STAT 428 Statistical Computing

Homework 6 Solutions

Department of Statistics, University of Illinois at Urbana-Champaign

March 28, 2017

Ex.1

```
###Data
x = c(1.34,-1.38,-0.19,-0.44,1.90,-0.80,0.91,0.26,1.37,-1.62,
    -0.96,1.90,0.99,1.96,-1.57,1.51,-1.61,-1.02,-0.92,-1.87,
    1.73,-1.23,-1.24,0.22,1.42)
y = c(1,0,0,1,1,0,1,1,1,0,1,1,1,1,1,0,0,0,0,0,1,0,0,1,0)

###Function to calculate success probability
success.prob <- function(beta,x){
    p = exp(beta[1]+beta[2]*x)/(1 + exp(beta[1]+beta[2]*x))
    return(p)
}

###Function to calculate log-likelihood
log.likelihood <- function(beta,x,y){
    prob = success.prob(beta,x)
    l = sum(y*log(prob) + (1-y)*log(1-prob))
    return(-l)
}
```

Part (a)

```
beta.hat = optim(par = c(0.25,0.75), fn = log.likelihood, x = x, y = y)$par
beta.hat
```

```
## [1] 0.1158987 1.0285634
```

Part (b)

We begin by calculating jacobian and hessian matrix of log-likelihood $\ell(\beta)$

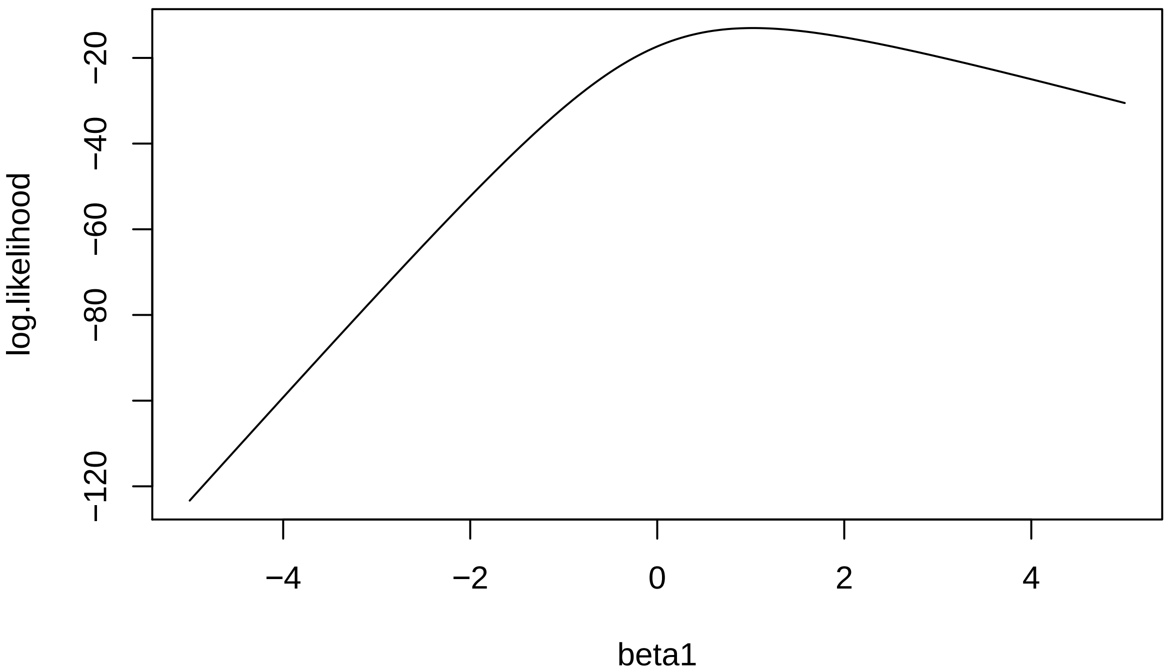
$$
\begin{aligned}
\frac{\partial \ell(\beta)}{\partial \beta_0} &= \sum_{i=1}^{n}[y_i \frac{\partial \log(\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}})}{\partial \beta_0} + (1-y_i)\frac{\partial \log(1-\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}})}{\partial \beta_0}] \\
&= \sum_{i=1}^{n}[y_i \frac{1}{1+e^{\beta_0+\beta_1 x_i}} - (1-y_i)\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}] \\
&= \sum_{i=1}^{n}(y_i - \frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}}) \\
&= \sum_{i=1}^{n}(y_i - p_i) \\
\frac{\partial \ell(\beta)}{\partial \beta_1} &= \sum_{i=1}^{n}[y_i \frac{\partial \log(\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}})}{\partial \beta_1} + (1-y_i)\frac{\partial \log(1-\frac{e^{\beta_0+\beta_1 x_i}}{1+e^{\beta_0+\beta_1 x_i}})}{\partial \beta_1}] \\
&= \sum_{i=1}^{n} x_i(y_i - p_i) \\
\frac{\partial^2 \ell(\beta)}{\partial \beta_0^2} &= -\sum_{i=1}^{n} p_i(1-p_i) \\
\frac{\partial^2 \ell(\beta)}{\partial \beta_0 \beta_1} &= -\sum_{i=1}^{n} x_i p_i(1-p_i) \\
\frac{\partial^2 \ell(\beta)}{\partial \beta_1^2} &= -\sum_{i=1}^{n} x_i^2 p_i(1-p_i)
\end{aligned}
$$

```r
###Function to calculate the next value using Newton-Raphson algorithm
LL.NR.step <- function(x,y,beta){
    #First derivatives
    prob = success.prob(beta,x)
    s1 = sum(y-prob)
    s2 = sum(x*(y-prob))
    s = c(s1,s2)
    #Second derivatives
    temp = exp(beta[1]+beta[2]*x)/((1 + exp(beta[1]+beta[2]*x))^2)
    h11 = -sum(prob*(1-prob))
    h12 = h21 = -sum(x*prob*(1-prob))
    h22 = -sum((x^2)*prob*(1-prob))
    H = matrix(c(h11,h21,h12,h22), nrow = 2)
    return(beta - solve(H)%*%s)
}
LL.NR.step(x,y,c(0.25,0.75))
```

```
##           [,1]
## [1,] 0.1269352
## [2,] 0.9984553
```

Part (c)

```
###The likelihood function with respect to beta1, given beta0 = 0
log.likelihood.beta1 <- function(beta1,x,y){
    prob = success.prob(c(0,beta1),x)
    l = sum(y*log(prob) + (1-y)*log(1-prob))
    return(-l)
}
#Plot the likelihood function as a function of beta1
beta1.val = seq(-5,5,0.05)
loglike.val = -apply(as.matrix(beta1.val), MARGIN = 1, FUN = log.likelihood.beta1, x=x, y=y)
plot(beta1.val,loglike.val, type ='l', xlab = "beta1", ylab = "log.likelihood")
```



Part (d)

Uniroot()

```
#Function to calculate the derivative of log-likelihood
dl.dbeta1 <- function(x,y,beta0,beta1){
    prob = success.prob(c(beta0,beta1),x)
    return(sum(x*(y-prob)))
}
beta1.uniroot = uniroot(dl.dbeta1, x = x, y = y, beta0 = 0, lower = 0, upper = 2)
```

Grid search

```
beta1.val = seq(0,2,0.02)
loglike.val = -apply(as.matrix(beta1.val), MARGIN = 1, FUN = log.likelihood.beta1, x=x, y=y)
beta1.max = beta1.val[loglike.val==max(loglike.val)]
```

Newton-Raphson method

```
NewtonRaph <- function(f = dl.dbeta1, tol = 1e-7, x0, N){
    h = 1e-7
    i = 1
    p = numeric(N)
    while(i<=N){
        f.prime = (f(x,y,0,x0+h)-f(x,y,0,x0))/h
        x1 = x0 - f(x,y,0,x0)/f.prime
        p[i] = x1
        i = i+1
        if(abs(x1-x0)<tol) break
        x0 = x1
    }
    return(p[i-1])
}
beta1.newton = NewtonRaph(x0 = 0, N = 100)
```

```
cat(paste("Uniroot:", round(beta1.uniroot$root,dig = 4)))
```

```
## Uniroot: 1.0219
```

```
cat(paste("Grid search:", round(beta1.max,dig = 4)))
```

```
## Grid search: 1.02
```

```
cat(paste("Newton-Raphson method:", round(beta1.newton,dig = 4)))
```

```
## Newton-Raphson method: 1.0219
```

Ex.2

Grid search

```
#Load the data
library("boot")
y = floor(coal[[1]])
y = tabulate(y)
y = y[1851:length(y)]

#Likelihood function
```

```
likelihood <- function(k,mu,lambda,y){
    y1 = y[1:k]
    y2 = y[-(1:k)]
    s1 = prod(dpois(y1,mu))
    s2 = prod(dpois(y2,lambda))
    return(s1*s2)
}
```

```
kval = 30:50
muval = seq(2,4,0.1)
lambdaval = seq(0.8,1,0.02)
par = expand.grid(kval,muval,lambdaval)
lval = c()
for(i in 1:dim(par)[1]){
    lval[i] = likelihood(par[i,1],par[i,2],par[i,3],y)
}
par.max = par[lval==max(lval),]
cat(paste("MLE of k:",round(par.max[1], dig = 4)))
```

```
## MLE of k: 41
```

```
cat(paste("MLE of mu:",round(par.max[2], dig = 4)))
```

```
## MLE of mu: 3.1
```

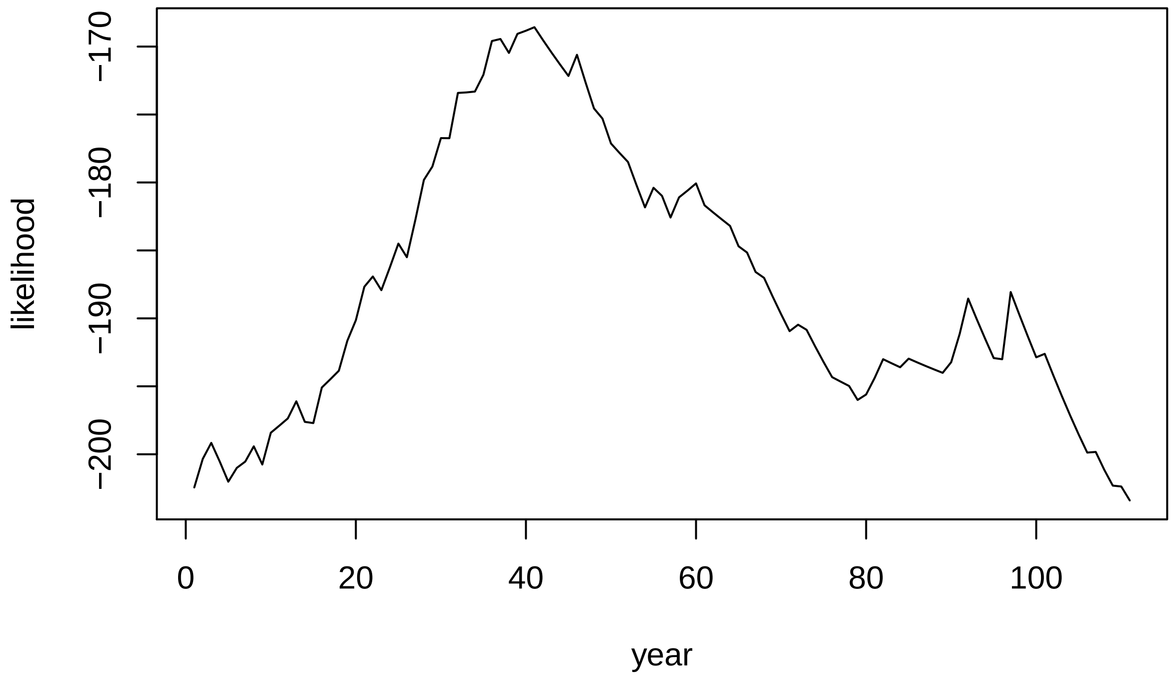
```
cat(paste("MLE of lambda:",round(par.max[3], dig = 4)))
```

```
## MLE of lambda: 0.9
```

MLE Solution

From STAT410, we know the MLE estimates for poisson distribution is the sample mean. We can get the μ_k and λ_k for every k and compute their corresponding log likelihood.

```
n = length(y)
mu = lambda = ll = numeric(n-1)
for(k in 1:(n-1)){
  y1 = y[1:k]
    y2 = y[(k+1):n]
  mu[k] = mean(y1)
  lambda[k] = mean(y2)
  #log transformation is monotonic
    ll[k] = log(prod(dpois(y1,mu[k]))*prod(dpois(y2,lambda[k])))
}
plot(1:(n-1),ll,"l",xlab="year",ylab="likelihood")
```




```
idx = which.max(ll)
cat(" MLE estimate\n k: ",idx, "\n mu: ",mu[idx], "\n lambda: ",lambda[idx],sep="")
```

```
##  MLE estimate
##  k: 41
##  mu: 3.097561
##  lambda: 0.9014085
```