Rate of Convergence for Newton and Fisher's scoring Method

We are given the following random numbers **11 12 8 5 11** drawn from $Poisson(\lambda)$
The following code will randomly select one number from the given numbers as our intial guess.

```
x = c(11, 12, 8,5, 11)
a = c(1, 2, 3, 4, 5)
start_value = sample(a, 1)
n = length(x)
```

The idea

* Find the Likelihood $L(\lambda) = \frac{\lambda^{\sum x_i}}{\prod x_i!}$

- Find the log-Likelihood $l(\lambda) = \sum x_i \ * \ ln(\lambda) - n\lambda$
- Find the Score equation $l'(\lambda) = \frac{\sum x_i}{\lambda} - n = 0$. This is the equation that we want to solve.
- Find the double derivative of the log-Likelihood $l''(\lambda) = -\frac{\sum x_i}{\lambda^2}$

R implementation

a.

```
x = c(11, 12, 8,5, 11)
a = c(1, 2, 3, 4, 5)
start_value = sample(a, 1)
n = length(x)

# Score
l_p_lambda = function(lambda) {
  sum(x)/lambda - n
}

# Double differentiation
l_pp_lambda = function(lambda) {
  -sum(x)/lambda^2
}

l_old = start_value
dif = 1

while(dif > 0.00001){
  l_new = l_old + l_p_lambda(l_old)/(-l_pp_lambda(l_old))
  dif = abs(l_new - l_old)
  l_old = l_new
}

l_old
```

```
## [1] 9.4
```

b.

Applying the expectation to the double differentiated log-likelihood we get the following **Fisher Information**.

$$-E[l''(\lambda)] = \frac{n}{\lambda}$$

```
x = c(11, 12, 8,5, 11)
a = c(1, 2, 3, 4, 5)
start_value = sample(a, 1)
n = length(x)

# Score
l_p_lambda = function(lambda) {
  sum(x)/lambda - n
}

# Double differentiation
fisher = function(lambda) {
  n/lambda
}

l_old = start_value
dif = 1

while(dif > 0.00001){
  l_new = l_old + l_p_lambda(l_old)/(fisher(l_old))
  dif = abs(l_new - l_old)
  l_old = l_new
}

l_old
```

```
## [1] 9.4
```

c.

One of the of main drawbacks of the **N-R** algorithm is that it fails to converge if our starting initial guess is not close enough to our convergence value. We are able to see this if we pick our initial guess as **20** as shown below, which as you can see is far away from our convergence value **9.4**. By picking the value **20** we observe two things, one is that **N-R** fails to converge and the other observation we make is that our **Fisher's scoring method** converges in just 2 iterations. This is phenomenally faster than the **N-S method**

```
## [1] "N-S Convergence"
```

```
## [1] "iternation:  1"
## [1] -2.553191
## [1] "iternation:  2"
## [1] -5.799871
## [1] "iternation:  3"
## [1] -15.17831
## [1] "iternation:  4"
## [1] -54.86523
## [1] "iternation:  5"
## [1] -429.9638
## [1] "iternation:  6"
## [1] -20526.82
## [1] "iternation:  7"
## [1] -44865577
## [1] "iternation:  8"
## [1] -2.141405e+14
## [1] "iternation:  9"
## [1] -4.878315e+27
## [1] "iternation:  10"
## [1] -2.531697e+54
## [1] "iternation:  11"
## [1] -6.818608e+107
## [1] "iternation:  12"
## [1] -4.946108e+214
## [1] "iternation:  13"
## [1] -Inf
```

```
## [1] "Fisher Convergence"
```

```
## [1] "iternation:  1"
## [1] 9.4
## [1] "iternation:  2"
## [1] 9.4
```

Picking another starting value **10** we show that the **Fisher's scoring method** is much faster than **N-S** by converging in just 2 (can also say 1) iterations while **N-S** taking 4 iterations.

```
## [1] "N-S Convergence"
```

```
## [1] "iternation:  1"
## [1] 9.361702
## [1] "iternation:  2"
## [1] 9.399844
## [1] "iternation:  3"
## [1] 9.4
## [1] "iternation:  4"
## [1] 9.4
```

```
## [1] "Fisher Convergence"
```

```
## [1] "iternation:  1"
## [1] 9.4
## [1] "iternation:  2"
## [1] 9.4
```

The Code For Part C:

```
start_value = 10 #Start value must be adjusted manually

print("N-S Convergence")
  #N-S
  x = c(11, 12, 8,5, 11)
  n = length(x)

  # Score
  l_p_lambda = function(lambda) {
    sum(x)/lambda - n
  }

  # Double differentiation
  l_pp_lambda = function(lambda) {
    -sum(x)/lambda^2
  }

  l_old = start_value
  dif = 1
  iter = 1

  while(dif > 0.00001 && iter < 14){
    l_new = l_old + l_p_lambda(l_old)/(-l_pp_lambda(l_old))
    print(paste("iternation: ", iter))
    print(l_new)
    dif = abs(l_new - l_old)
    l_old = l_new
    iter = iter + 1

  }

  print("Fisher Convergence")

  # fisher Convergence
  x = c(11, 12, 8,5, 11)
  n = length(x)

  # Score
  l_p_lambda = function(lambda) {
    sum(x)/lambda - n
  }

  # fisher
  fisher = function(lambda) {
    n/lambda
  }

  l_old = start_value
  dif = 1
  iter = 1

  while(dif > 0.00001){
    l_new = l_old + l_p_lambda(l_old)/(fisher(l_old))
    print(paste("iternation: ", iter))
    print(l_new)
```

```
    dif = abs(l_new - l_old)
    l_old = l_new
    iter = iter + 1
  }
```