DirectedStudiesAssignment1

Sarah Chopra

22/09/2022

```
rm(list = ls())

# Loading packages
require(aplore3)
```

```
## Loading required package: aplore3
```

```
require(numDeriv)
```

```
## Loading required package: numDeriv
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

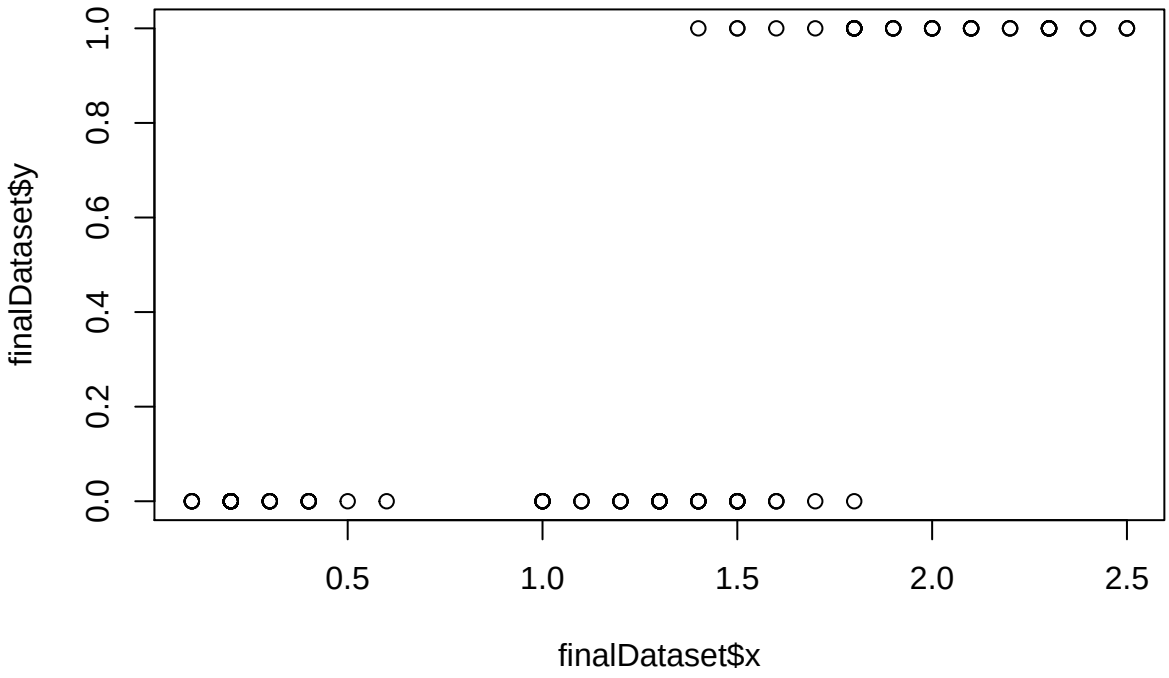
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
?iris
df = iris
df1 <-data.frame(y = c(ifelse((df$Species)=='setosa',0,ifelse((df$Species)=='virginica',1,0))),
                 x = (df$Petal.Width
                 ))


finalDataset <- dplyr::filter(df1, y==0|y==1)  ## "dplyr::" not required
plot(finalDataset$y~finalDataset$x)
```



MLE

#Function MLE:

```
F1 <- function(data){
  B0 <- data[1]
  B1 <- data[2]
  y <- finalDataset$y
  x1 <- finalDataset$x
  X <- B0 + B1 * x1
  p <- exp(X) / (1 + exp(X))
  mle <- -sum(y * log(p) + (1 - y) * log(1 - p))
# mle <- sum(log(pi)*(y) + (1-(y))*log(1-pi))

  (mle)
}
```

```
MLE <- optim(c(0.1,0.1),
             fn = F1,
             control = list(trace = 0),
             hessian = FALSE)
MLE
```

```
## $par
```

```
## [1] -21.12900  12.94969
##
## $value
## [1] 16.7104
##
## $counts
## function gradient
##       95       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
print("Paramters")
```

```
## [1] "Paramters"
```

```
print(MLE$par)
```

```
## [1] -21.12900  12.94969
```

```
paste("MLE", F1(MLE$par))
```

```
## [1] "MLE 16.7104045105802"
```

#In-Build GLM

```
glm_model <- glm(y ~ x ,
                 data = finalDataset,
                 family = binomial(link = "logit"))

coef(glm_model)
```

```
## (Intercept)           x
##   -21.12564    12.94751
```

#Newton Raphson Method

```
y <- finalDataset$y
x1 <- finalDataset$x

ZTP_MLE_NewtonRaphson <- function (x1,y, itr, B0=0,B1=0) {
b0 <- rep(0,itr+1)
b1 <- rep(0,itr+1)
b0[1] <- B0
b1[1] <- B1

for (i in 1:itr){
```

```
e <- exp(b0[i] + b1[i] * x1)

#b0[i+1] =b0[i]-((sum(y)-sum(e/(1+e))/-sum(e/e^2)))
#b1[i+1]=b1[i]-((sum(x1*y)-sum(x1*e/(1+e)))/-sum(((x1^2)*e)/(e^2)))

b0[i+1]=b0[i]-((sum(y)-sum(e/(1+e)))/(-sum((e)/(e+1)^2)))
b1[i+1]=b1[i]-((sum(x1*y)-sum(x1*e/(1+e)))/(-sum(((x1^2)*e)/(e^2))))
}

data = c(b0,b1)
  finaldf = data.frame (b0=b0,b1=b1, log.lik.value=F1(data))
finaldf

}
```

```
y <- finalDataset$y
x1 <- finalDataset$x

finalDF = ZTP_MLE_NewtonRaphson(x1,y,150)
finalDF
```

```
##             b0         b1 log.lik.value
## 1    0.0000000 0.00000000      127.3203
## 2   -0.6666667 0.03754176      127.3203
## 3   -0.7375138 0.10619167      127.3203
## 4   -0.8205170 0.17522489      127.3203
## 5   -0.9051525 0.24450281      127.3203
## 6   -0.9910310 0.31381530      127.3203
## 7   -1.0778961 0.38295220      127.3203
## 8   -1.1654819 0.45171016      127.3203
## 9   -1.2535186 0.51989734      127.3203
## 10  -1.3417390 0.58733731      127.3203
## 11  -1.4298844 0.65387196      127.3203
## 12  -1.5177107 0.71936343      127.3203
## 13  -1.6049917 0.78369498      127.3203
## 14  -1.6915230 0.84677097      127.3203
## 15  -1.7771235 0.90851611      127.3203
## 16  -1.8616361 0.96887412      127.3203
## 17  -1.9449277 1.02780602      127.3203
## 18  -2.0268884 1.08528823      127.3203
## 19  -2.1074296 1.14131055      127.3203
## 20  -2.1864828 1.19587416      127.3203
## 21  -2.2639972 1.24898979      127.3203
## 22  -2.3399376 1.30067593      127.3203
## 23  -2.4142829 1.35095733      127.3203
## 24  -2.4870233 1.39986361      127.3203
## 25  -2.5581595 1.44742810      127.3203
## 26  -2.6277003 1.49368680      127.3203
## 27  -2.6956613 1.53867756      127.3203
## 28  -2.7620638 1.58243940      127.3203
## 29  -2.8269338 1.62501191      127.3203
## 30  -2.8903005 1.66643479      127.3203
## 31  -2.9521960 1.70674750      127.3203
```

```
## 32  -3.0126542 1.74598898      127.3203
## 33  -3.0717107 1.78419740      127.3203
## 34  -3.1294017 1.82141001      127.3203
## 35  -3.1857641 1.85766302      127.3203
## 36  -3.2408349 1.89299152      127.3203
## 37  -3.2946512 1.92742940      127.3203
## 38  -3.3472495 1.96100937      127.3203
## 39  -3.3986659 1.99376288      127.3203
## 40  -3.4489361 2.02572018      127.3203
## 41  -3.4980947 2.05691030      127.3203
## 42  -3.5461758 2.08736109      127.3203
## 43  -3.5932125 2.11709923      127.3203
## 44  -3.6392369 2.14615030      127.3203
## 45  -3.6842804 2.17453877      127.3203
## 46  -3.7283733 2.20228808      127.3203
## 47  -3.7715450 2.22942065      127.3203
## 48  -3.8138237 2.25595794      127.3203
## 49  -3.8552371 2.28192049      127.3203
## 50  -3.8958115 2.30732794      127.3203
## 51  -3.9355728 2.33219911      127.3203
## 52  -3.9745455 2.35655199      127.3203
## 53  -4.0127534 2.38040380      127.3203
## 54  -4.0502198 2.40377104      127.3203
## 55  -4.0869666 2.42666952      127.3203
## 56  -4.1230153 2.44911436      127.3203
## 57  -4.1583865 2.47112006      127.3203
## 58  -4.1931001 2.49270052      127.3203
## 59  -4.2271752 2.51386907      127.3203
## 60  -4.2606304 2.53463847      127.3203
## 61  -4.2934834 2.55502099      127.3203
## 62  -4.3257513 2.57502838      127.3203
## 63  -4.3574509 2.59467195      127.3203
## 64  -4.3885979 2.61396252      127.3203
## 65  -4.4192078 2.63291052      127.3203
## 66  -4.4492955 2.65152595      127.3203
## 67  -4.4788753 2.66981843      127.3203
## 68  -4.5079609 2.68779721      127.3203
## 69  -4.5365658 2.70547118      127.3203
## 70  -4.5647028 2.72284890      127.3203
## 71  -4.5923844 2.73993862      127.3203
## 72  -4.6196224 2.75674826      127.3203
## 73  -4.6464286 2.77328546      127.3203
## 74  -4.6728141 2.78955759      127.3203
## 75  -4.6987898 2.80557173      127.3203
## 76  -4.7243660 2.82133472      127.3203
## 77  -4.7495529 2.83685316      127.3203
## 78  -4.7743602 2.85213342      127.3203
## 79  -4.7987975 2.86718162      127.3203
## 80  -4.8228737 2.88200369      127.3203
## 81  -4.8465979 2.89660535      127.3203
## 82  -4.8699784 2.91099214      127.3203
## 83  -4.8930237 2.92516937      127.3203
## 84  -4.9157417 2.93914223      127.3203
## 85  -4.9381401 2.95291568      127.3203
```

```
## 86  -4.9602265 2.96649456     127.3203
## 87  -4.9820081 2.97988353     127.3203
## 88  -5.0034920 2.99308709     127.3203
## 89  -5.0246851 3.00610961     127.3203
## 90  -5.0455939 3.01895533     127.3203
## 91  -5.0662249 3.03162833     127.3203
## 92  -5.0865843 3.04413257     127.3203
## 93  -5.1066782 3.05647190     127.3203
## 94  -5.1265123 3.06865003     127.3203
## 95  -5.1460924 3.08067058     127.3203
## 96  -5.1654241 3.09253703     127.3203
## 97  -5.1845125 3.10425278     127.3203
## 98  -5.2033631 3.11582111     127.3203
## 99  -5.2219807 3.12724521     127.3203
## 100 -5.2403704 3.13852818     127.3203
## 101 -5.2585369 3.14967303     127.3203
## 102 -5.2764847 3.16068267     127.3203
## 103 -5.2942185 3.17155993     127.3203
## 104 -5.3117426 3.18230756     127.3203
## 105 -5.3290612 3.19292824     127.3203
## 106 -5.3461784 3.20342456     127.3203
## 107 -5.3630984 3.21379905     127.3203
## 108 -5.3798250 3.22405416     127.3203
## 109 -5.3963620 3.23419227     127.3203
## 110 -5.4127131 3.24421571     127.3203
## 111 -5.4288820 3.25412673     127.3203
## 112 -5.4448721 3.26392753     127.3203
## 113 -5.4606868 3.27362025     127.3203
## 114 -5.4763296 3.28320695     127.3203
## 115 -5.4918036 3.29268968     127.3203
## 116 -5.5071120 3.30207039     127.3203
## 117 -5.5222578 3.31135101     127.3203
## 118 -5.5372441 3.32053341     127.3203
## 119 -5.5520738 3.32961940     127.3203
## 120 -5.5667498 3.33861076     127.3203
## 121 -5.5812748 3.34750923     127.3203
## 122 -5.5956515 3.35631648     127.3203
## 123 -5.6098826 3.36503416     127.3203
## 124 -5.6239706 3.37366387     127.3203
## 125 -5.6379182 3.38220717     127.3203
## 126 -5.6517276 3.39066559     127.3203
## 127 -5.6654014 3.39904062     127.3203
## 128 -5.6789418 3.40733369     127.3203
## 129 -5.6923512 3.41554624     127.3203
## 130 -5.7056317 3.42367964     127.3203
## 131 -5.7187856 3.43173523     127.3203
## 132 -5.7318149 3.43971434     127.3203
## 133 -5.7447218 3.44761826     127.3203
## 134 -5.7575083 3.45544822     127.3203
## 135 -5.7701763 3.46320547     127.3203
## 136 -5.7827278 3.47089121     127.3203
## 137 -5.7951647 3.47850659     127.3203
## 138 -5.8074889 3.48605276     127.3203
## 139 -5.8197021 3.49353085     127.3203
```

```
## 140 -5.8318061 3.50094194      127.3203
## 141 -5.8438026 3.50828711      127.3203
## 142 -5.8556934 3.51556739      127.3203
## 143 -5.8674800 3.52278382      127.3203
## 144 -5.8791642 3.52993738      127.3203
## 145 -5.8907474 3.53702905      127.3203
## 146 -5.9022313 3.54405979      127.3203
## 147 -5.9136173 3.55103054      127.3203
## 148 -5.9249070 3.55794221      127.3203
## 149 -5.9361017 3.56479569      127.3203
## 150 -5.9472030 3.57159186      127.3203
## 151 -5.9582122 3.57833159      127.3203
```

```
F2 <- function(p){
    y <- finalDataset$y
    p<-p/100
    if(any(p > 1) | any(p < 0)) {
    val <- 0
  } else {
    val <- -sum(y * log(p) + (1 - y) * log(1 - p))
  }
  val

}
uniroot(f = F2, interval=c(1,99),extendInt = "yes")
```

```
## $root
## [1] 101.97
##
## $f.root
## [1] 0
##
## $iter
## [1] 2
##
## $init.it
## [1] 2
##
## $estim.prec
## [1] 0
```

Comprison

```
print("In build GLM")
```

```
## [1] "In build GLM"
```

```
print((glm_model))
```

```
##
## Call:  glm(formula = y ~ x, family = binomial(link = "logit"), data = finalDataset)
##
```

```
## Coefficients:
## (Intercept)            x
##      -21.13        12.95
##
## Degrees of Freedom: 149 Total (i.e. Null);  148 Residual
## Null Deviance:       191
## Residual Deviance: 33.42     AIC: 37.42
```

```
## [1] "Using Optim"
```

```
## [1] "b0 -21.1289971502894"
```

```
## [1] "b1 12.9496947420738"
```

```
## [1] "log.lik.value 16.7104045105802"
```

```
print("Using Newton Raphson")
```

```
## [1] "Using Newton Raphson"
```

```
last_row = tail(finalDF, n=1)
paste("b0",last_row[1])
```

```
## [1] "b0 -5.95821217724173"
```

```
paste("b1",last_row[2])
```

```
## [1] "b1 3.57833158893427"
```

```
paste("log.lik.value",last_row[3])
```

```
## [1] "log.lik.value 127.320265537562"
```