# MA 576 HW 2

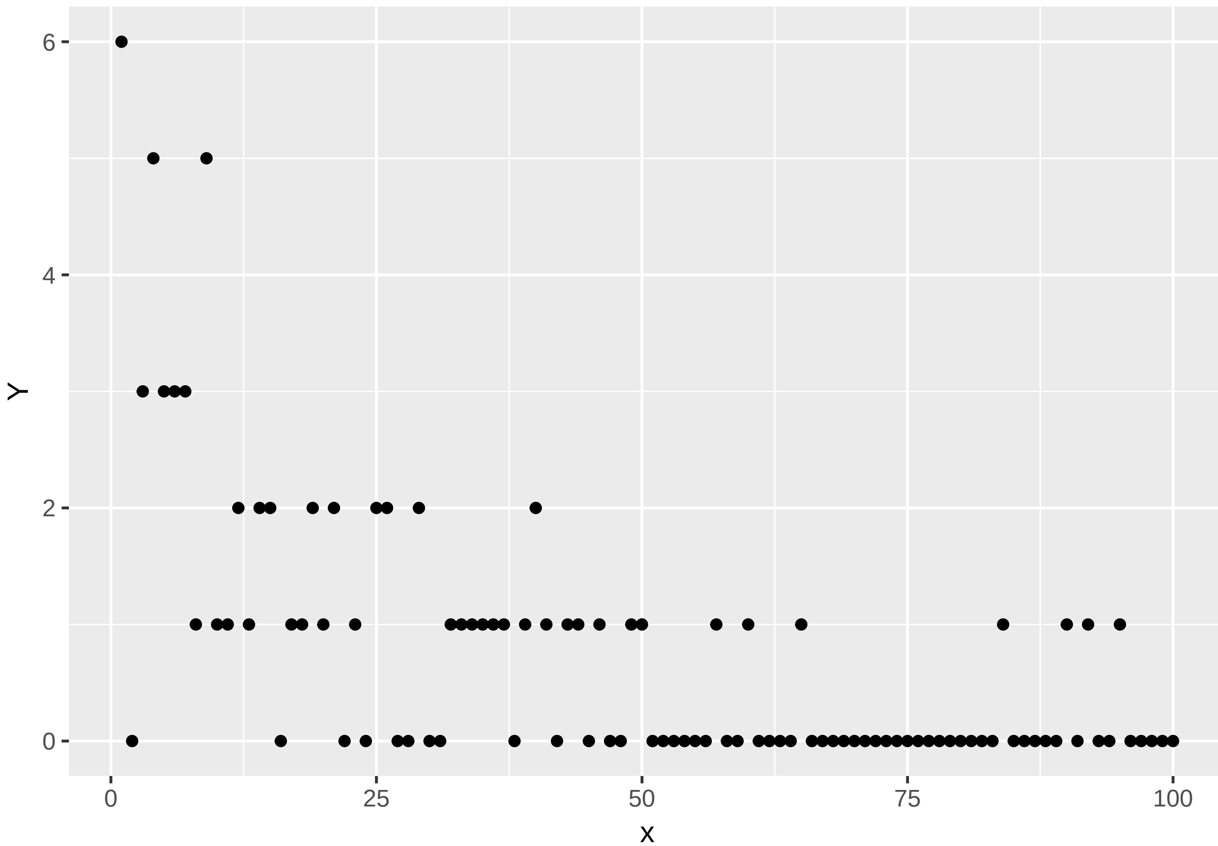
*Benjamin Draves*

## 4

### (a)

Here, we define our $x$ and $\lambda$ vector according to the function $\lambda_i = e^{1-0.04x_i}$. Lastly, we define $y_i$ to be a single realization of the random variable $Y_i \sim \text{Pois}(\lambda_i)$. This $(X, Y)$ relationship is plotted below.

```r
#load necessary packages
library(ggplot2)
library(gridExtra)

#set seed
set.seed(576)

#define X
x = 1:100

#define lamda
lam = exp(1 - 0.04*x)

#get Y sample
Y = sapply(lam, function(x) rpois(1, x))

#plot relationship
dat = data.frame(cbind(x,Y))
p1<- ggplot(dat, aes(x, Y)) +
  geom_point() +
  labs(x = "x", y = "Y")
p1
```




**(b)**

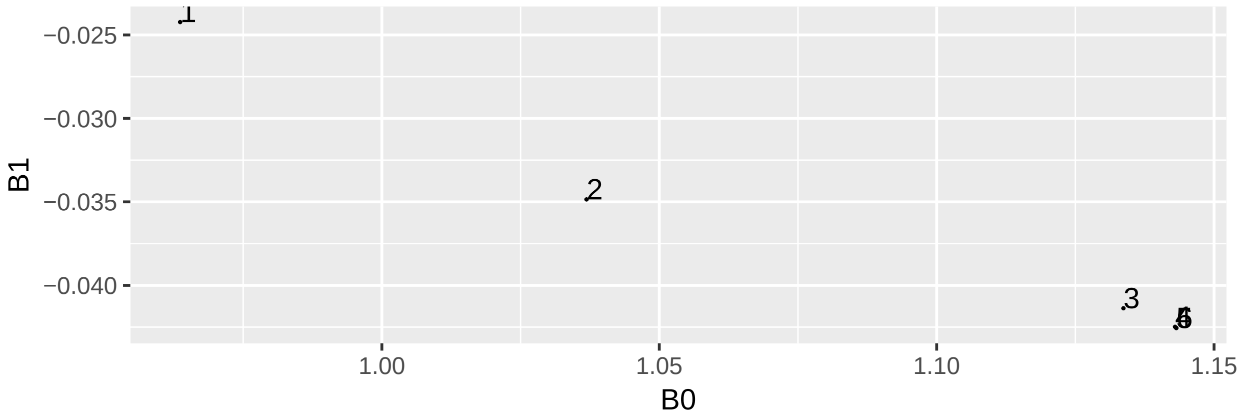
Here we implement IRLS for a GLM with canonical link $g(t) = \log(t)$. We initialize $\beta^{(0)} = (0, 0)^T$ and $\eta^{(0)} = \mathbf{X}\beta^{(0)}$. From here, we define $\mu = g^{-1}(\eta) = e^{\eta}$. Now, to define the weight matrix we see that $Var(t) = a(\phi)b''(t) = e^t$ and hence $V(\mu) = e^{\mu}$. Moreover, $g'(t) = 1/t$ so $g'(\mu)^2 = 1/\mu^2$ which gives $\mathbf{W} = \text{diag}[\frac{\mu^2}{e^{\mu_i}}]$. Using this we can define our linear space predictor as $Z = \eta^{(k)} + \text{diag}(\frac{1}{\mu_i}) * (\mathbf{Y} - \mu)$. Lastly, we simply update our parameters $\beta^{(k+1)} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{Z}$ and $\eta^{(k+1)} = \mathbf{X}\beta^{(k+1)}$. Lastly, we iterate until $\beta^{(k+1)} - \beta^{(k)}$ is zero for the first four decimal places.

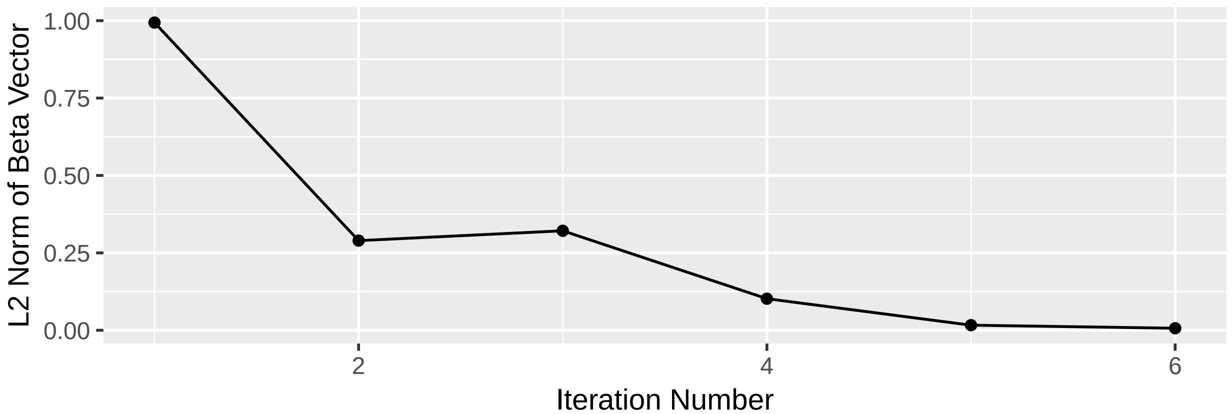
```
#implement IRLS

#set up some cool plotting things
df = matrix(NA, nrow = 0, ncol = 3)
colnames(df) = c("B0", "B1", "Difference")

#set up design matrix
X = cbind(rep(1, length(x)), x)

#initialize values
beta = matrix(0, nrow = ncol(X))
eta = X%*%beta
diff = rbind(Inf, Inf)
```

```r
#iterative loop
while(round(diff[1,1], 4) != 0 && round(diff[2,1], 4) != 0){
  #define mu
  u = exp(eta)

  #define weight matrix
  W = diag(c(u^2/exp(u)))

  #Define the linear space response
  Z = eta + diag(c(1/u))%*%(Y - u)

  #Define Fisher Information
  I = t(X)%*%W%*%X

  #Update beta
  beta.new = solve(I)%*%t(X)%*%W%*%Z

  diff = abs(beta.new - beta)
  beta = beta.new

  #update eta
  eta = X%*%beta

  #save for plotting purposes
  df = rbind(df, c(t(beta), sqrt(sum(diff))))
}

#print beta
print(beta)
```

```
##            [,1]
##      1.14321344
## x -0.04255422
```

```r
#plot estimate progressions
dat = data.frame(df, name = 1:nrow(df))
p2 = ggplot(dat, aes(B0, B1, label = name))+
  geom_point(size = 0.1)+
  geom_text(aes(label=name),hjust=0, vjust=0)

p3 = ggplot(dat, aes(name,Difference))+
  geom_line()+
  geom_point()+
  labs(x = "Iteration Number", y = "L2 Norm of Beta Vector")

grid.arrange(p2, p3, nrow=2)
```




Plotted above we have the progression of the $(\hat{\beta}_0, \hat{\beta}_1)$ estimates as they converge to the $(\hat{\beta}_0^{ML}, \hat{\beta}_1^{ML})$ estimates. The second plot is the $L^2$ norm of the difference in the $\beta$ estimates. Here our final estimates are given by $\hat{\beta} = (1.14321344, -0.04255422)^T$ which was completed in 6 iterations. We compare this to R's built in *glm* function.

```
#R's built in glm function
summary(glm(Y ~ x, family="poisson"))
```

```
##
## Call:
## glm(formula = Y ~ x, family = "poisson")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4084  -0.7080  -0.4349   0.3357   1.9030
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.146739   0.175230   6.544 5.98e-11 ***
## x           -0.040982   0.005647  -7.257 3.97e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
```

```
##     Null deviance: 149.573  on 99  degrees of freedom
## Residual deviance:  76.106  on 98  degrees of freedom
## AIC: 182.21
##
## Number of Fisher Scoring iterations: 5
```

We see that the estimates are given by $\hat{\beta} = (1.146739, -0.04288)^T$ which is completed in 5 iterations. We note that these estimates are quite similar to the estimates given above while the convergence here occurs in one fewer iteration.

### (c)

Seeing that $\hat{\beta} \overset{asy}{\sim} N(\beta, I(\beta)^{-1})$. Using the fisher information matrix from our IRLS procedure, we can build confidence intervals for the $\hat{\beta}$ vector.

```r
#set multipliers and standard errors
z = qnorm(1 - .05/2)
se = sqrt(diag(solve(I)))

#build intervals
c(beta[1] - z*se[1], beta[1] + z*se[1])
```

```
##
## 0.377170 1.909257
```

```r
c(beta[2] - z*se[2], beta[2] + z*se[2])
```

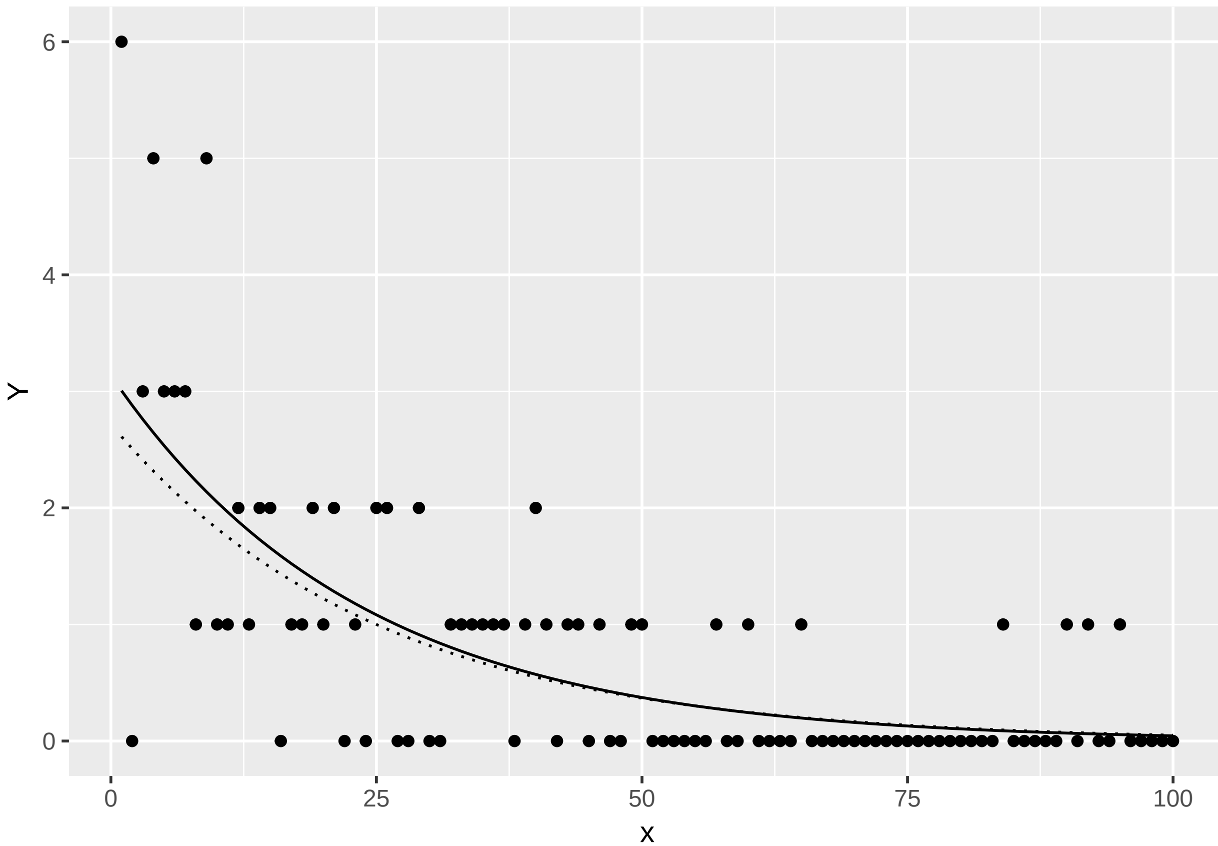
```
##           x            x
## -0.07032886 -0.01477958
```

The first confidence interval contains $\beta_0 = 1$ and the second interval contains the true value $\beta_1 = -0.04$.

### (d)

Plotted below is the fitted model $\hat{\lambda}_i$ (solid), as well as the true $\lambda_i$ (dashed). For $x > 50$ the lines are indistinguishable while for $x \leq 50$ our estimates appear to be too high. This could be attributable to only having one sample less than 3 for $\lambda$ in the neighborhood of 1. In general, however, this model fits the mean $\lambda_i$ very well.

```r
dat = data.frame(x,Y,W = lam, Z = exp(X%*%beta.new))
p3<- ggplot(dat, aes(x, Y)) +
  geom_point() +
  geom_line(aes(y = Z))+
  geom_line(aes(y = W), linetype=3)+
  labs(x = "x", y = "Y")
p3
```


6
4
2
0
Y
0
25
50
75
100
x