Homework 6

Elena Volpi

11/26/2021

```
oilspills <- read.csv("~/Desktop/PCS/oilspills.dat", sep="")
```

#Problem 2.5

a) Derive the Newton-Raphson update for finding the MLE's α_1 and α_2

```
#Lab 8 code modified

# x = initial value
# itr = number of iterations to run
# x.values = contains values of x for each iteration
# g = objective function
# g.prime = first derivative of objective function
# g.2prime = second derivative of objective function

#Alpha_1 and Alpha_2 represent rate of spill occurence per Bbbl oil shipped during import/export and do

newton <- function(x,itr) {
  N <- oilspills[,2]  #Spills
  alpha_1 <- oilspills[,3] #import/export
  alpha_2 <- oilspills[,4] #domestic

  x.values = matrix(0,itr+2,2)
  x.values[1,] = x

  #objective fxn and derivatives
  g <- function(x){  #log-likelihood by-hand
    sum(N*log(x[1]*alpha_1 + x[2]*alpha_2)) - sum(x[1]*alpha_1 + x[2]*alpha_2) - sum(log(factorial(N)))
  }

  g.prime <- function(x){
    g.prime.da1 = sum((N*alpha_1)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_1) #partial wrt alpha_1
    g.prime.da2 = sum((N*alpha_2)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_2) #partial wrt alpha_2
    out = matrix(c(g.prime.da1,g.prime.da2),ncol=1)  #combine return as matrix
    return(out)
  }

  g.2prime=function(x){
    g.2prime.da1 = -sum((N*(alpha_1^2))/((x[1]*alpha_1 + x[2]*alpha_2)^2))  # da1da1
    g.2prime.da2 = -sum((N*(alpha_2^2))/((x[1]*alpha_1 + x[2]*alpha_2)^2)) #da2da2
    g.2prime.da1a2 = -sum((N*alpha_1*alpha_2)/((x[1]*alpha_1 + x[2]*alpha_2)^2))
```

```
    out = matrix(c(g.2prime.da1,g.2prime.da1a2,
                   g.2prime.da1a2,g.2prime.da2),nrow=2, byrow=TRUE)   #da1da2 (equaival to da2da1)
    return(out)
  }

  ## MAIN
  for(i in 1:itr){
    x = x - solve(g.2prime(x))%*%g.prime(x)
    x.values[i+1,] = x
  }


  ## OUTPUT
  print(x)      # FINAL ESTIMATE
  print(g(x))       # OBJECTIVE FUNCTION AT ESTIMATE
  print(g.prime(x))     # GRADIENT AT ESTIMATE

}
```

b) Derive the Fisher-Scoring update for finding MLEs of α_1 and α_2.

```
N <- oilspills[,2]  #Spills
alpha_1 <- oilspills[,3] #import/export
alpha_2 <- oilspills[,4] #domestic

#Need I matrix for Fishers, also used for steepest ascent
info_matrix <- function(x){
    info_11 <- sum((alpha_1^2)/(x[1]*alpha_1 + x[2]*alpha_2))
    info_12 <- sum((alpha_1*alpha_2)/(x[1]*alpha_1 + x[2]*alpha_2))
    info_22 <- sum((alpha_2^2)/(x[1]*alpha_1 + x[2]*alpha_2))
    out = matrix(c(info_11,info_12,info_12,info_22),nrow = 2, byrow = TRUE)
    return(out)
}

fishers <- function(x,itr) {
  #same initial values as above
  x.values = matrix(0,itr+2,2)
  x.values[1,] = x

  # same objective fxn and first derivative
  g <- function(x){  #log-likelihood by-hand
    sum(N*log(x[1]*alpha_1 + x[2]*alpha_2)) - sum(x[1]*alpha_1 + x[2]*alpha_2) - sum(log(factorial(N)))
  }

  g.prime <- function(x){
    g.prime.da1 = sum((N*alpha_1)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_1) #partial wrt alpha_1
    g.prime.da2 = sum((N*alpha_2)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_2) #partial wrt alpha_2
    out = matrix(c(g.prime.da1,g.prime.da2),ncol=1)  #combine return as matrix
    return(out)
  }


  ## MAIN
```

```r
  for(i in 1:itr){
    x = x + solve(info_matrix(x))%*%g.prime(x)
    x.values[i+1,] = x
  }

  ## OUTPUT
  print(x)      # FINAL ESTIMATE
  print(g(x))       # OBJECTIVE FUNCTION AT ESTIMATE
  print(g.prime(x))      # GRADIENT AT ESTIMATE
}
```

c)

```r
#Newton method
x = c(1,1)
itr = 40
newton(x,itr)
```

```
##           [,1]
## [1,] 1.0971525
## [2,] 0.9375546
## [1] -48.02716
##               [,1]
## [1,] -1.110223e-16
## [2,]  4.440892e-16
```

Our MLE for α_1 and α_2 are 1.097 and 0.938, respectively. It took three iterations for stability to the MLEs with the Newton-Raphson method.

```r
#Fishers Scoring
x = c(1,1)
itr = 40
fishers(x,itr)
```

```
##           [,1]
## [1,] 1.0971525
## [2,] 0.9375546
## [1] -48.02716
##               [,1]
## [1,] -6.661338e-16
## [2,]  0.000000e+00
```

We see the same MLEs but thirteen iterations for both MLE's to converge.

e) Apply the method of steepest ascent, use step-halving backtracking as necessary.

```r
# x = initial value
# M = Hessian approximation
# itr = number of iterations to run
# alpha = scale parameter
# x.values = contains values of x for each iteration
```

```
# g = objective function
# g.prime = first derivative of objective function

## INITIAL VALUES
x = c(1,1)
M = -info_matrix(x) #not totally sure if this is right
itr = 65
alpha.default = 1
alpha = alpha.default
x.values = matrix(0,itr+1,2)
x.values[1,] = x

## OBJECTIVE FUNCTION AND DERIVATIVES
g <- function(x){
  sum(N*log(x[1]*alpha_1 + x[2]*alpha_2)) - sum(x[1]*alpha_1 + x[2]*alpha_2) - sum(log(factorial(N)))
}


g.prime <- function(x){
    g.prime.da1 = sum((N*alpha_1)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_1)
    g.prime.da2 = sum((N*alpha_2)/(x[1]*alpha_1 + x[2]*alpha_2) - alpha_2)
    out = matrix(c(g.prime.da1,g.prime.da2),ncol=1)
    return(out)
}

## MAIN
for (i in 1:itr){
    hessian.inv = solve(M)
    xt = x - alpha*hessian.inv%*%g.prime(x)
    # REDUCE ALPHA UNTIL A CORRECT STEP IS REACHED
    while(g(xt) < g(x)){
            alpha = alpha/2
        xt = x - alpha*hessian.inv%*%g.prime(x)
    }
    x.values[i+1,] = x = xt
    alpha = alpha.default
}

## OUTPUT
x       # FINAL ESTIMATE
```

```
##           [,1]
## [1,] 1.0971525
## [2,] 0.9375546
```

```
g(x)        # OBJECTIVE FUNCTION AT ESTIMATE
```

```
## [1] -48.02716
```

```
g.prime(x)  # GRADIENT AT ESTIMATE
```

```
##                [,1]
```

```
## [1,] -3.885781e-16
## [2,] -5.551115e-16
```

This took 13 iterations to converge.

f)

```r
neg_likelihood <- function(x){
  -1 * (sum(N*log(x[1]*alpha_1 + x[2]*alpha_2)) - sum(x[1]*alpha_1 + x[2]*alpha_2) - sum(log(factorial(N
}

optim(par = c(1, 1), fn = neg_likelihood, method = "BFGS")
```

```
## $par
## [1] 1.0971528 0.9375544
##
## $value
## [1] 48.02716
##
## $counts
## function gradient
##       18        5
##
## $convergence
## [1] 0
##
## $message
## NULL
```

g) All methods gave the MLEs to be 1.0971525 and 0.9375546 for α_1 and α_2. The method with the least amount of iterations to convergence was the Newton-Raphson method with three iterations. Next was Fisher's scoring and steepest ascent with thirteen iterations. The method with the most iterations was the quasi-Newton's method.