HW3

Cheng Huang[*]

21 September 2018

Abstract

Consider estimating the location parameter of a Cauchy distribution with a known scale parameter.

Derivation 1

$$
\begin{aligned}
\ell(\theta) &= ln \prod_{i=1}^{n} f(x_i;\theta) = ln \prod_{i=1}^{n} \frac{1}{\pi[1+(x_i-\theta)^2]} = -n\ln\pi - \sum_{i=1}^{n} \ln[1+(\theta-X_i)^2], \\
\ell'(\theta) &= -\sum_{i=1}^{n} \frac{2(\theta-X_i)}{1+(\theta-X_i)^2} \\
&= -2\sum_{i=1}^{n} \frac{\theta-X_i}{1+(\theta-X_i)^2}, \\
\ell''(\theta) &= -2\sum_{i=1}^{n} \frac{(1+(\theta-X_i)^2-2(\theta-X_i)^2)}{[1+(\theta-X_i)^2]^2} \\
&= -2\sum_{i=1}^{n} \frac{1-(\theta-X_i)^2}{[1+(\theta-X_i)^2]^2}, \\
I_n(\theta) &= -E[(\ell''(\theta))|\theta] \\
&= 2nE[\frac{1-(\theta-X_i)^2}{(1+(\theta-X_i)^2)^2}] \\
&= 2n\int_R \frac{1-(\theta-X_i)^2}{(1+(\theta-X_i)^2)^2} \frac{1}{\pi[1+(\theta-X_i)^2]} dx \\
&= \frac{4n}{\pi}\int_{-\infty}^{\infty} \frac{x^2\,dx}{(1+x^2)^3} \\
&= \frac{8n}{\pi}\int_{0}^{\infty} \frac{x^2\,dx}{(1+x^2)^3}
\end{aligned}
$$

let $y = \frac{1}{(1+x^2)}$, $dy = -2x(1+x^2)^{-2}dx$, so

$$
\begin{aligned}
I_n(\theta) &= \frac{4n}{\pi}\int_0^1 y(\frac{1}{y-1})^{1/2} dy \\
&= \frac{4n}{\pi}\frac{\Gamma(1.5)\Gamma(1.5)}{\Gamma(3)} \\
&= \frac{4n}{\pi}\frac{\pi}{8} \\
&= \frac{n}{2}
\end{aligned}
$$

[*]; Ph.D. student at Department of Statistics, University of Connecticut.

where I_n is the Fisher information of this sample.

generate a random sample

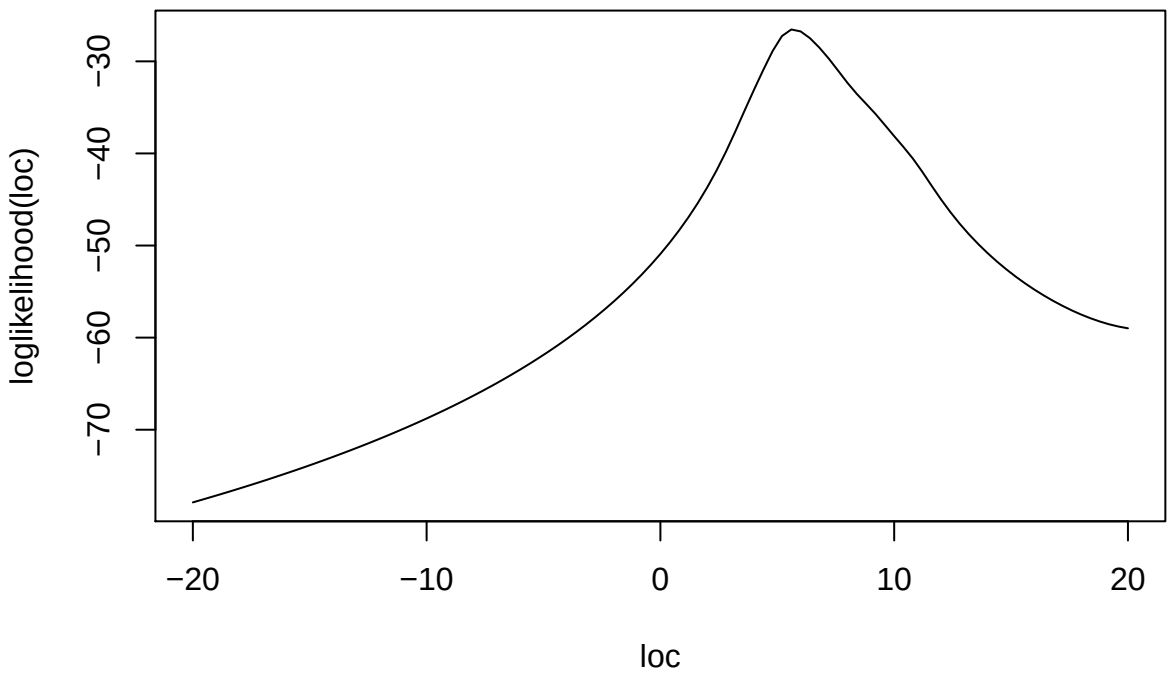
Generate a sample with $n = 10$ and $\theta = 5$.

```
set.seed(20180909)
n <- 10
par.loc <- 5
data.cauchy <- rcauchy(n, location = par.loc)
```

Implement a loglikelihood function and plot against θ.

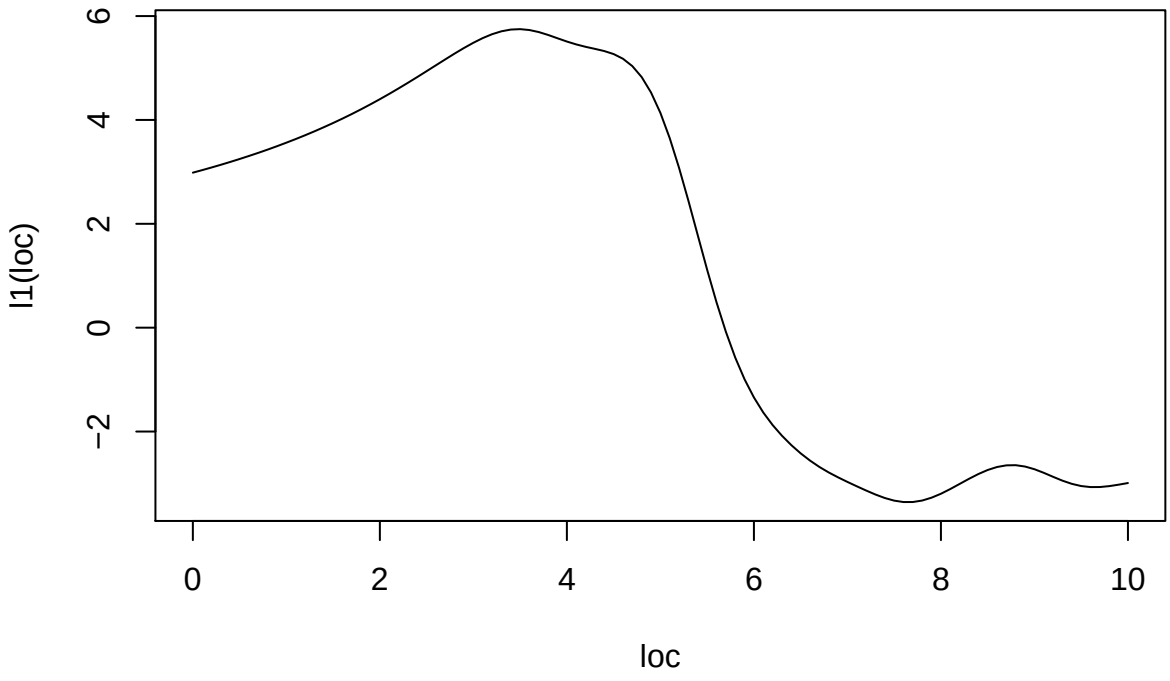
```
llk <- function(loc) {
  sum(dcauchy(data.cauchy,
              location = loc,
              log = TRUE))
}

loglikelihood <- Vectorize(llk)
curve(loglikelihood, -20, 20, xname = 'loc')
```



Newton–Raphson method

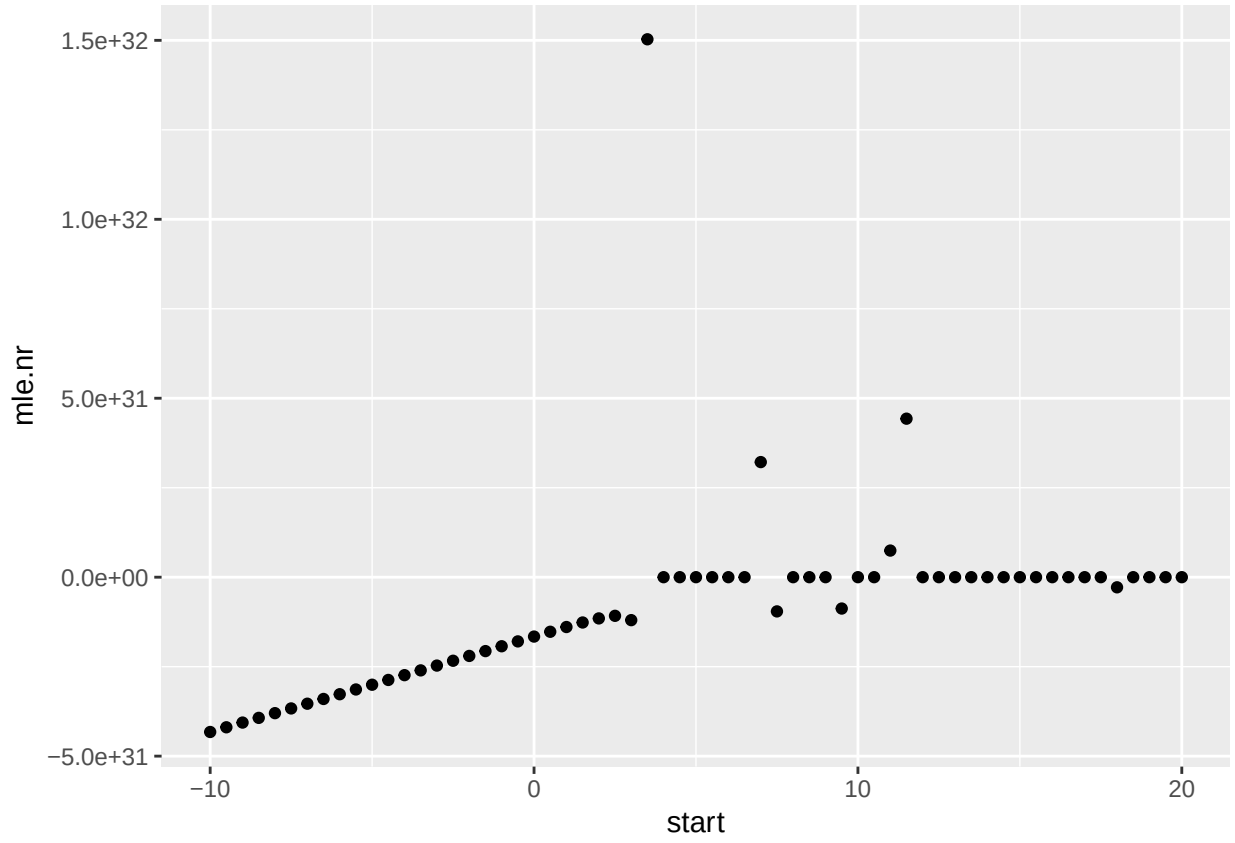
```
require(ggplot2)
start <- seq(-10, 20, by=0.5)
# newton.method require(animation)
# newtonRaphson require(pracma)
llk.1st <- function(x) {
  -2 * sum((x - data.cauchy)/(1 + (x - data.cauchy) ^ 2))
}
l1 <- Vectorize(llk.1st)
curve(l1, 0, 10, xname = 'loc')
```




```
llk.2nd <- function(x){
  -2 * sum((1 - (x - data.cauchy) ^ 2)/((1 + (x - data.cauchy) ^ 2) ^ 2))
}
## l2 <- Vectorize(llk.2nd)
## curve(l2, -10, 20, xname = 'loc')
mle.nr <- double(length(start))
iter.nr <- double(length(start)) ## record iteration time
for (i in start){
  NR <- newtonRaphson(fun = llk.1st, x0 = i, dfun = llk.2nd, maxiter = 100)
```

```
  mle.nr[which(start == i)] <- NR$root
  iter.nr[which(start == i)] <- NR$niter
}

dt <- data.table(start.value = start, mle = mle.nr, iter.nr = iter.nr)
ggplot(dt, aes(x=start, y=mle.nr)) + geom_point()
```




```
kable(dt)
```

start.value	mle	iter.nr
-10.0	-4.324741e+31	101
-9.5	-4.193577e+31	101
-9.0	-4.062249e+31	101
-8.5	-3.930748e+31	101
-8.0	-3.799064e+31	101
-7.5	-3.667185e+31	101
-7.0	-3.535100e+31	101
-6.5	-3.402796e+31	101
-6.0	-3.270261e+31	101
-5.5	-3.137479e+31	101
-5.0	-3.004436e+31	101

start.value	mle	iter.nr
-4.5	-2.871118e+31	101
-4.0	-2.737510e+31	101
-3.5	-2.603599e+31	101
-3.0	-2.469374e+31	101
-2.5	-2.334832e+31	101
-2.0	-2.199981e+31	101
-1.5	-2.064850e+31	101
-1.0	-1.929508e+31	101
-0.5	-1.794100e+31	101
0.0	-1.658922e+31	101
0.5	-1.524582e+31	101
1.0	-1.392396e+31	101
1.5	-1.265439e+31	101
2.0	-1.151924e+31	101
2.5	-1.079358e+31	101
3.0	-1.199750e+31	101
3.5	1.502957e+32	101
4.0	2.056366e+01	101
4.5	2.108229e+01	101
5.0	5.685422e+00	6
5.5	5.685422e+00	4
6.0	5.685422e+00	5
6.5	5.685422e+00	8
7.0	3.215974e+31	101
7.5	-9.558888e+30	101
8.0	1.937744e+01	101
8.5	2.108229e+01	101
9.0	5.685422e+00	7
9.5	-8.759488e+30	101
10.0	2.108229e+01	101
10.5	5.685422e+00	6
11.0	7.439560e+30	101
11.5	4.429077e+31	101
12.0	2.056366e+01	101
12.5	2.056366e+01	101
13.0	2.056366e+01	101
13.5	2.108229e+01	101
14.0	2.108229e+01	101
14.5	2.108230e+01	101
15.0	1.937743e+01	101
15.5	2.056366e+01	101
16.0	2.056366e+01	101
16.5	1.937744e+01	101
17.0	1.937743e+01	101
17.5	1.937744e+01	101
18.0	-2.825479e+30	101

start.value	mle	iter.nr
18.5	2.056366e+01	101
19.0	2.056366e+01	101
19.5	2.056366e+01	101
20.0	2.056366e+01	101

```
## data.table(cn = names(dt), transpose(dt))
```

Fixed Point Iteration

```
## Function calculate Fixed Point Iteration estimates
FPI <- function(fn, x0, tol = .Machine$double.eps^0.5, maxiter = 100){
    xold <- x0
    xnew <- fn(xold)
    iter <- 1
    while ((abs(xnew - xold) > tol) && (iter < maxiter)) {
        xold <- xnew
        xnew <- fn(xold)
        iter <- iter + 1
    }
    return(list(root = xnew, iter = iter))
}
## function: return result in datatable
rs <- function(alpha, start, ...){
  return(data.table(alpha = alpha,
                    start = start,
                    estimates = FPI(..., x0 = start)$root,
                    iter.fpi = FPI(..., x0 = start)$iter))
}
alpha <- c(1, 0.64, 0.25)

for (i in alpha){
  G <- function(x){
    i * llk.1st(x) + x
  }
    for (j in start){
      if(i == alpha[1] & j ==start[1]){
      tble <- rs(i, j, G)
    }else{
      dt <- rs(i, j, G)
      tble <- rbind(tble, dt)
    }
  }
}
kable(tble)
```

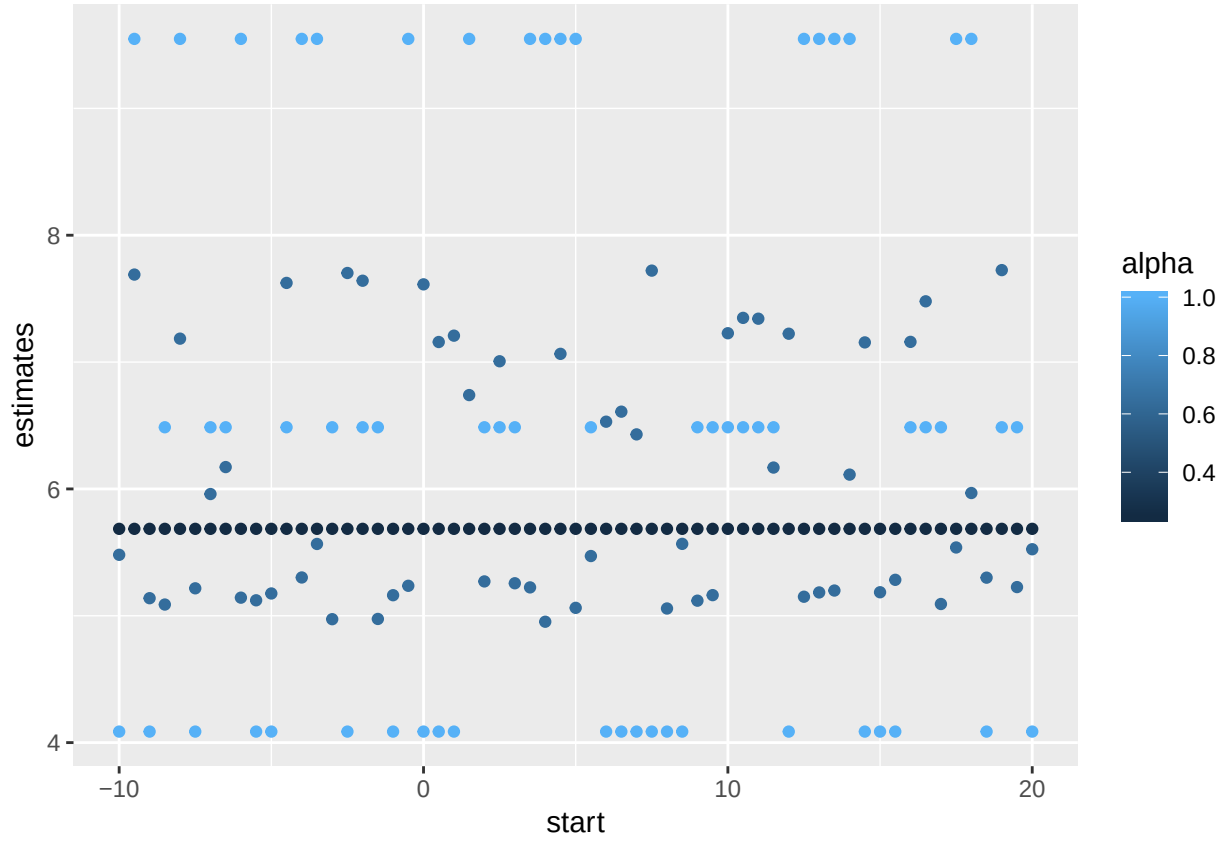
alpha	start	estimates	iter.fpi
1.00	-10.0	4.087057	100
1.00	-9.5	9.547862	100
1.00	-9.0	4.087057	100
1.00	-8.5	6.486114	100
1.00	-8.0	9.547862	100
1.00	-7.5	4.087057	100
1.00	-7.0	6.486114	100
1.00	-6.5	6.486114	100
1.00	-6.0	9.547862	100
1.00	-5.5	4.087057	100
1.00	-5.0	4.087057	100
1.00	-4.5	6.486114	100
1.00	-4.0	9.547862	100
1.00	-3.5	9.547862	100
1.00	-3.0	6.486114	100
1.00	-2.5	4.087057	100
1.00	-2.0	6.486114	100
1.00	-1.5	6.486114	100
1.00	-1.0	4.087057	100
1.00	-0.5	9.547862	100
1.00	0.0	4.087057	100
1.00	0.5	4.087057	100
1.00	1.0	4.087057	100
1.00	1.5	9.547862	100
1.00	2.0	6.486114	100
1.00	2.5	6.486114	100
1.00	3.0	6.486114	100
1.00	3.5	9.547862	100
1.00	4.0	9.547862	100
1.00	4.5	9.547862	100
1.00	5.0	9.547862	100
1.00	5.5	6.486114	100
1.00	6.0	4.087057	100
1.00	6.5	4.087057	100
1.00	7.0	4.087057	100
1.00	7.5	4.087057	100
1.00	8.0	4.087057	100
1.00	8.5	4.087057	100
1.00	9.0	6.486114	100
1.00	9.5	6.486114	100
1.00	10.0	6.486114	100
1.00	10.5	6.486114	100
1.00	11.0	6.486114	100
1.00	11.5	6.486114	100
1.00	12.0	4.087057	100
1.00	12.5	9.547862	100

alpha	start	estimates	iter.fpi
1.00	13.0	9.547862	100
1.00	13.5	9.547862	100
1.00	14.0	9.547862	100
1.00	14.5	4.087057	100
1.00	15.0	4.087057	100
1.00	15.5	4.087057	100
1.00	16.0	6.486114	100
1.00	16.5	6.486114	100
1.00	17.0	6.486114	100
1.00	17.5	9.547862	100
1.00	18.0	9.547862	100
1.00	18.5	4.087057	100
1.00	19.0	6.486114	100
1.00	19.5	6.486114	100
1.00	20.0	4.087057	100
0.64	-10.0	5.480685	100
0.64	-9.5	7.689808	100
0.64	-9.0	5.138555	100
0.64	-8.5	5.088589	100
0.64	-8.0	7.184783	100
0.64	-7.5	5.216443	100
0.64	-7.0	5.959799	100
0.64	-6.5	6.172220	100
0.64	-6.0	5.142718	100
0.64	-5.5	5.121909	100
0.64	-5.0	5.175734	100
0.64	-4.5	7.624016	100
0.64	-4.0	5.301880	100
0.64	-3.5	5.566243	100
0.64	-3.0	4.973132	100
0.64	-2.5	7.702228	100
0.64	-2.0	7.641402	100
0.64	-1.5	4.974966	100
0.64	-1.0	5.162085	100
0.64	-0.5	5.236354	100
0.64	0.0	7.612730	100
0.64	0.5	7.158276	100
0.64	1.0	7.208280	100
0.64	1.5	6.739926	100
0.64	2.0	5.270786	100
0.64	2.5	7.006485	100
0.64	3.0	5.256228	100
0.64	3.5	5.223783	100
0.64	4.0	4.953024	100
0.64	4.5	7.064915	100
0.64	5.0	5.062436	100

alpha	start	estimates	iter.fpi
0.64	5.5	5.470996	100
0.64	6.0	6.529974	100
0.64	6.5	6.609016	100
0.64	7.0	6.429923	100
0.64	7.5	7.720920	100
0.64	8.0	5.057480	100
0.64	8.5	5.566835	100
0.64	9.0	5.119058	100
0.64	9.5	5.162731	100
0.64	10.0	7.227417	100
0.64	10.5	7.348496	100
0.64	11.0	7.341966	100
0.64	11.5	6.168473	100
0.64	12.0	7.223644	100
0.64	12.5	5.150102	100
0.64	13.0	5.183969	100
0.64	13.5	5.198920	100
0.64	14.0	6.112803	100
0.64	14.5	7.155070	100
0.64	15.0	5.185394	100
0.64	15.5	5.283418	100
0.64	16.0	7.159304	100
0.64	16.5	7.479077	100
0.64	17.0	5.092643	100
0.64	17.5	5.538366	100
0.64	18.0	5.967631	100
0.64	18.5	5.300515	100
0.64	19.0	7.725016	100
0.64	19.5	5.226117	100
0.64	20.0	5.524462	100
0.25	-10.0	5.685422	47
0.25	-9.5	5.685422	45
0.25	-9.0	5.685422	44
0.25	-8.5	5.685422	42
0.25	-8.0	5.685422	41
0.25	-7.5	5.685422	39
0.25	-7.0	5.685422	38
0.25	-6.5	5.685422	36
0.25	-6.0	5.685422	34
0.25	-5.5	5.685422	34
0.25	-5.0	5.685422	30
0.25	-4.5	5.685422	31
0.25	-4.0	5.685422	29
0.25	-3.5	5.685422	28
0.25	-3.0	5.685422	27
0.25	-2.5	5.685422	26

alpha	start	estimates	iter.fpi
0.25	-2.0	5.685422	25
0.25	-1.5	5.685422	25
0.25	-1.0	5.685422	24
0.25	-0.5	5.685422	23
0.25	0.0	5.685422	23
0.25	0.5	5.685422	21
0.25	1.0	5.685422	20
0.25	1.5	5.685422	21
0.25	2.0	5.685422	20
0.25	2.5	5.685422	20
0.25	3.0	5.685422	18
0.25	3.5	5.685422	18
0.25	4.0	5.685422	19
0.25	4.5	5.685422	18
0.25	5.0	5.685422	17
0.25	5.5	5.685422	18
0.25	6.0	5.685422	17
0.25	6.5	5.685422	18
0.25	7.0	5.685422	19
0.25	7.5	5.685422	19
0.25	8.0	5.685422	20
0.25	8.5	5.685422	21
0.25	9.0	5.685422	19
0.25	9.5	5.685422	22
0.25	10.0	5.685422	23
0.25	10.5	5.685422	22
0.25	11.0	5.685422	24
0.25	11.5	5.685422	20
0.25	12.0	5.685422	25
0.25	12.5	5.685422	25
0.25	13.0	5.685422	25
0.25	13.5	5.685422	27
0.25	14.0	5.685422	28
0.25	14.5	5.685422	29
0.25	15.0	5.685422	30
0.25	15.5	5.685422	31
0.25	16.0	5.685422	32
0.25	16.5	5.685422	33
0.25	17.0	5.685422	35
0.25	17.5	5.685422	36
0.25	18.0	5.685422	36
0.25	18.5	5.685422	40
0.25	19.0	5.685422	43
0.25	19.5	5.685422	46
0.25	20.0	5.685422	50

```r
## data.table(cn = names(tble), transpose(tble))
ggplot(tble, aes(x = start, y = estimates, col = alpha)) + geom_point()
```



Fisher scoring & Newton

```r
## replace l''(theta) with -I(theta)
I <- function(x){
  -length(data.cauchy)/2
}
mle.fs <- double(length(start))
mle.nt <- double(length(start))
itr.fs <- double(length(start))
itr.nt <- double(length(start))
for (i in start){
  mle.fs[which(start == i)] = newtonRaphson(fun = llk.1st, x0 = i, dfun = I, maxiter = 100)$ro
  itr.fs[which(start == i)] = newtonRaphson(fun = llk.1st, x0 = i, dfun = I, maxiter = 100)$ni
  mle.nt[which(start == i)] = newtonRaphson(fun = llk.1st, x0 = mle.fs[which(start == i)], dfu
  itr.nt[which(start == i)] = newtonRaphson(fun = llk.1st, x0 = mle.fs[which(start == i)], dfu

}
mle.nt
```

```
##  [1] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
##  [8] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [15] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [22] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [29] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [36] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [43] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [50] 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422 5.685422
## [57] 5.685422 5.685422 5.685422 5.685422 5.685422
```

```
dt <- data.table(start.value = start, mle.fs = mle.fs, itr.fs = itr.fs, mle.nt = mle.nt, itr.nt
kable(dt)
```

start.value	mle.fs	itr.fs	mle.nt	itr.nt
-10.0	5.685422	45	5.685422	1
-9.5	5.685422	42	5.685422	1
-9.0	5.685422	40	5.685422	1
-8.5	5.685422	38	5.685422	1
-8.0	5.685422	36	5.685422	1
-7.5	5.685422	35	5.685422	1
-7.0	5.685422	33	5.685422	1
-6.5	5.685422	31	5.685422	1
-6.0	5.685422	29	5.685422	1
-5.5	5.685422	28	5.685422	1
-5.0	5.685422	27	5.685422	1
-4.5	5.685422	25	5.685422	1
-4.0	5.685422	23	5.685422	1
-3.5	5.685422	22	5.685422	1
-3.0	5.685422	21	5.685422	1
-2.5	5.685422	19	5.685422	1
-2.0	5.685422	19	5.685422	1
-1.5	5.685422	17	5.685422	1
-1.0	5.685422	16	5.685422	1
-0.5	5.685422	15	5.685422	1
0.0	5.685422	15	5.685422	1
0.5	5.685422	13	5.685422	1
1.0	5.685422	12	5.685422	1
1.5	5.685422	12	5.685422	1
2.0	5.685422	12	5.685422	1
2.5	5.685422	10	5.685422	1
3.0	5.685422	10	5.685422	1
3.5	5.685422	9	5.685422	1
4.0	5.685422	10	5.685422	1
4.5	5.685422	9	5.685422	1
5.0	5.685422	8	5.685422	1
5.5	5.685422	8	5.685422	1
6.0	5.685422	8	5.685422	1

start.value	mle.fs	itr.fs	mle.nt	itr.nt
6.5	5.685422	9	5.685422	1
7.0	5.685422	10	5.685422	1
7.5	5.685422	11	5.685422	1
8.0	5.685422	11	5.685422	1
8.5	5.685422	12	5.685422	1
9.0	5.685422	13	5.685422	1
9.5	5.685422	14	5.685422	1
10.0	5.685422	15	5.685422	1
10.5	5.685422	16	5.685422	1
11.0	5.685422	15	5.685422	1
11.5	5.685422	17	5.685422	1
12.0	5.685422	18	5.685422	1
12.5	5.685422	19	5.685422	1
13.0	5.685422	19	5.685422	1
13.5	5.685422	20	5.685422	1
14.0	5.685422	21	5.685422	1
14.5	5.685422	23	5.685422	1
15.0	5.685422	24	5.685422	1
15.5	5.685422	25	5.685422	1
16.0	5.685422	27	5.685422	1
16.5	5.685422	28	5.685422	1
17.0	5.685422	30	5.685422	1
17.5	5.685422	32	5.685422	1
18.0	5.685422	34	5.685422	1
18.5	5.685422	37	5.685422	1
19.0	5.685422	40	5.685422	1
19.5	5.685422	44	5.685422	1
20.0	5.685422	49	5.685422	1

Comparison between methods

We could see from the loglikelihood function that there is a maximum value. However, the performance of Newton–Raphson method depend on the initial values chosen, and many cases the result could not converge. For Fixed Point Iteration method, the choice of α greatly impact on whether the result would converge. For Fisher Scoring refined by Newton–Raphson method, the results converge for all initial values.
Comparison of the iteration times of the three methods are listed in the above sections. The Fisher Scoring refined by Newton–Raphson method is most stable one and the iteration time is also the smallest.

```
range(iter.nr) ## Newton-Raphson method
```

```
## [1]   4 101
```

```r
range(tble$iter.fpi) ## Fixed Point Iteration
```

```
## [1]  17 100
```

```r
range(itr.fs) ## Fisher Scoring
```

```
## [1]  8 49
```

```r
range(itr.nt) ## Newton-Raphson after Fisher Scoring
```

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

Reference

https://stackoverflow.com/questions/28253327/error-with-curve-expr-did-not-evaluate-to-an-object-of-length-n
[jun-yan/stat-5361]https://github.com/jun-yan/stat-5361