GRADUATE STUDENT STAT 840 A3

Vsevolod Ladtchenko 20895137

Problem 2

a)

$$
\begin{aligned}
f(y) &= \binom{6}{y} p^y (1-p)^{6-y} \\
&= \binom{6}{y} p(\alpha,\beta,x)^y (1-p(\alpha,\beta,x))^{6-y} \\
Lik &= \prod_{i=1}^{n} \binom{6}{y_i} p(\alpha,\beta,x_i)^{y_i} (1-p(\alpha,\beta,x_i))^{6-y_i} \\
lik &= \sum_{i=1}^{n} \log \binom{6}{y_i} + y_i \log p(\alpha,\beta,x_i) + (6-y_i)\log(1-p(\alpha,\beta,x_i))
\end{aligned}
$$

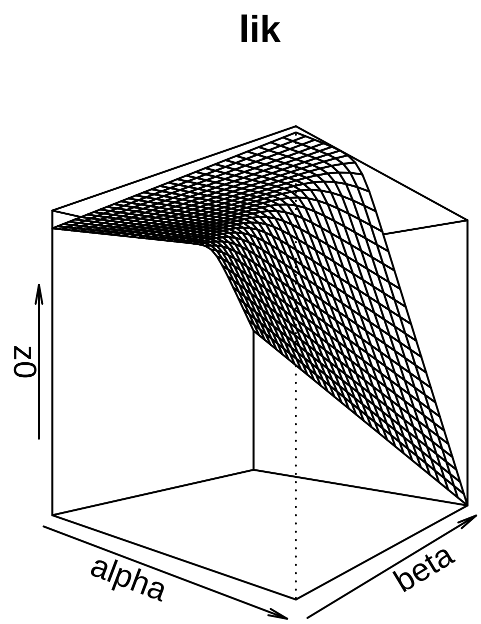
$$
\begin{aligned}
p(x) &= \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}} \frac{e^{-\alpha+-\beta x}}{e^{-\alpha+-\beta x}} \\
&= \frac{1}{1+e^{-\alpha-\beta x}} \\
\partial_\alpha p(x) &= \frac{e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2} \\
\partial_\beta p(x) &= \frac{x e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2}
\end{aligned}
$$

$$
\begin{aligned}
\partial_\alpha lik &= \sum_{i=1}^{n} \frac{y_i}{p(\alpha,\beta,x_i)} \partial_\alpha p(\alpha,\beta,x_i) - \frac{6-y_i}{1-p(\alpha,\beta,x_i)} \partial_\alpha p(\alpha,\beta,x_i) \\
&= \sum_{i=1}^{n} \frac{y_i}{p(\alpha,\beta,x_i)} \frac{e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2} - \frac{6-y_i}{1-p(\alpha,\beta,x_i)} \frac{e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2} \\
&= \sum_{i=1}^{n} y_i(1+e^{-\alpha-\beta x}) \frac{e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2} - (6-y_i) \frac{1+e^{-\alpha-\beta x}}{e^{-\alpha-\beta x}} \frac{e^{-\alpha-\beta x}}{(1+e^{-\alpha-\beta x})^2} \\
&= \sum_{i=1}^{n} \frac{y_i e^{-\alpha-\beta x}}{1+e^{-\alpha-\beta x}} - \frac{6-y_i}{1+e^{-\alpha-\beta x}} \\
&= \sum_{i=1}^{n} \frac{y_i e^{-\alpha-\beta x_i} - 6 + y_i}{1+e^{-\alpha-\beta x_i}}
\end{aligned}
$$

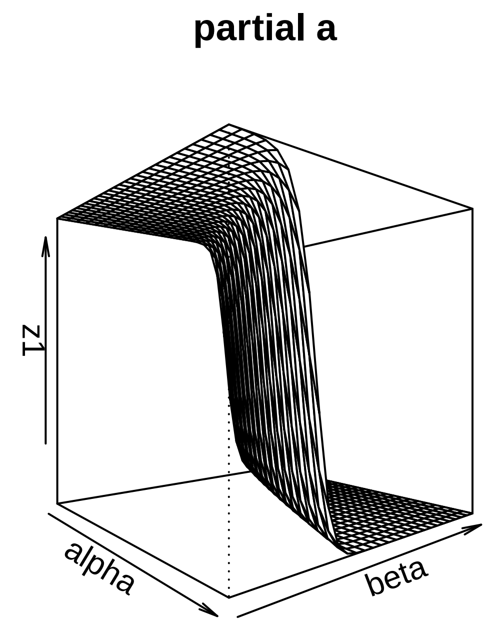
$$
\begin{aligned}
\partial_\beta lik &= \sum_{i=1}^{n} \frac{y_i}{p(\alpha,\beta,x_i)} \partial_\beta p(\alpha,\beta,x_i) - \frac{6-y_i}{1-p(\alpha,\beta,x_i)} \partial_\beta p(\alpha,\beta,x_i) \\
&= \sum_{i=1}^{n} \frac{y_i}{p(\alpha,\beta,x_i)} x_i \partial_\alpha p(\alpha,\beta,x_i) - \frac{6-y_i}{1-p(\alpha,\beta,x_i)} x_i \partial_\alpha p(\alpha,\beta,x_i) \\
&= \sum_{i=1}^{n} \frac{x_i(y_i e^{-\alpha-\beta x_i} - 6 + y_i)}{1+e^{-\alpha-\beta x_i}}
\end{aligned}
$$

```r
data_x = c(53,70,57,70,58,72,63,73,66,75,67,75,67,76,67,76,68,78,69,79,70,81,70)
data_y = c(2,1,1,1,1,0,1,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0)

p = function(a,b,x) 1 / (1 + exp(-a-b*x))

lik = function(a,b)
{
  l = 0
  for (i_ in 1:length(data_x))
  {
    xi = data_x[i_]
    yi = data_y[i_] # log(choose(6, yi)) +
    l = l + yi*log(p(a,b,xi)) + (6-yi)*log(1 - p(a,b,xi))
  }
  return(l)
}

partial_a = function(a,b)
{
  l = 0
  for (i_ in 1:length(data_x))
  {
    xi = data_x[i_]
    yi = data_y[i_]
    e_ = exp(-a-b*xi)
    l = l + (yi*e_ -6 +yi) / (1 + e_)
  }
  return(l)
}

partial_b = function(a,b)
{
  l = 0
  for (i_ in 1:length(data_x))
  {
    xi = data_x[i_]
    yi = data_y[i_]
    e_ = exp(-a-b*xi)
    l = l + (xi*(yi*e_ -6 +yi)) / (1 + e_)
  }
  return(l)
}

# we choose limits (-0.3, 0.3) for beta because larger causes overflow
```

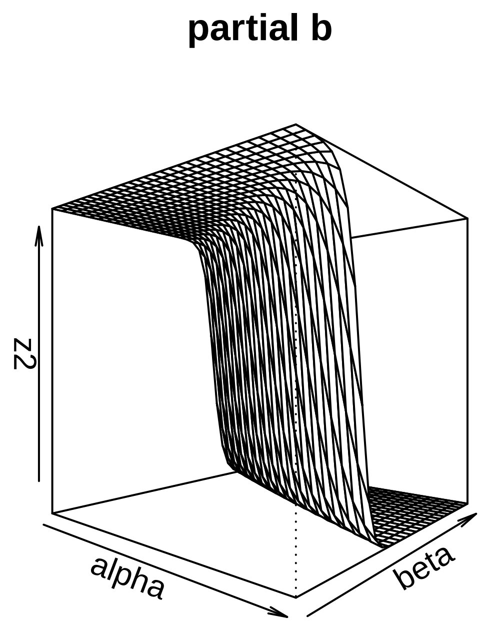
```
# similarly, choose limits (-10, 10) for alpha
N = 30
L = 0.1
x = seq(-L,L,length=N) * 100
y = seq(-L,L,length=N) * 3
z0 = outer(x,y,lik)
persp(x, y, z0, phi=0,theta=40, main='lik',xlab='alpha',ylab='beta')
```




```
z1 = outer(x,y,partial_a)
persp(x, y, z1, phi=0,theta=50, main='partial a',xlab='alpha',ylab='beta')
```




```
z2 = outer(x,y,partial_b)
persp(x, y, z2, phi=0,theta=40, main='partial b',xlab='alpha',ylab='beta')
```



b)

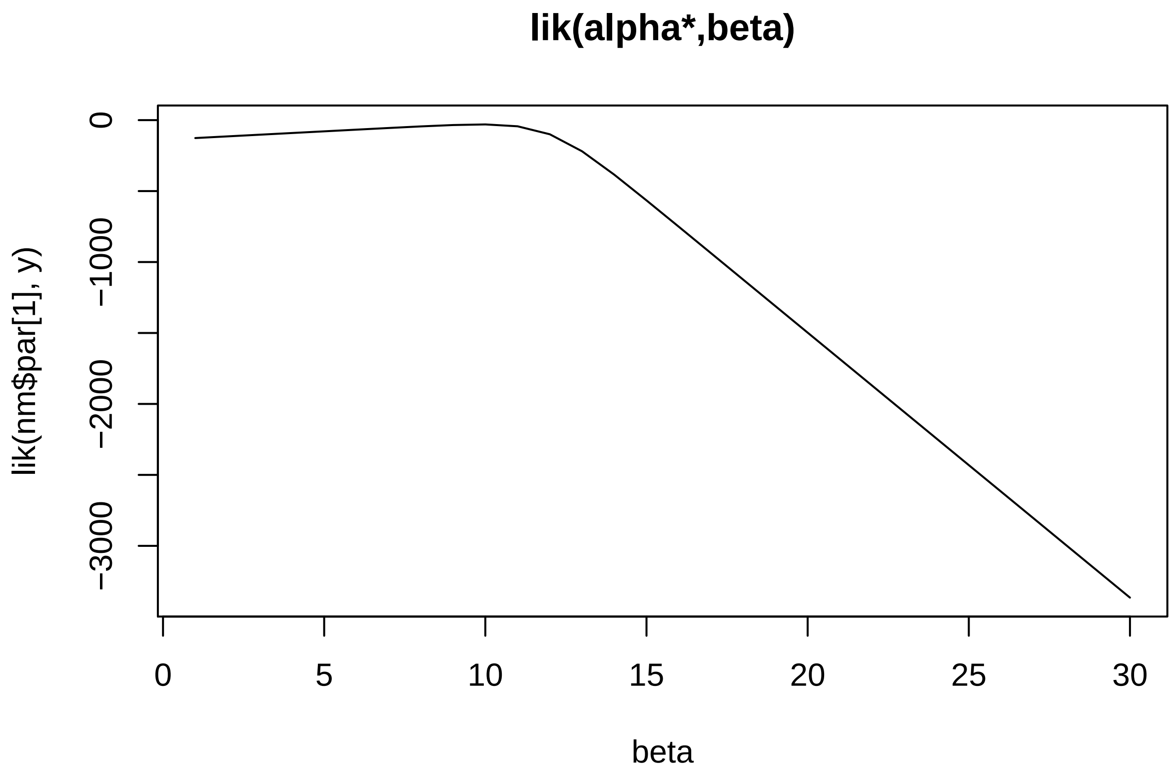
```
# for the optimization routine, make our likelihood negative to minimize.
lik_opt = function(x) -lik(x[1],x[2])

# try different starting point, in our grid
N = 10
L = 0.1
grid = matrix(nrow=N*N,ncol=3)
colnames(grid) = c('alpha','beta','lik')
xx = seq(-L,L,length=N) * 100
yy = seq(-L,L,length=N) * 3
for (i_ in 1:length(xx))
{
  for (j_ in 1:length(yy))
  {
    x_ = xx[i_]
    y_ = yy[j_]
    nm = optim(c(x_,y_), lik_opt, method="Nelder-Mead")
    grid[j_ + (i_-1)*N,] = c(x_, y_,-nm$value)
  }
}
grid
```

```
##              alpha        beta       lik
##   [1,] -10.000000 -0.30000000 -30.19817
```

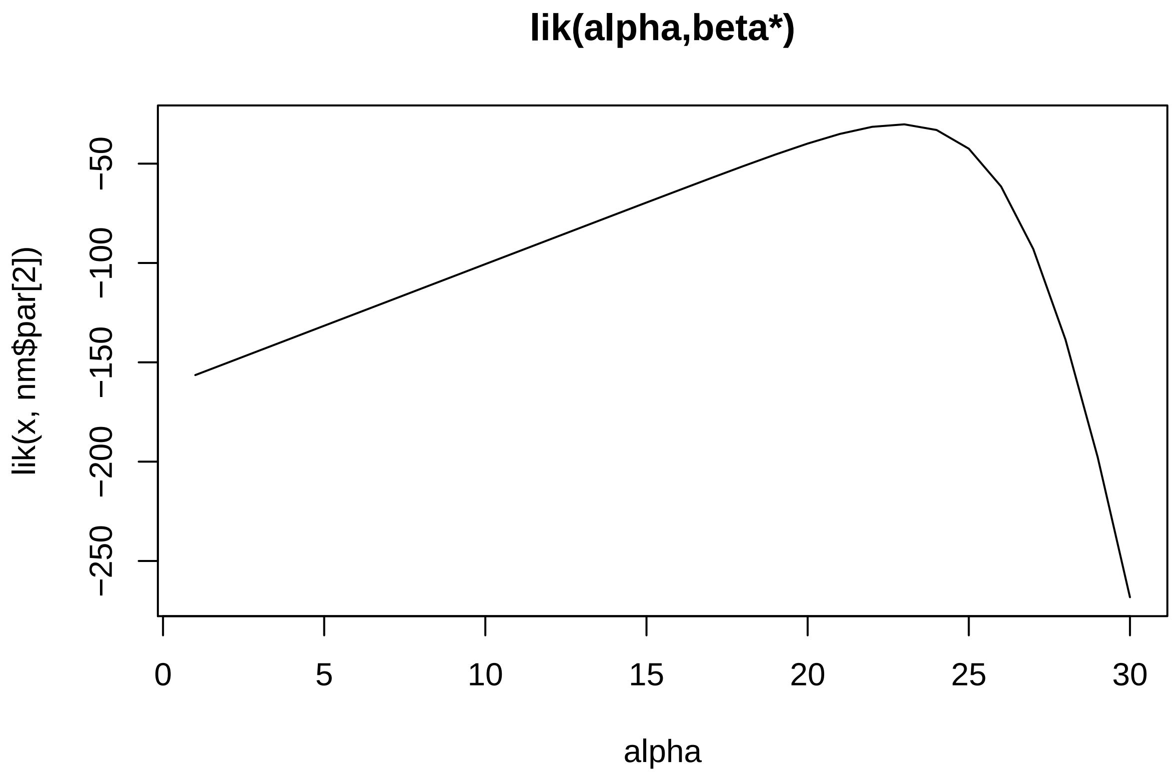
```
##   [2,] -10.000000 -0.23333333 -30.19818
##   [3,] -10.000000 -0.16666667 -30.19817
##   [4,] -10.000000 -0.10000000 -30.19817
##   [5,] -10.000000 -0.03333333 -30.19817
##   [6,] -10.000000  0.03333333 -30.19817
##   [7,] -10.000000  0.10000000 -30.19817
##   [8,] -10.000000  0.16666667 -30.19817
##   [9,] -10.000000  0.23333333 -30.19818
##  [10,] -10.000000  0.30000000 -30.19817
##  [11,]  -7.777778 -0.30000000 -30.19817
##  [12,]  -7.777778 -0.23333333 -30.19817
##  [13,]  -7.777778 -0.16666667 -30.19817
##  [14,]  -7.777778 -0.10000000 -30.19817
##  [15,]  -7.777778 -0.03333333 -30.19817
##  [16,]  -7.777778  0.03333333 -30.19817
##  [17,]  -7.777778  0.10000000 -30.19817
##  [18,]  -7.777778  0.16666667 -30.19817
##  [19,]  -7.777778  0.23333333 -30.19818
##  [20,]  -7.777778  0.30000000 -30.19818
##  [21,]  -5.555556 -0.30000000 -30.19817
##  [22,]  -5.555556 -0.23333333 -30.19817
##  [23,]  -5.555556 -0.16666667 -30.19817
##  [24,]  -5.555556 -0.10000000 -30.19817
##  [25,]  -5.555556 -0.03333333 -30.19817
##  [26,]  -5.555556  0.03333333 -30.19817
##  [27,]  -5.555556  0.10000000 -30.19817
##  [28,]  -5.555556  0.16666667 -30.19817
##  [29,]  -5.555556  0.23333333 -30.19818
##  [30,]  -5.555556  0.30000000 -30.19818
##  [31,]  -3.333333 -0.30000000 -30.19817
##  [32,]  -3.333333 -0.23333333 -30.19817
##  [33,]  -3.333333 -0.16666667 -30.19818
##  [34,]  -3.333333 -0.10000000 -30.19817
##  [35,]  -3.333333 -0.03333333 -30.19817
##  [36,]  -3.333333  0.03333333 -30.19817
##  [37,]  -3.333333  0.10000000 -30.19817
##  [38,]  -3.333333  0.16666667 -30.19818
##  [39,]  -3.333333  0.23333333 -30.19818
##  [40,]  -3.333333  0.30000000 -30.19818
##  [41,]  -1.111111 -0.30000000 -30.19817
##  [42,]  -1.111111 -0.23333333 -30.19817
##  [43,]  -1.111111 -0.16666667 -30.19817
##  [44,]  -1.111111 -0.10000000 -30.19817
##  [45,]  -1.111111 -0.03333333 -30.19817
##  [46,]  -1.111111  0.03333333 -30.19817
##  [47,]  -1.111111  0.10000000 -30.19817
##  [48,]  -1.111111  0.16666667 -30.19819
##  [49,]  -1.111111  0.23333333 -30.19819
##  [50,]  -1.111111  0.30000000 -30.19818
##  [51,]   1.111111 -0.30000000 -30.19817
##  [52,]   1.111111 -0.23333333 -30.19817
##  [53,]   1.111111 -0.16666667 -30.19817
##  [54,]   1.111111 -0.10000000 -30.19817
##  [55,]   1.111111 -0.03333333 -30.19817
```

```
##  [56,]   1.111111  0.03333333 -30.19818
##  [57,]   1.111111  0.10000000 -30.19818
##  [58,]   1.111111  0.16666667 -30.19819
##  [59,]   1.111111  0.23333333 -30.19818
##  [60,]   1.111111  0.30000000 -30.19822
##  [61,]   3.333333 -0.30000000 -30.19817
##  [62,]   3.333333 -0.23333333 -30.19817
##  [63,]   3.333333 -0.16666667 -30.19817
##  [64,]   3.333333 -0.10000000 -30.19817
##  [65,]   3.333333 -0.03333333 -30.19817
##  [66,]   3.333333  0.03333333 -30.19817
##  [67,]   3.333333  0.10000000 -30.19818
##  [68,]   3.333333  0.16666667 -30.28403
##  [69,]   3.333333  0.23333333 -30.19818
##  [70,]   3.333333  0.30000000 -30.33498
##  [71,]   5.555556 -0.30000000 -30.19817
##  [72,]   5.555556 -0.23333333 -30.19817
##  [73,]   5.555556 -0.16666667 -30.19817
##  [74,]   5.555556 -0.10000000 -30.19817
##  [75,]   5.555556 -0.03333333 -30.19817
##  [76,]   5.555556  0.03333333 -30.19817
##  [77,]   5.555556  0.10000000 -30.19818
##  [78,]   5.555556  0.16666667 -30.19818
##  [79,]   5.555556  0.23333333 -30.19827
##  [80,]   5.555556  0.30000000 -30.19820
##  [81,]   7.777778 -0.30000000 -30.19817
##  [82,]   7.777778 -0.23333333 -30.19817
##  [83,]   7.777778 -0.16666667 -30.19817
##  [84,]   7.777778 -0.10000000 -30.19817
##  [85,]   7.777778 -0.03333333 -30.19817
##  [86,]   7.777778  0.03333333 -30.19817
##  [87,]   7.777778  0.10000000 -30.19819
##  [88,]   7.777778  0.16666667 -30.19819
##  [89,]   7.777778  0.23333333 -30.19819
##  [90,]   7.777778  0.30000000 -30.19821
##  [91,]  10.000000 -0.30000000 -30.19817
##  [92,]  10.000000 -0.23333333 -30.19817
##  [93,]  10.000000 -0.16666667 -30.19817
##  [94,]  10.000000 -0.10000000 -30.19817
##  [95,]  10.000000 -0.03333333 -30.19817
##  [96,]  10.000000  0.03333333 -30.19818
##  [97,]  10.000000  0.10000000 -30.19817
##  [98,]  10.000000  0.16666667 -30.19821
##  [99,]  10.000000  0.23333333 -30.19822
## [100,]  10.000000  0.30000000 -30.19818
```

```r
# for added clarity we can look at the 2D plots
plot(lik(nm$par[1],y),type='l', main='lik(alpha*,beta)',xlab='beta')
```




```
plot(lik(x, nm$par[2]),type='l',main='lik(alpha,beta*)',xlab='alpha')
```



c)

$$\partial_{\alpha\alpha} lik = \sum_{i=1}^{n} \frac{-6e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2}$$
$$\partial_{\beta\beta} lik = \sum_{i=1}^{n} \frac{-6x_i^2 e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2}$$
$$\partial_{\alpha\beta} lik = \sum_{i=1}^{n} \frac{-6x_i e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2} = \partial_{\beta\alpha} lik$$

```
partial_a_opt = function(x) partial_a(x[1],x[2])
partial_b_opt = function(x) partial_b(x[1],x[2])
grad = function(x) c(partial_a_opt(x), partial_b_opt(x))

hess = function(pt)
{
  a = pt[1]
  b = pt[2]
  l1 = 0
  l2 = 0
  l3 = 0
  for (i_ in 1:length(data_x))
  {
    xi = data_x[i_]
    yi = data_y[i_]
```

```
    e_ = exp(-a-b*xi)
    p_aa = (-6*e_) / ((1 + e_)^2)
    p_ab = xi*p_aa
    p_bb = xi*p_ab
    l1 = l1 + p_aa
    l2 = l2 + p_ab
    l3 = l3 + p_bb
  }
  hess = matrix(data = c(l1,l2,l2,l3),nrow=2,ncol=2)
  return(hess)
}

hess_inv = function(pt)
{
  h = hess(pt)
  l1 = h[1,1]
  l2 = h[1,2]
  l3 = h[2,2]
  # inverse of 2x2 matrix
  det = l1*l3 - l2*l2
  hess = (1/det) * matrix(data = c(l3,-l2,-l2,l1),nrow=2,ncol=2)
  return(hess)
}

newton_step = function(old) # returns list of (point, flag)
{
  CALCULATION_IS_FINE = TRUE

  # grad
  g = matrix(grad(old), nrow=2,ncol=1)
  if (any(is.nan(g)) || any(is.infinite(g)))
  {
    #print("grad broken")
    CALCULATION_IS_FINE = FALSE
  }

  # solve system of equations
  if (FALSE)
  {
    # hessian
    h = hess(old)
    if (any(is.nan(h)) || any(is.infinite(h)))
    {
      #print("hess_inv broken")
      CALCULATION_IS_FINE = FALSE
    }

    ret = tryCatch(
      {
        list("point"=solve(h, -g) + old, "flag"=CALCULATION_IS_FINE)
      },
      error=function(cond) {
        message(cond)
```

```
        list("point"=matrix(nrow=2,ncol=1), "flag"=FALSE)
      },
      warning=function(cond) {
        message(cond)
        stop(cond)
      }
    )
    return(ret)
  }

  # matrix inverse method
  else
  {
    # hessian
    h = hess_inv(old)
    if (any(is.nan(h)) || any(is.infinite(h)))
    {
      #print("hess_inv broken")
      CALCULATION_IS_FINE = FALSE
    }

    # new point
    new = old - h %*% g
    return(list("point"=new, "flag"=CALCULATION_IS_FINE))
  }
}

newton_raphson = function(old)
{
  step_ = newton_step(old)
  new   = step_$point
  flag  = step_$flag
  if (flag == FALSE) # nan/inf
    return(step_)

  while (sum(abs(new-old)) > 0.001)
  {
    old   = new
    step_ = newton_step(old)
    new   = step_$point
    flag  = step_$flag
    if (flag == FALSE) # nan/inf
      break
  }
  return(step_)
}

# try different starting point, in our grid
grid = matrix(nrow=N*N,ncol=3)
colnames(grid) = c('alpha','beta','Newton')
STOP_LOOPING = FALSE
for (i_ in 1:length(xx))
{
```

```
  if (STOP_LOOPING == TRUE) break

  for (j_ in 1:length(yy))
  {
    if (STOP_LOOPING == TRUE) break

    x_ = xx[i_]
    y_ = yy[j_]
    newt_ = newton_raphson(c(x_,y_))
    optimum = newt_$point
    flag = newt_$flag
    #if (flag == FALSE) # nan/inf in newton

    lik_ = -lik_opt(optimum)
    grid[j_ + (i_-1)*N,] = c(x_, y_,lik_)
  }
}
grid
```

```
##              alpha        beta    Newton
##    [1,] -10.000000 -0.30000000       NaN
##    [2,] -10.000000 -0.23333333       NaN
##    [3,] -10.000000 -0.16666667       NaN
##    [4,] -10.000000 -0.10000000       NaN
##    [5,] -10.000000 -0.03333333       NaN
##    [6,] -10.000000  0.03333333       NaN
##    [7,] -10.000000  0.10000000 -30.19817
##    [8,] -10.000000  0.16666667       NaN
##    [9,] -10.000000  0.23333333       NaN
##   [10,] -10.000000  0.30000000       NaN
##   [11,]  -7.777778 -0.30000000       NaN
##   [12,]  -7.777778 -0.23333333       NaN
##   [13,]  -7.777778 -0.16666667       NaN
##   [14,]  -7.777778 -0.10000000       NaN
##   [15,]  -7.777778 -0.03333333       NaN
##   [16,]  -7.777778  0.03333333       NaN
##   [17,]  -7.777778  0.10000000 -30.19817
##   [18,]  -7.777778  0.16666667       NaN
##   [19,]  -7.777778  0.23333333       NaN
##   [20,]  -7.777778  0.30000000       NaN
##   [21,]  -5.555556 -0.30000000       NaN
##   [22,]  -5.555556 -0.23333333       NaN
##   [23,]  -5.555556 -0.16666667       NaN
##   [24,]  -5.555556 -0.10000000       NaN
##   [25,]  -5.555556 -0.03333333       NaN
##   [26,]  -5.555556  0.03333333 -30.19817
##   [27,]  -5.555556  0.10000000 -30.19817
##   [28,]  -5.555556  0.16666667       NaN
##   [29,]  -5.555556  0.23333333       NaN
##   [30,]  -5.555556  0.30000000       NaN
##   [31,]  -3.333333 -0.30000000       NaN
##   [32,]  -3.333333 -0.23333333       NaN
##   [33,]  -3.333333 -0.16666667       NaN
##   [34,]  -3.333333 -0.10000000       NaN
```

```
## [35,] -3.333333 -0.03333333       NaN
## [36,] -3.333333  0.03333333 -30.19817
## [37,] -3.333333  0.10000000       NaN
## [38,] -3.333333  0.16666667       NaN
## [39,] -3.333333  0.23333333       NaN
## [40,] -3.333333  0.30000000       NaN
## [41,] -1.111111 -0.30000000       NaN
## [42,] -1.111111 -0.23333333       NaN
## [43,] -1.111111 -0.16666667       NaN
## [44,] -1.111111 -0.10000000       NaN
## [45,] -1.111111 -0.03333333 -30.19817
## [46,] -1.111111  0.03333333 -30.19817
## [47,] -1.111111  0.10000000       NaN
## [48,] -1.111111  0.16666667       NaN
## [49,] -1.111111  0.23333333       NaN
## [50,] -1.111111  0.30000000       NaN
## [51,]  1.111111 -0.30000000       NaN
## [52,]  1.111111 -0.23333333       NaN
## [53,]  1.111111 -0.16666667       NaN
## [54,]  1.111111 -0.10000000       NaN
## [55,]  1.111111 -0.03333333 -30.19817
## [56,]  1.111111  0.03333333       NaN
## [57,]  1.111111  0.10000000       NaN
## [58,]  1.111111  0.16666667       NaN
## [59,]  1.111111  0.23333333       NaN
## [60,]  1.111111  0.30000000       NaN
## [61,]  3.333333 -0.30000000       NaN
## [62,]  3.333333 -0.23333333       NaN
## [63,]  3.333333 -0.16666667       NaN
## [64,]  3.333333 -0.10000000 -30.19817
## [65,]  3.333333 -0.03333333 -30.19817
## [66,]  3.333333  0.03333333       NaN
## [67,]  3.333333  0.10000000       NaN
## [68,]  3.333333  0.16666667       NaN
## [69,]  3.333333  0.23333333       NaN
## [70,]  3.333333  0.30000000       NaN
## [71,]  5.555556 -0.30000000       NaN
## [72,]  5.555556 -0.23333333       NaN
## [73,]  5.555556 -0.16666667       NaN
## [74,]  5.555556 -0.10000000 -30.19817
## [75,]  5.555556 -0.03333333       NaN
## [76,]  5.555556  0.03333333       NaN
## [77,]  5.555556  0.10000000       NaN
## [78,]  5.555556  0.16666667       NaN
## [79,]  5.555556  0.23333333       NaN
## [80,]  5.555556  0.30000000       NaN
## [81,]  7.777778 -0.30000000       NaN
## [82,]  7.777778 -0.23333333       NaN
## [83,]  7.777778 -0.16666667 -30.19817
## [84,]  7.777778 -0.10000000 -30.19817
## [85,]  7.777778 -0.03333333       NaN
## [86,]  7.777778  0.03333333       NaN
## [87,]  7.777778  0.10000000       NaN
## [88,]  7.777778  0.16666667       NaN
```

```
## [89,]   7.777778  0.23333333       NaN
## [90,]   7.777778  0.30000000       NaN
## [91,]  10.000000 -0.30000000       NaN
## [92,]  10.000000 -0.23333333       NaN
## [93,]  10.000000 -0.16666667 -30.19817
## [94,]  10.000000 -0.10000000       NaN
## [95,]  10.000000 -0.03333333       NaN
## [96,]  10.000000  0.03333333       NaN
## [97,]  10.000000  0.10000000       NaN
## [98,]  10.000000  0.16666667       NaN
## [99,]  10.000000  0.23333333       NaN
## [100,]  10.000000  0.30000000       NaN
```

Looking at our results with Newton-Raphson, we see that this algorithm's success is very dependent on the initial point. When it works, it works well. Due to the algorithm depending on the first and second derivatives, for this problem it is prone to floating point overflows due to the large numbers involved with exponents and logarithms. In part (b), the Nelder-Mead algorithm was able to succeed at every starting point (although we picked the intervals to be admissible to begin with), but using these same intervals with Newton, only a small percentage were successful. Although it boasts quadratic convergence, this algorithm needs to be chosen with a compatible starting point, as well as other numerical considerations, which is a lot of work during which the Nelder-Mead algorithm can find the optimum instead. Additionally, we tried both inverting the Hessian and solving a system of linear equations, and the results were the same in terms of NaNs produced.

d)

In the problem formulation, y_i is a random variable, which is the number of failed O-rings, and it follows a binomial distribution. We are modeling this distribution as having parameter $p(\alpha, \beta, x_i)$ dependent on the weather x_i. Since x_i is a predictor, it is not a random variable. $\alpha.\beta$ are parameters of this function. Thus the only random variable here is y_i.

Fisher scoring requires us to take the Expectation of the Hessian (notes 4 p 25), apply a negative sign, and invert it. If the Hessian is not random, this simply becomes the Newton-Raphson algorithm.

Looking again at our second derivatives (replicated below for convenience) we see that y_i is not present. Thus when we take their expectation, we get the same result back because it is a constant with respect to the Expectation operation. Thus in this case Fisher Scoring will be equivalent to Newton-Raphson. Therefore, the same exact initial points will lead to the same exact solutions for the MLE as in part (c).

$$
\begin{aligned}
\partial_{\alpha\alpha} lik &= \sum_{i=1}^{n} \frac{-6e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2} \\
\partial_{\beta\beta} lik &= \sum_{i=1}^{n} \frac{-6x_i^2 e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2} \\
\partial_{\alpha\beta} lik &= \sum_{i=1}^{n} \frac{-6x_i e^{-\alpha-\beta x_i}}{(1+e^{-\alpha-\beta x_i})^2} = \partial_{\beta\alpha} lik
\end{aligned}
$$

This procedure is identical to part (c) because the Hessian is not random, and its expectation is the same as the Hessian itself.

$$
\begin{aligned}
(\mathbb{E}[-\nabla^2 l(\theta^{(k)})])^{-1} &= (-\nabla^2 l(\theta^{(k)}))^{-1} \\
&= (-\mathbf{H}_l(\theta^{(k)}))^{-1} \\
&= -\mathbf{H}_l^{-1}(\theta^{(k)})
\end{aligned}
$$

e)

```r
# to feed gradient to optim, we add negative since its minimizing lik
grad_bfgs = function(x) -c(partial_a_opt(x), partial_b_opt(x))

grid = matrix(nrow=N*N,ncol=3)
colnames(grid) = c('alpha','beta','lik')
for (i_ in 1:length(xx))
{
  for (j_ in 1:length(yy))
  {
    x_ = xx[i_]
    y_ = yy[j_]
    o_ = optim(c(x_,y_), lik_opt, gr=grad_bfgs,method="BFGS")
    grid[j_ + (i_-1)*N,] = c(x_, y_,-o_$value)
  }
}
grid
```

```
##            alpha        beta       lik
##   [1,] -10.000000 -0.30000000 -30.19817
##   [2,] -10.000000 -0.23333333 -30.19817
##   [3,] -10.000000 -0.16666667 -30.19817
##   [4,] -10.000000 -0.10000000 -30.19817
##   [5,] -10.000000 -0.03333333 -30.19817
##   [6,] -10.000000  0.03333333 -30.19817
##   [7,] -10.000000  0.10000000 -30.19817
##   [8,] -10.000000  0.16666667 -30.19817
##   [9,] -10.000000  0.23333333 -30.19817
##  [10,] -10.000000  0.30000000 -30.19817
##  [11,]  -7.777778 -0.30000000 -30.19817
##  [12,]  -7.777778 -0.23333333 -30.19817
##  [13,]  -7.777778 -0.16666667 -30.19817
##  [14,]  -7.777778 -0.10000000 -30.19817
##  [15,]  -7.777778 -0.03333333 -30.19817
##  [16,]  -7.777778  0.03333333 -30.19817
##  [17,]  -7.777778  0.10000000 -30.19817
##  [18,]  -7.777778  0.16666667 -30.19817
##  [19,]  -7.777778  0.23333333 -30.19817
##  [20,]  -7.777778  0.30000000 -30.19817
##  [21,]  -5.555556 -0.30000000 -30.19817
##  [22,]  -5.555556 -0.23333333 -30.19817
##  [23,]  -5.555556 -0.16666667 -30.19817
##  [24,]  -5.555556 -0.10000000 -30.19817
##  [25,]  -5.555556 -0.03333333 -30.19817
##  [26,]  -5.555556  0.03333333 -30.19817
##  [27,]  -5.555556  0.10000000 -30.19817
##  [28,]  -5.555556  0.16666667 -30.19817
##  [29,]  -5.555556  0.23333333 -30.19817
##  [30,]  -5.555556  0.30000000 -30.19817
##  [31,]  -3.333333 -0.30000000 -30.19817
##  [32,]  -3.333333 -0.23333333 -30.19817
##  [33,]  -3.333333 -0.16666667 -30.19817
##  [34,]  -3.333333 -0.10000000 -30.19817
```

```
## [35,]  -3.333333 -0.03333333 -30.19817
## [36,]  -3.333333  0.03333333 -30.19817
## [37,]  -3.333333  0.10000000 -30.19817
## [38,]  -3.333333  0.16666667 -30.19817
## [39,]  -3.333333  0.23333333 -30.19817
## [40,]  -3.333333  0.30000000 -30.19817
## [41,]  -1.111111 -0.30000000 -30.19817
## [42,]  -1.111111 -0.23333333 -30.19817
## [43,]  -1.111111 -0.16666667 -30.19817
## [44,]  -1.111111 -0.10000000 -30.19817
## [45,]  -1.111111 -0.03333333 -30.19817
## [46,]  -1.111111  0.03333333 -30.19817
## [47,]  -1.111111  0.10000000 -30.19817
## [48,]  -1.111111  0.16666667 -30.19817
## [49,]  -1.111111  0.23333333 -30.19817
## [50,]  -1.111111  0.30000000 -30.19817
## [51,]   1.111111 -0.30000000 -30.19817
## [52,]   1.111111 -0.23333333 -30.19817
## [53,]   1.111111 -0.16666667 -30.19817
## [54,]   1.111111 -0.10000000 -30.19817
## [55,]   1.111111 -0.03333333 -30.19817
## [56,]   1.111111  0.03333333 -30.19817
## [57,]   1.111111  0.10000000 -30.19817
## [58,]   1.111111  0.16666667 -30.19817
## [59,]   1.111111  0.23333333 -30.19817
## [60,]   1.111111  0.30000000 -30.19817
## [61,]   3.333333 -0.30000000 -30.19817
## [62,]   3.333333 -0.23333333 -30.19817
## [63,]   3.333333 -0.16666667 -30.19817
## [64,]   3.333333 -0.10000000 -30.19817
## [65,]   3.333333 -0.03333333 -30.19817
## [66,]   3.333333  0.03333333 -30.19817
## [67,]   3.333333  0.10000000 -30.19817
## [68,]   3.333333  0.16666667 -30.19817
## [69,]   3.333333  0.23333333 -30.19817
## [70,]   3.333333  0.30000000 -30.19817
## [71,]   5.555556 -0.30000000 -30.19817
## [72,]   5.555556 -0.23333333 -30.19817
## [73,]   5.555556 -0.16666667 -30.19817
## [74,]   5.555556 -0.10000000 -30.19817
## [75,]   5.555556 -0.03333333 -30.19817
## [76,]   5.555556  0.03333333 -30.19817
## [77,]   5.555556  0.10000000 -30.19817
## [78,]   5.555556  0.16666667 -30.19817
## [79,]   5.555556  0.23333333 -30.19817
## [80,]   5.555556  0.30000000 -30.19817
## [81,]   7.777778 -0.30000000 -30.19817
## [82,]   7.777778 -0.23333333 -30.19817
## [83,]   7.777778 -0.16666667 -30.19817
## [84,]   7.777778 -0.10000000 -30.19817
## [85,]   7.777778 -0.03333333 -30.19817
## [86,]   7.777778  0.03333333 -30.19817
## [87,]   7.777778  0.10000000 -30.19817
## [88,]   7.777778  0.16666667 -30.19817
```

```
##  [89,]   7.777778  0.23333333 -30.19817
##  [90,]   7.777778  0.30000000 -30.19817
##  [91,]  10.000000 -0.30000000 -30.19817
##  [92,]  10.000000 -0.23333333 -30.19817
##  [93,]  10.000000 -0.16666667 -30.19817
##  [94,]  10.000000 -0.10000000 -30.19817
##  [95,]  10.000000 -0.03333333 -30.19817
##  [96,]  10.000000  0.03333333 -30.19817
##  [97,]  10.000000  0.10000000 -30.19817
##  [98,]  10.000000  0.16666667 -30.19817
##  [99,]  10.000000  0.23333333 -30.19817
## [100,]  10.000000  0.30000000 -30.19817
```