implies \log\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta_0 + \beta_1 x_i$$

Let's show that this works

```r
# Inverse logit function with covariates
inv_logit <- function(x,b0,b1){ exp(b0+b1*x)/(1+exp(b0+b1*x)) }

# Sequences of
x <- seq(20,90,len=200)

# Plotting output of inverse logit function for different betas
plot(x,inv_logit(x, b0 = 20, b1 = -0.6), type = "l",lwd = 2,
     ylim = c(0, 1), ylab = "inv_logit(b0+b1x)")
lines(x,inv_logit(x, b0 = 1, b1 = 0), lwd = 4,col = "red")
lines(x,inv_logit(x, b0 = -20, b1 = 0.35), lwd = 4, col = "blue")
```



So using this we have what we need to specify a model. We have a distribution for our response variable which is a distribution that respects the nature of the data and a specified model for the parameter of interest. The next question is how do we estimate the unknowns (β_0 and β_1) in this model? (Everything else is known, the number of escaped fish and the total number of fish).

Maximum Likelihood Estimation

When we were in the LM framework we just used least squares estimation (see MTHM502/MTHM503 for more details). Here we cannot do that so in this module we will learn a general technique called Maximum Likelihood Estimation (MLE) and is a commonly used technique for estimating model parameters in these types of models. In short, from the lecture notes we have seen that the likelihood is the joint probability of having obtained the data $\boldsymbol{y}$ from our model given our unknown parameters $\boldsymbol{\theta}$

$$Y_i \sim p(\boldsymbol{\theta})$$

$$L(\boldsymbol{\theta}; \boldsymbol{y}) = \prod_{i=1}^{n} p(\boldsymbol{y}; \boldsymbol{\theta})$$

In the fish example, different values of β_0 and β_1 would give different probabilities/different chances of fish escaping from our net. MLE theory says lets choose the values of β_0 and β_1 that maximises the likelihood, the values that give us the biggest chance that we saw the data we saw.

To fit using MLE we need to write down the likelihood and/or the log-likelihood down. The Binomial probability mass function for $Y \sim Bin(N_i, \pi_i)$ is given by

$$P(Y_i = y_i) = \binom{N_i}{y_i} \pi_i^{y_i} (1 - \pi_i)^{(N_i - y_i)}$$

From Slide 49 in the Topic 1 slide deck we obtain the likelihood by taking the product of the probability mass function for all i in our dataset and so:

$$L(\beta_0, \beta_1; y_1, \cdots, y_n) = L(\boldsymbol{\theta}; \boldsymbol{y}) = \prod_{i=1}^{n} P(Y_i = y_i) = \prod_{i=1}^{n} \binom{N_i}{y_i} \pi_i^{y_i} (1 - \pi_i)^{(N_i - y_i)}$$

where $\boldsymbol{y} = (y_1, \cdots, y_n)$ and $\boldsymbol{\theta} = (\beta_0, \beta_1)$. In practice though, the log-likelihood is easier to work with numerically, and we can obtain the log-likelihood by taking logs and rearranging,

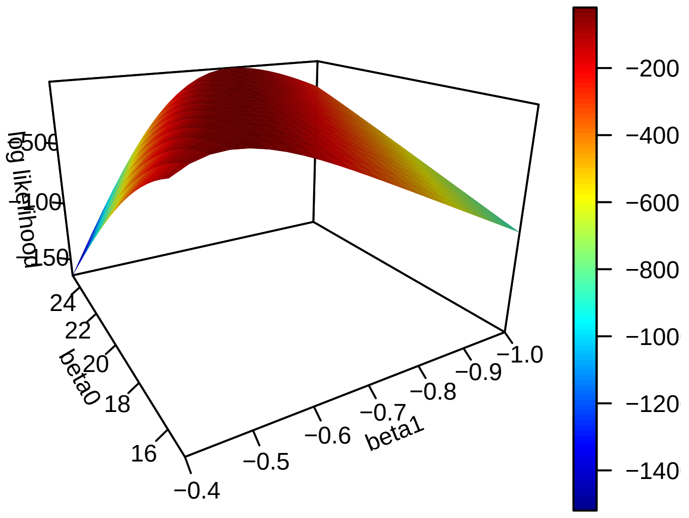
$$\ell(\boldsymbol{\theta}; \boldsymbol{y}) = \sum_{i=1}^{n} \log \binom{N_i}{y_i} + \sum_{i=1}^{n} y_i \log \pi_i + \sum_{i=1}^{n} (N_i - y_i) \log(1 - \pi_i)$$

and then seeing that $logit(\pi_i) = \beta_0 + \beta_1 x_i$ the log likelihood becomes

$$\ell(\boldsymbol{\theta}; \boldsymbol{y}) = \sum_{i=1}^{n} \log \binom{N_i}{y_i} + \sum_{i=1}^{n} y_i(\beta_0 + \beta_1 x_i) - \sum_{i=1}^{n} N_i \log(1 + e^{\beta_0 + \beta_1 x_i})$$

Let's visualise and illustrate what a log-likelihood looks like in terms of β_0 and β_1 in the case of the `fish` dataframe

```r
fish.likelihood <- function (like_look = c(15, 25, -1, -0.4)){
        ### Create a function to evaluate minus the log-likelihood
        likelihood <- function(beta){
                result <- sum(lchoose(fish$total, fish$escape)) + sum(fish$escape *
                (beta[1] + beta[2] * fish$length)) - sum(fish$total *
        log(1 + exp(beta[1] + beta[2] * fish$length)))
        return(result)
    }
    ### Plot the likelihood surface
    ## Create sequences of length 25 for beta0 and beta1 values to be used for plotting
    beta0grid <- seq(like_look[1], like_look[2], len = 25)
    beta1grid <- seq(like_look[3], like_look[4], len = 25)
     ## Create a grid from those sequences (grid size is 25x25=625)
    grid <- expand.grid(beta0grid, beta1grid)
     ## Create empty vector to store the value of the log-likelihood at each grid point
    like <- rep(0, 625)
      ## For loop to calculate the likelihood at each gridpoint
    for (i in 1:625){
       like[i] = likelihood(as.numeric(grid[i, ]))
    }
  ## Put the likelihood values into a matrix to use with the persp3D command
    like <- matrix(like, 25, 25)
  ## Plot the 3D surface
    persp3D(beta0grid, beta1grid, like, theta = 240, phi = 20,
    ticktype = "detailed", xlab = "beta0", ylab = "beta1",
    zlab = "log likelihood", expand = 2/3, shade = 0.5)
}
# run the function
fish.likelihood()
```

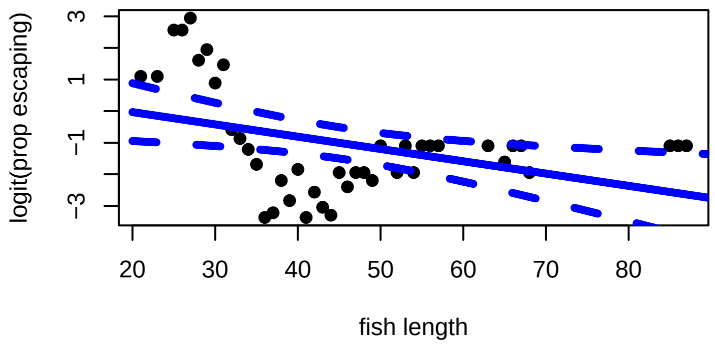


This has plotted a surface for varying values of β_0 and β_1 and MLE theory says, lets take the values of β_0 and β_1 which maximises this surface (i.e. the values which make the data more probable). The question then becomes, how to we find where the maximum occurs.

When we want to maximise something we need to differentiate (See Slide 49 in the Topic 1 Notes) and solve for where the derivatives equal zero. In LM, this is exactly the same as least squares, however, in a more general case like the one here we cannot solve analytically. WE have do do it numerically (using algorithms like Newton-Raphson). In `R` it is relatively simple to do this, with the `nlm()` function! With this function we will manually maximise the likelihood.

First lets recap our previous attempts at fitting this data

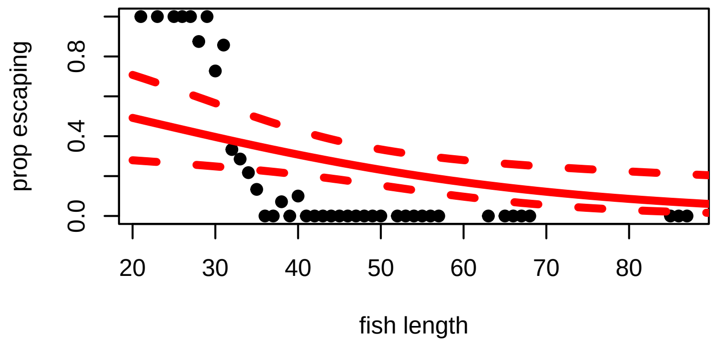
```
# define the logit as an R function
logit <- function(z){ log(z/(1-z)) }

# Create new transformed proportion variable as per lecture slides
fish$logitZ <- logit((fish$escape+0.5)/(fish$total+1))

# Plot this transformed proportion variable against fish length
plot(fish$length,fish$logitZ,xlab="fish length",ylab="logit(prop escaping)",pch=19)

# Fit the linear model
fishmodel_lm <- lm(logitZ~length,data=fish)
summary(fishmodel_lm)

Call:
lm(formula = logitZ ~ length, data = fish)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7127 -1.0365  0.0273  1.2279  3.2488

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.74623    0.71702   1.041   0.3042
length      -0.03891    0.01464  -2.657   0.0113 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.586 on 40 degrees of freedom
Multiple R-squared:   0.15, Adjusted R-squared:  0.1288
F-statistic: 7.061 on 1 and 40 DF,  p-value: 0.01127

# Add the estimated line to the existing scatterplot
xx <- seq(20,90,len=200)
preds <- predict(fishmodel_lm,newdata=data.frame(length=xx),interval="confidence")
lines(xx,preds[,"fit"],col="blue",lwd=4)
lines(xx,preds[,"upr"],col="blue",lwd=4,lty=2)
lines(xx,preds[,"lwr"],col="blue",lwd=4,lty=2)
```



Instead lets see the results on the probability scale

```
# Create prop. vs length plot again
plot(fish$length,fish$escape/fish$total,xlab="fish length", ylab="prop escaping",pch=19)
```

```r
# Add the fitted proportions from the linear model, with 95% CIs calculated
# by exponentiating end points of the CI at the logit scale.
inv_logit <- function(x){
  exp(x)/(1+exp(x))
}
lines(xx,inv_logit(preds[,"fit"]),lwd=4,col="red")
lines(xx,inv_logit(preds[,"lwr"]),lwd=4,col="red",lty=2)
lines(xx,inv_logit(preds[,"upr"]),lwd=4,col="red",lty=2)
```

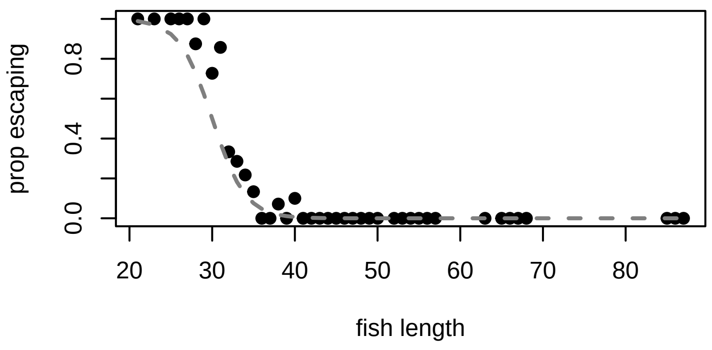


Now we consider the more sensible Binomial model. We first need to create a function that evaluates the negative log-likelihood (for different values of β_0 and β_1) as `nlm()` will MINIMISE by default so if you want to maximise a log-likelihood you need a function to compute MINUS the log-likelihood to pass to nlm().

```r
# Likelihood function
likelihood <- function(beta){
  # Log-likelihood
    result <- sum(lchoose(fish$total, fish$escape)) + sum(fish$escape *
    (beta[1] + beta[2] * fish$length)) - sum(fish$total *
    log(1 + exp(beta[1] + beta[2] * fish$length)))
    # Returning negative log-likelihood
    return(-result)
}
```

We will need starting guesses to kick the algorithm off, which can have a big effect on the optimisation algorithm so sensible ones will need to be chosen. Lets choose $\beta_0 = 15$ and $\beta_1 = -0.5$. Lets plot the starting guess of π_i over the data to see if it is sensible.

```r
## Involves some trial and error of course
beta_init <-  c(15, -0.5) # I started up using c(0, 1)

## Calculate probabilities (proportions) using those starting values
yfit <- (exp(beta_init[1] + beta_init[2] * fish$length))/(1 + exp(beta_init[1] + beta_init[2] * fish$len

# Plot the probability curve
plot(fish$length,fish$escape/fish$total,xlab="fish length", ylab="prop escaping",pch=19)
lines(sort(fish$length), yfit[order(fish$length)],lwd=2,lty=2,col="grey50")
```



Now that we have starting guesses we can minimise the negative log-likelihood (or maximise the log-likelihood)

```
## Minimize minus the log-likelihood
out <- nlm(likelihood, # Our likelihood function
           p = beta_init, # Our initial values
           hessian = T, # We want to compute the hessian
           gradtol = 1e-10, # Tolerance for converging to the maximum
           iterlim = 1000) # Will only consider 1000 iterations in our algorithm
```

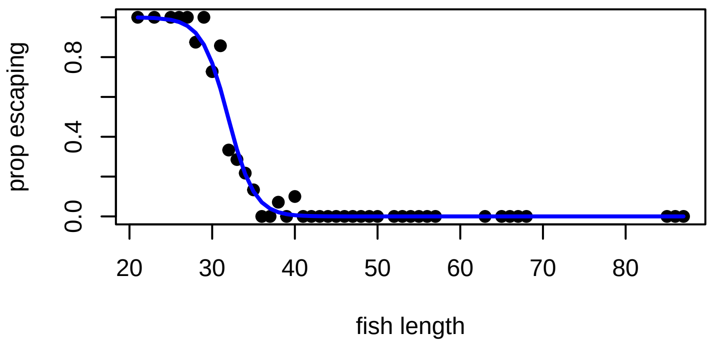
The arguments `gradtol` and `iterlim` can be changed in case the function fails to converge. gradtol controls how close the estimated have to be in value before the optimisation algorithm stops. `iterlim` is the maximum number of iterations the algorithm will do before giving up. Type `?nlm` for more details. You will only need these if the algorithm doesn't run so well or if you have a very complicated model.

Lets look at the outputs

```
## Print out estimates of beta0 and beta1
betas <- out$estimate
round(betas,3) # beta0 and beta1 estimates
[1] 20.124 -0.631

## Print out the log-likelihood at maximum
round(-out$minimum,3) # log likelihood at maximum
[1] -17.31
```

Are the fitted values any good? Lets plot them:

```
# Plot the data
plot(fish$length,fish$escape/fish$total,xlab="fish length", ylab="prop escaping",pch=19)

## Add estimated probabilities to the plot
yfit <- (exp(betas[1] + betas[2] * fish$length))/(1 + exp(betas[1] + betas[2] * fish$length))
lines(sort(fish$length), yfit[order(fish$length)],lwd=2,col="blue")
```



We can see that its generally a good fit! It goes through the dataset. We can see that modelling things by respecting the data generating process gives us much better results. Now, we have a reasonable model here, and we have used MLE theory to estimate the model parameters (β_0 and β_1) and with this being a statistical model, uncertainty isn't something that we can escape. What about the uncertainty surrounding the model parameters?