Generalised Linear Models: R Lab: Probit

Tarun Agarwal

BS2039

We will learn how to fit a Generalised Linear Model. We would follow the footsteps of D J Finney and carry out a similar analysis without using any specialised generalised linear model tool using R.

Experiment

Let us first create our dataset. We want to administer 7 quantity of doses, starting from 7 units to 13 units, applying each quantity to 100 mice. Thus we have 7 quantity of doses and 100 mice for each quantity of dose. At first, we take 100 mice randomly with average lethal dose, i.e, the average of the maximum quantity of dose each mice can endure before dying, as 10 and administer a dose of 7 units to each one of them. We note whether the mice is dead or not, i.e, we check whether dose applied is greater than or equal to the actual lethal dose of that particular mice and calculate the proportion of dead mice for each quantity of dose. We repeat this process by taking another 100 mice and administering dose 8 (instead of 7), and so on.

We record our observations in a matrix with two columns, namely *Dose* and *Dead*. Thus, this matrix has 700 rows, each row referring to each mice and 2 columns.

R Code

We create a variable *dose* which stores the quantity of doses. It takes values 7 to 13. So, there are 7 quantities of doses, which is reflected by the variable n as it stores the length of *dose*. Further, we create an array to store the proportion of dead mice for each quantity of dose.

```
dose=7:13                           # We are registering 7 doses of intensity
                                    # 7 to 13
n=length(dose)                      # Number of doses is thus 7
```

```
propdeath=numeric(n)              # Calculating the proportion of deaths
                                  # for each amount of dose
```

Now we create a matrix *mat* of two columns *Dose* and *Dead*. For each dose, we take 100 actual lethal dose of mice from normal distribution with mean 10. We store the result in the variable *dead*, i.e,

$$\text{dead} = \begin{cases} 1 & \text{if dose applied is greater than or equal to lethal dose} \\ 0 & \text{otherwise} \end{cases}$$

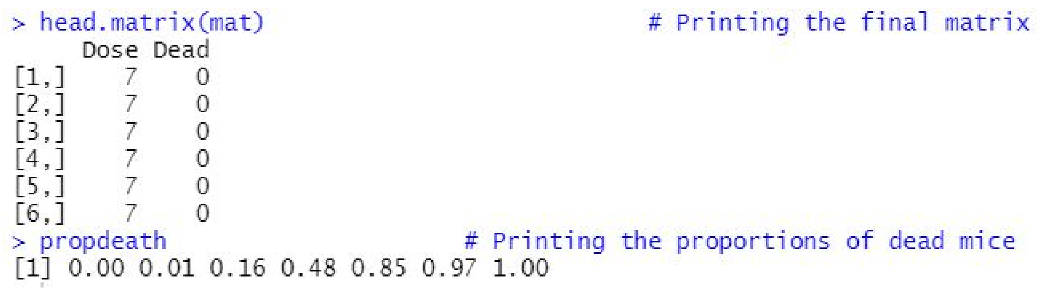
We add this information about each mice by appending an additional row to our matrix with two columns (dose applied, dead). Finally, we calculate the proportion of dead mice for each dose and store it in *propdeath*. The R-code is as follows :

```
for(i in 1:n)
{
  lethal=rnorm(100,mean=10)      # Taking 100 mice randomly for each dose
                                 # with the actual lethal dose approx. 10
  dead=dose[i]>=lethal           # Stores value 1 if dose is greater than
                                 # lethal dose of that particular mice and
                                 # hence the mice is dead, otherwise 0
  mat=rbind(mat,cbind(dose[i],dead))     #Appending additional row with
                                 # information about the next mice, i.e,
                                 # the dose applied to that mice and the
                                 # result(whether alive or dead)
  propdeath[i]=mean(dead)        # Calculating the proportion of mice dead
                                 # for each dose
}
```

For printing the first few rows of *mat* and *propdeath*, we write

```
head.matrix(mat)                 # Printing the final matrix
propdeath                        # Printing the proportions of dead mice
                                 # for each dose
```

The Output :



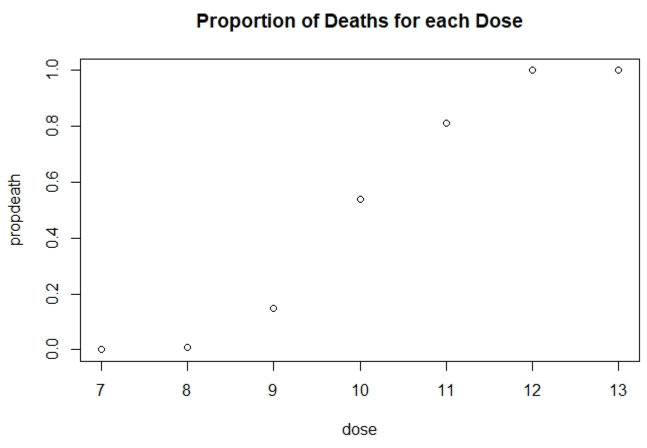
```
> head.matrix(mat)                              # Printing the final matrix
     Dose Dead
[1,]    7    0
[2,]    7    0
[3,]    7    0
[4,]    7    0
[5,]    7    0
[6,]    7    0
> propdeath                       # Printing the proportions of dead mice
[1] 0.00 0.01 0.16 0.48 0.85 0.97 1.00
```

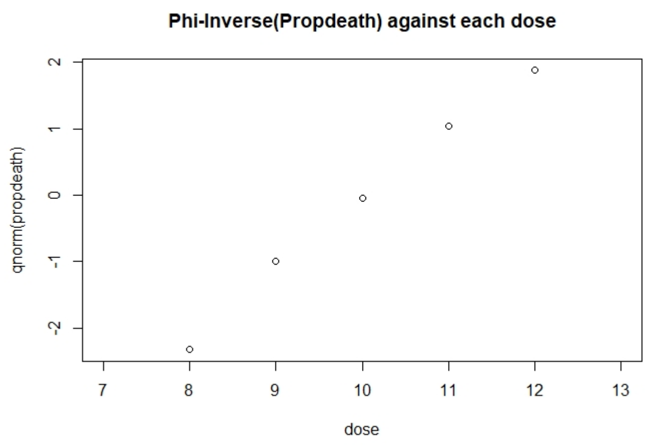
Now, for plotting *propdeath* against dose and then $\Phi^{-1}(propdeath)$ against dose, we write

```
plot(dose,propdeath,main="Proportion of Deaths for each Dose")
                                        # Plotting the Proportion of Deaths
                                        # for each Dose

plot(dose,qnorm(propdeath),main="Phi-Inverse(Propdeath) against
each dose")
                                        # Plotting Phi-inverse of propdeath
                                        # against dose
```

The plots are as follows :

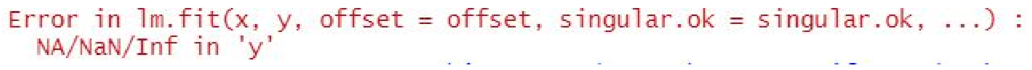




Notice that the first plot is not linear but the second plot is approximately linear, as we should get. So let us fit a line to confirm this.

```
    lm(qnorm(propdeath)~dose)     # Plotting the fitted line
                                  # This command may show error if propdeath
                                  # takes value 0
```

The output is :



```
Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
  NA/NaN/Inf in 'y'
```

Notice, an error message gets printed as *propdeath* takes value 0 when dose is 7.

Thus, $\Phi^{-1}(propdeath) = -\infty$.

We can avoid this by deliberately setting the value as some constant. Thus, this method is not free of trouble and is a crude technique. Instead we use Maximum Likelihood Estimate.