# Homework 4 (part 2)

Michael Pena

2024-03-03

## Exercise J-2.4

### Part (a).

```
# pulling in data
dose <- c(0,2,10,50,250)
at <- c(111,105,124,104,90)
ti <- c(4,4,11,13,60)
data <- cbind(dose,at,ti)
X <- cbind(1,dose,dose^2)

# render IRWLS math
binomIRWLS <- function(m,X,beta){
  E = exp(-X%*%beta)
  p = 1 - E
  f = m*p
  J <- cbind(m*E,m*(X[,2])*E,m*(X[,3])*E)
  var <- m*p*(1-p)
  W <- diag(as.numeric(1/var))
  list(f=f, J=J, W=W)
}


# render algorithm
DJShaarvi <- function(y,m,X,beta0,W=1,maxit,tolerr = 1e-06){
  for(it in 1:maxit){

    # initial
    f = binomIRWLS(m,X,beta0)$f
    J = binomIRWLS(m,X,beta0)$J
    W = binomIRWLS(m,X,beta0)$W

    #print heading
    print(sprintf('iteration = %3.0f   beta_0 = %6.6f  beta_1 = %6.6f  beta_2 = %6.6f',
                  it,beta0[1],beta0[2],beta0[3]))

    # mathematics
    direc <- solve(t(J)%*%W%*%J)%*%t(J)%*%W%*%(y-f)
    beta1 = beta0 + direc

    # get threshhold
```

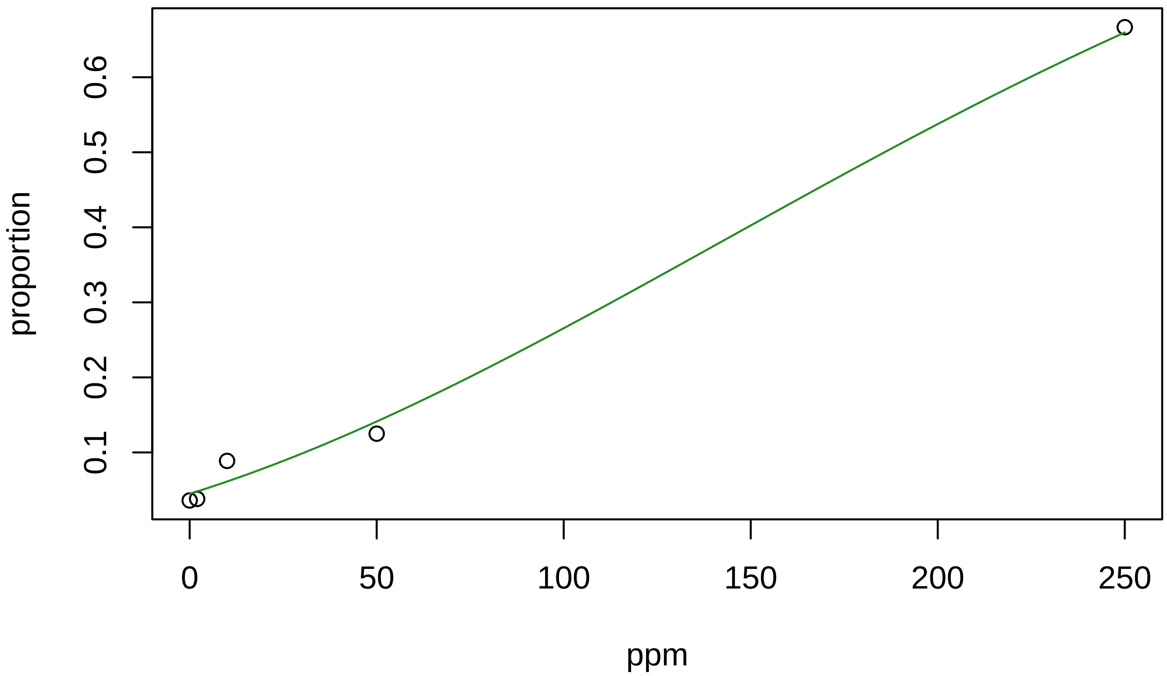
```r
    mre <- max(abs(beta1 - beta0)/abs(pmax(1,abs(beta0))))
    if(mre < tolerr){break}

    # cycle beta
    beta0 <- beta1
  }
  print(sprintf("The MLE's are beta_1 = %f, beta_2=%f, beta_3 = %f",
                beta0[1],
                beta0[2],
                beta0[3]))
}
```

```r
# run algorithm
beta_init = c(.01,.01,.00001)
DJShaarvi(ti,m = data[,2],X,beta_init,W=1,maxit=100)
```

```
## [1] "iteration =   1   beta_0 = 0.010000  beta_1 = 0.010000  beta_2 = 0.000010"
## [1] "iteration =   2   beta_0 = 0.036092  beta_1 = 0.004413  beta_2 = -0.000075"
## [1] "iteration =   3   beta_0 = 0.036365  beta_1 = 0.005049  beta_2 = -0.000062"
## [1] "iteration =   4   beta_0 = 0.037971  beta_1 = 0.004200  beta_2 = -0.000043"
## [1] "iteration =   5   beta_0 = 0.040180  beta_1 = 0.003302  beta_2 = -0.000024"
## [1] "iteration =   6   beta_0 = 0.042467  beta_1 = 0.002504  beta_2 = -0.000008"
## [1] "iteration =   7   beta_0 = 0.044419  beta_1 = 0.001926  beta_2 = 0.000004"
## [1] "iteration =   8   beta_0 = 0.045552  beta_1 = 0.001669  beta_2 = 0.000009"
## [1] "iteration =   9   beta_0 = 0.045893  beta_1 = 0.001629  beta_2 = 0.000010"
## [1] "iteration =  10   beta_0 = 0.045940  beta_1 = 0.001627  beta_2 = 0.000010"
## [1] "iteration =  11   beta_0 = 0.045944  beta_1 = 0.001627  beta_2 = 0.000010"
## [1] "The MLE's are beta_1 = 0.045944, beta_2=0.001627, beta_3 = 0.000010"
```

### Part (b).

```r
proportion <- data[,3]/data[,2] #ratio of number of incidents and the number tested.
ppm <- data[,1]

plot(ppm,proportion)

x = seq(0,250,.1)
lines(x,1 - 1/exp(0.045944 + 0.001627*x + 0.000010*x^2),col='forestgreen')
```




### Exercise J-2.5

**Part (a).**

```
# getting data
data2 <- read.table('blowBF.txt',head=T)
X2 <- cbind(1,data2[,2],log(data2[,1]))

# IRWLS setup
bernIRLWS <- function(X,B){ #bernoulli is just binom where n=1
  E = exp(X%*%B)
  p = E/(1+E)
  f = p
  var = p*(1-p)
  J <- cbind(E/(1+E)^2,X[,2]*E/(1+E)^2,X[,3]*E/(1+E)^2)
  W <- diag(as.numeric(1/var))

  list(f=f, W=W, J=J)
}

# render algorithm
DJShaarvi2 <- function(y,X,beta0,W=1,maxit,tolerr = 1e-06){
  for(it in 1:maxit){

    # initial
    f = bernIRLWS(X,beta0)$f
    J = bernIRLWS(X,beta0)$J
    W = bernIRLWS(X,beta0)$W

    #print heading
    print(sprintf('iteration = %3.0f    beta_0 = %6.6f   beta_1 = %6.6f   beta_2 = %6.6f',
                  it,beta0[1],beta0[2],beta0[3]))
```

```r
    # mathematics
    direc <- solve(t(J)%*%W%*%J)%*%t(J)%*%W%*%(y-f)
    beta1 = beta0 + direc

    # get threshhold
    mre <- max(abs(beta1 - beta0)/abs(pmax(1,abs(beta0))))
    if(mre < tolerr){break}

    # cycle beta
    beta0 <- beta1
  }
  print(sprintf("The MLE's are beta_1 = %f, beta_2=%f, beta_3 = %f",
                beta0[1],
                beta0[2],
                beta0[3]))
}
```

```r
# look for MLEs
DJShaarvi2(data2[,3],X2,beta_init,maxit=500)
```

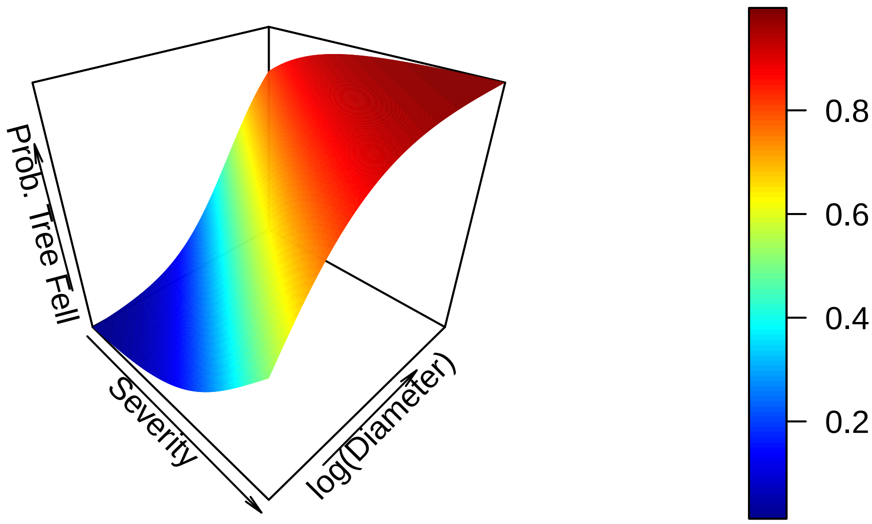
```
## [1] "iteration =   1   beta_0 = 0.010000  beta_1 = 0.010000  beta_2 = 0.000010"
## [1] "iteration =   2   beta_0 = -6.125281  beta_1 = 2.857871  beta_2 = 2.035826"
## [1] "iteration =   3   beta_0 = -8.673565  beta_1 = 4.076699  beta_2 = 2.895913"
## [1] "iteration =   4   beta_0 = -9.492713  beta_1 = 4.474466  beta_2 = 3.173975"
## [1] "iteration =   5   beta_0 = -9.561642  beta_1 = 4.508373  beta_2 = 3.197413"
## [1] "iteration =   6   beta_0 = -9.562085  beta_1 = 4.508593  beta_2 = 3.197563"
## [1] "The MLE's are beta_1 = -9.562085, beta_2=4.508593, beta_3 = 3.197563"
```

### Part (b).

```r
# 3-d rendering

# initials for graphing
X_1 <- X2[,2]
X_2 <- X2[,3]

# pi function
PI <- function(x1,x2){
 E <- exp(-9.562085 + 4.508593*x1 + 3.197563*x2)
 return(E/(1+E))
}

# ranges
x1 = seq(0,1,0.01)
x2 = seq(min(X_2),max(X_2),0.01)
x3 = outer(x1,x2,FUN = PI)

# render 3d plot
persp3D(x1,x2,x3,
        main = "Speculative Probability that Tree Falls",
        xlab = "Severity",
        ylab = "log(Diameter)",
        zlab = "Prob. Tree Fell",
```

```r
        theta = 45, phi = 33)
```

## Speculative Probability that Tree Falls




Note: "Severity" refers to severity level (ranging from 0 to 1) of storm "log(Diameter)" refers to the Log of the diameter of a tree.

```r
PI(.3,log(10))
```

```
## [1] 0.3000949
```

The model predicts approximately a 30% chance that a tree with diameter of 10 in storm of 0.3 severity will fall over.