HW3

JooChul Lee

21 September 2018

Contents

1 Howework 3

Consider estimating the location parameter of a Cauchy distribution with a known scale parameter. The density function is

$$f(x;\theta) = \frac{1}{\pi[1+(x-\theta)^2]}, \quad x \in R, \quad \theta \in R.$$

Let $X_1, \ldots, X_n$ be a random sample of size n and $\ell(\theta)$ the log-likelihood function of θ based on the sample.

1.1 Show that $\ell(\theta), \ell'(\theta), \ell''(\theta), I_n(\theta)$

Likelhood function :

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\pi[1+(x_i-\theta)^2]}$$

Then, loglikelhood funcion, $\ell(\theta)$:

$$\ell(\theta) = log\prod_{i=1}^{n} \frac{1}{\pi[1+(x_i-\theta)^2]} = log\pi^{-n} + \sum_{i=1}^{n} log\frac{1}{\pi[1+(x_i-\theta)^2]} = -nlog\pi - \sum_{i=1}^{n} log[1+(\theta-x_i)^2]$$

$\ell'(\theta)$:

$$\ell'(\theta) = \frac{d}{d\theta}\ell(\theta) = -2\sum_{i=1}^{n}\frac{\theta-x_i}{1+(\theta-x_i)^2}$$

$\ell''(\theta)$:

$$\ell'(\theta) = \frac{d}{d\theta}\ell(\theta) = -2\sum_{i=1}^{n}(\frac{1}{1+(\theta-x_i)^2} - \frac{\theta-x_i}{1+(\theta-x_i)^2}) = -2\sum_{i=1}^{n}\frac{1-(\theta-x_i)^2}{(1+(\theta-x_i)^2)^2}$$

$I_n(\theta)$:

$$-E(\ell''(\theta)) = n\int_R \frac{1-(x-\theta)^2}{(1+(x-\theta)^2)^2}\frac{1}{\pi(1+(x-\theta)^2)}d$$

Since $\frac{d(\frac{x}{1+x^2})}{dx} = \frac{1-(x-\theta)^2}{(1+(x-\theta)^2)^2}$,

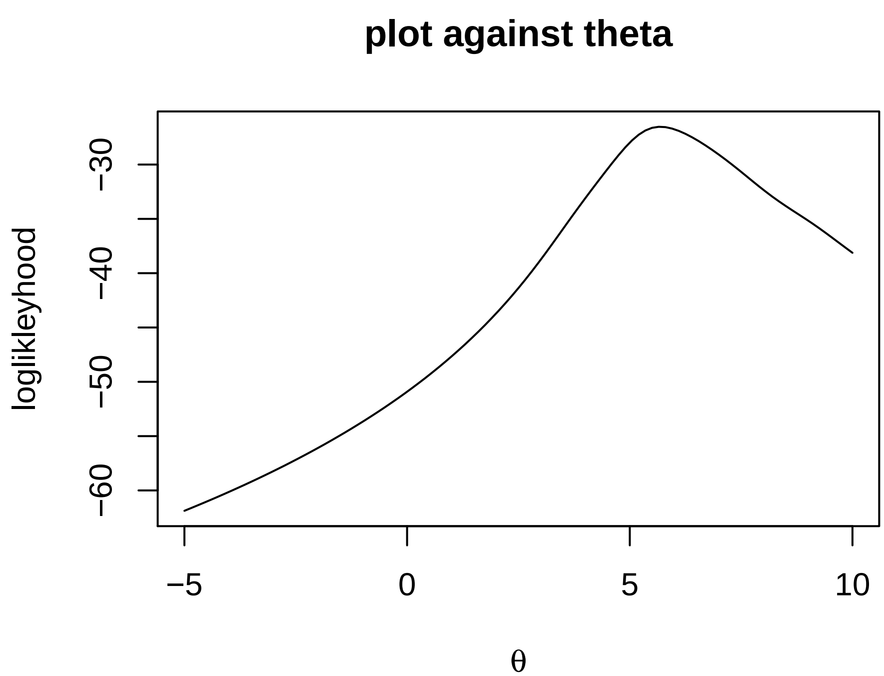
$$I_n(\theta) = \frac{2n}{\pi}\int_R \frac{d(\frac{x}{1+x^2})}{dx}\frac{1}{1+x^2}dx$$

Then by integration by Parts,

$$I_n(\theta) = \frac{4n}{\pi}\int_R \frac{x^2}{(1+x^2)^3}dx = \frac{4n}{\pi}\int_{\frac{\pi}{2}}^{\frac{\pi}{2}} sin^2\theta cos^2\theta d\theta = \frac{4n}{\pi}\int_{\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{sin^2 2\theta}{4}d\theta = \frac{n}{\pi}\frac{\theta}{2} - \frac{n}{\pi}\frac{sin4\theta}{8}|_{\frac{\pi}{2}}^{\frac{\pi}{2}} = \frac{n}{\pi}\frac{\pi}{2} = \frac{n}{2}$$

1.2 Implement a loglikelihood function and plot against θ

```
set.seed(20180909)
n = 10
sample = rcauchy(n, location = 5, scale = 1)

log_lik = function(theta)
{
  result = -length(sample) * log(pi) - sum( log( 1 + (theta - sample)^2 ) )
  return(result)
}
curve(sapply(x, FUN = log_lik), -5, 10, xlab = expression(theta),ylab = 'loglikleyhood')
title(main="plot against theta")
```

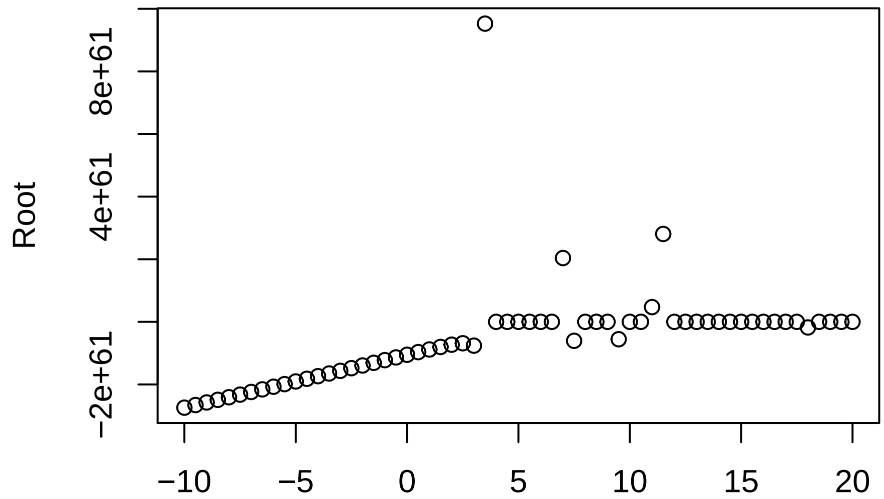


It set the random seed as 20180909 and generate a random sample of size $n = 10$ with $\theta = 5$. Form the plot, MLE is around 5.6

1.3 Find the MLE of θ using the Newton-Raphson method

When you look at the plot and table, 6 cases for intial vaules is converged. For the other cases, it is not converged even though the maximum iteration inceases. The converged values are around 5.68. Also, we can check that the root is converged when the initial values is around 5.

```
set.seed(20180909)
n = 10
sample = rcauchy(n, location = 5, scale = 1)
F_S_D = function(theta)
{
   First = -2 * sum( (theta-sample)/(1+(theta-sample)^2) )
   Second = -2 * sum( (1 - (theta-sample)^2)/(1+(theta-sample)^2)^2 )
   list(First = First, Second= Second )
}

N_R = function(initial, max = 100, tol = 1e-5)
{
   current = initial
   for(i in 1:max)
   {
      new = current - F_S_D(current)$First/F_S_D(current)$Second
      if(abs(new -current) < tol) break
      current1 = current
      current = new
   }
   return( c(current, i, diff = abs(current -current1) ) )
}
initial = seq(-10, 20, by = 0.5)
result = matrix(0, length(initial), 3)
for(i in 1:length(initial))
{
   result[i,] = N_R(initial[i],200)
}
colnames(result) = c('Root', '# iter.', 'Difference b/w new and current')
rownames(result) = paste('Initial =',seq(-10, 20, by = 0.5))

plot(seq(-10, 20, by = 0.5), result[,1], ylab = 'Root', xlab = '')
```




```
knitr::kable(result[1:30,], booktabs = TRUE,align='c',
             row.names = 1)
```

	Root	# iter.	Difference b/w new and current
Initial = -10	-2.741130e+61	200	1.370565e+61
Initial = -9.5	-2.657995e+61	200	1.328997e+61
Initial = -9	-2.574756e+61	200	1.287378e+61
Initial = -8.5	-2.491407e+61	200	1.245704e+61
Initial = -8	-2.407943e+61	200	1.203971e+61
Initial = -7.5	-2.324355e+61	200	1.162177e+61
Initial = -7	-2.240636e+61	200	1.120318e+61
Initial = -6.5	-2.156778e+61	200	1.078389e+61
Initial = -6	-2.072774e+61	200	1.036387e+61
Initial = -5.5	-1.988614e+61	200	9.943068e+60
Initial = -5	-1.904288e+61	200	9.521439e+60
Initial = -4.5	-1.819787e+61	200	9.098937e+60
Initial = -4	-1.735103e+61	200	8.675515e+60
Initial = -3.5	-1.650227e+61	200	8.251133e+60
Initial = -3	-1.565152e+61	200	7.825758e+60
Initial = -2.5	-1.479876e+61	200	7.399378e+60
Initial = -2	-1.394404e+61	200	6.972018e+60
Initial = -1.5	-1.308754e+61	200	6.543770e+60
Initial = -1	-1.222971e+61	200	6.114854e+60
Initial = -0.5	-1.137146e+61	200	5.685729e+60
Initial = 0	-1.051467e+61	200	5.257333e+60
Initial = 0.5	-9.663188e+60	200	4.831594e+60
Initial = 1	-8.825356e+60	200	4.412678e+60
Initial = 1.5	-8.020672e+60	200	4.010336e+60
Initial = 2	-7.301183e+60	200	3.650591e+60
Initial = 2.5	-6.841241e+60	200	3.420621e+60
Initial = 3	-7.604318e+60	200	3.802159e+60
Initial = 3.5	9.526123e+61	200	4.763062e+61
Initial = 4	2.056366e+01	200	5.186377e-01
Initial = 4.5	2.108229e+01	200	1.704859e+00

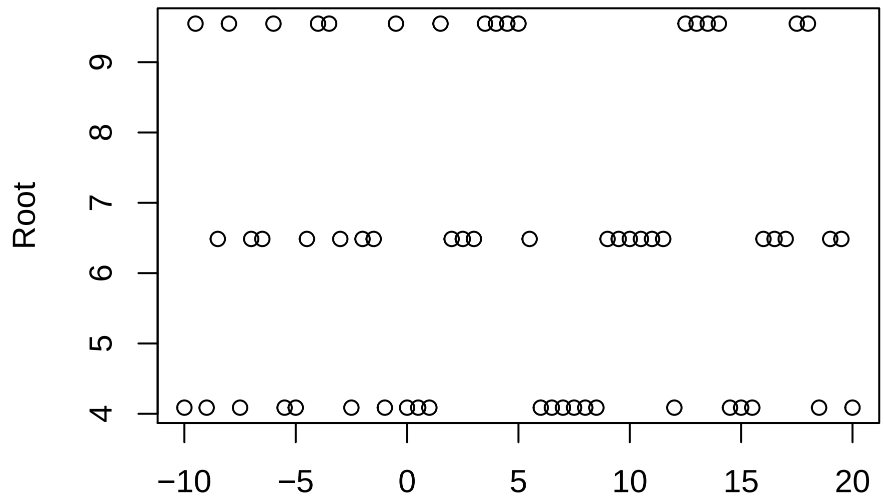
```
knitr::kable(result[31:61,], booktabs = TRUE,align='c',
             row.names = 1)
```

	Root	# iter.	Difference b/w new and current
Initial = 5	5.685418e+00	5	2.380900e-03
Initial = 5.5	5.685422e+00	4	1.141000e-04
Initial = 6	5.685422e+00	5	2.280000e-05
Initial = 6.5	5.685421e+00	7	6.893000e-04
Initial = 7	2.038366e+61	200	1.019183e+61
Initial = 7.5	-6.058665e+60	200	3.029333e+60
Initial = 8	1.937744e+01	200	1.186222e+00
Initial = 8.5	2.108229e+01	200	1.704859e+00
Initial = 9	5.685422e+00	7	9.790000e-05
Initial = 9.5	-5.551985e+60	200	2.775993e+60
Initial = 10	2.108229e+01	200	1.704859e+00
Initial = 10.5	5.685422e+00	6	1.050000e-05
Initial = 11	4.715381e+60	200	2.357691e+60
Initial = 11.5	2.807261e+61	200	1.403631e+61
Initial = 12	2.056366e+01	200	5.186377e-01
Initial = 12.5	2.056366e+01	200	5.186377e-01
Initial = 13	2.056366e+01	200	5.186377e-01
Initial = 13.5	2.108229e+01	200	1.704859e+00
Initial = 14	2.108229e+01	200	1.704859e+00
Initial = 14.5	2.108229e+01	200	1.704859e+00
Initial = 15	1.937744e+01	200	1.186222e+00
Initial = 15.5	2.056366e+01	200	5.186377e-01
Initial = 16	2.056366e+01	200	5.186377e-01
Initial = 16.5	1.937744e+01	200	1.186222e+00
Initial = 17	1.937744e+01	200	1.186222e+00
Initial = 17.5	1.937744e+01	200	1.186222e+00
Initial = 18	-1.790860e+60	200	8.954302e+59
Initial = 18.5	2.056366e+01	200	5.186377e-01
Initial = 19	2.056366e+01	200	5.186377e-01
Initial = 19.5	2.056366e+01	200	5.186377e-01
Initial = 20	2.056366e+01	200	5.186377e-01

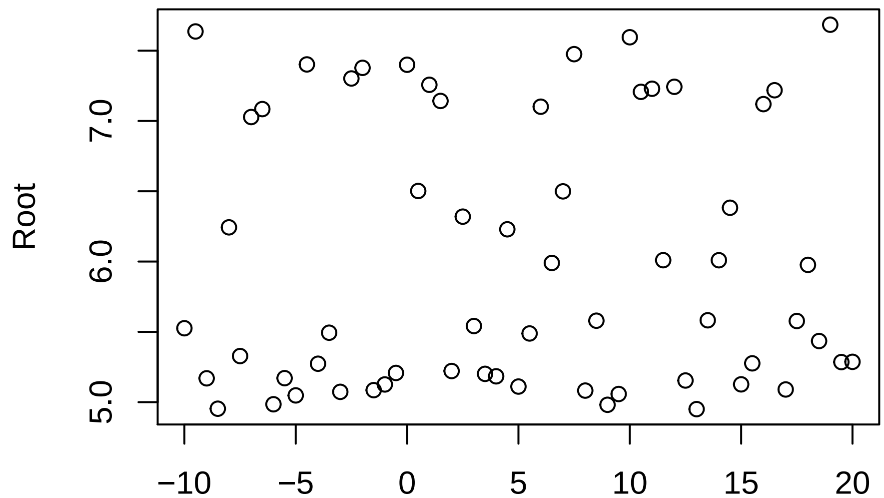
1.4 Apply fixed-point iterations using $G(\theta) = \alpha\ell'(\theta) + \theta$

When you look at the plot and table, it is not converged in the case of $\alpha = 1, 0.64$ even though the maximum iteration inceases for all initial values. But when $\alpha = 0.25$, values are converged at around 5.68 regardless intital values.

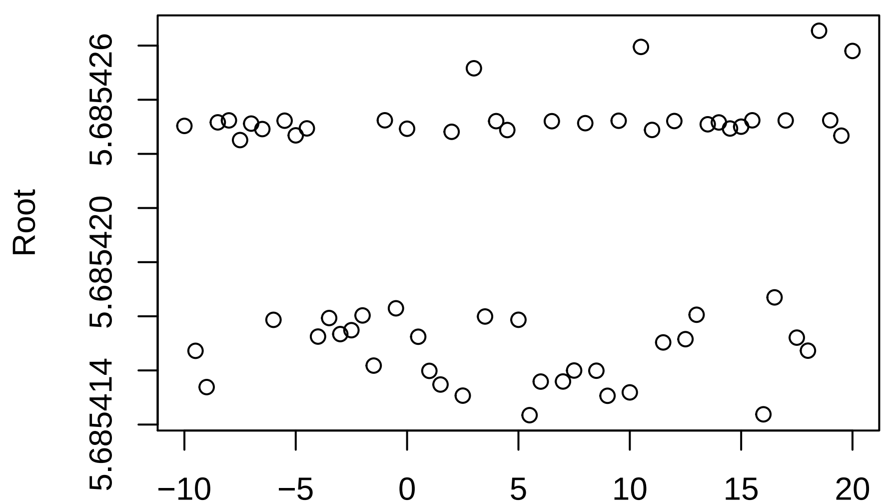
```
set.seed(20180909)
n = 10
sample = rcauchy(n, location = 5, scale = 1)
F_D = function(theta)
{
   First = -2 * sum( (theta-sample)/(1+(theta-sample)^2) )
   return(First)
}

Fixed = function(initial, alpha, max = 100, tol = 1e-5)
{
   current = initial
   for(i in 1:max)
   {
      new = current + (alpha*F_D(current))
      if(abs(new -current) < tol) break
      current1 = current
      current = new
   }
   return( c(current, i, diff = abs(current -current1) ) )
}
initial = seq(-10, 20, by = 0.5)
result_1 = matrix(0, length(initial), 3)
result_0.64 = matrix(0, length(initial), 3)
result_0.25 = matrix(0, length(initial), 3)
for(i in 1:length(initial))
{
   result_1[i,] = Fixed(initial[i], 1 ,10000)
   result_0.64[i,] = Fixed(initial[i],0.64,10000)
   result_0.25[i,] = Fixed(initial[i],0.25, 10000)
}
colnames(result_1)<- colnames(result_0.64)<- colnames(result_0.25)<-
    c('Root', '# iter.', 'Difference b/w new and current')
rownames(result_1) <- rownames(result_0.64) <- rownames(result_0.25) <-
    paste('Initial =',seq(-10, 20, by = 0.5))

plot(seq(-10, 20, by = 0.5), result_1[,1], ylab = 'Root', xlab = '')
```




```r
plot(seq(-10, 20, by = 0.5), result_0.64[,1], ylab = 'Root', xlab = '')
```




```r
plot(seq(-10, 20, by = 0.5), result_0.25[,1], ylab = 'Root', xlab = '')
```




```r
knitr::kable(result_1[1:30,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =1')
```

```r
knitr::kable(result_1[31:61,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =1')
```

```r
knitr::kable(result_0.64[1:30,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =0.64')
```

```r
knitr::kable(result_0.64[31:61,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =0.64')
```

```r
knitr::kable(result_0.25[1:30,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =0.25')
```

```r
knitr::kable(result_0.25[31:61,], booktabs = TRUE,align='c',
             row.names = 1,caption = 'Table for alpha =0.25')
```

Table 1: Table for alpha =1

	Root	# iter.	Difference b/w new and current
Initial = -10	4.087057	10000	2.399057
Initial = -9.5	9.547862	10000	5.460805
Initial = -9	4.087057	10000	2.399057
Initial = -8.5	6.486114	10000	3.061748
Initial = -8	9.547862	10000	5.460805
Initial = -7.5	4.087057	10000	2.399057
Initial = -7	6.486114	10000	3.061748
Initial = -6.5	6.486114	10000	3.061748
Initial = -6	9.547862	10000	5.460805
Initial = -5.5	4.087057	10000	2.399057
Initial = -5	4.087057	10000	2.399057
Initial = -4.5	6.486114	10000	3.061748
Initial = -4	9.547862	10000	5.460805
Initial = -3.5	9.547862	10000	5.460805
Initial = -3	6.486114	10000	3.061748
Initial = -2.5	4.087057	10000	2.399057
Initial = -2	6.486114	10000	3.061748
Initial = -1.5	6.486114	10000	3.061748
Initial = -1	4.087057	10000	2.399057
Initial = -0.5	9.547862	10000	5.460805
Initial = 0	4.087057	10000	2.399057
Initial = 0.5	4.087057	10000	2.399057
Initial = 1	4.087057	10000	2.399057
Initial = 1.5	9.547862	10000	5.460805
Initial = 2	6.486114	10000	3.061748
Initial = 2.5	6.486114	10000	3.061748
Initial = 3	6.486114	10000	3.061748
Initial = 3.5	9.547862	10000	5.460805
Initial = 4	9.547862	10000	5.460805
Initial = 4.5	9.547862	10000	5.460805

Table 2: Table for alpha =1

	Root	# iter.	Difference b/w new and current
Initial = 5	9.547862	10000	5.460805
Initial = 5.5	6.486114	10000	3.061748
Initial = 6	4.087057	10000	2.399057
Initial = 6.5	4.087057	10000	2.399057
Initial = 7	4.087057	10000	2.399057
Initial = 7.5	4.087057	10000	2.399057
Initial = 8	4.087057	10000	2.399057
Initial = 8.5	4.087057	10000	2.399057
Initial = 9	6.486114	10000	3.061748
Initial = 9.5	6.486114	10000	3.061748
Initial = 10	6.486114	10000	3.061748
Initial = 10.5	6.486114	10000	3.061748
Initial = 11	6.486114	10000	3.061748
Initial = 11.5	6.486114	10000	3.061748
Initial = 12	4.087057	10000	2.399057
Initial = 12.5	9.547862	10000	5.460805
Initial = 13	9.547862	10000	5.460805
Initial = 13.5	9.547862	10000	5.460805
Initial = 14	9.547862	10000	5.460805
Initial = 14.5	4.087057	10000	2.399057
Initial = 15	4.087057	10000	2.399057
Initial = 15.5	4.087057	10000	2.399057
Initial = 16	6.486114	10000	3.061748
Initial = 16.5	6.486114	10000	3.061748
Initial = 17	6.486114	10000	3.061748
Initial = 17.5	9.547862	10000	5.460805
Initial = 18	9.547862	10000	5.460805
Initial = 18.5	4.087057	10000	2.399057
Initial = 19	6.486114	10000	3.061748
Initial = 19.5	6.486114	10000	3.061748
Initial = 20	4.087057	10000	2.399057

Table 3: Table for alpha =0.64

	Root	# iter.	Difference b/w new and current
Initial = -10	5.525759	10000	2.1501473
Initial = -9.5	7.636703	10000	2.6313086
Initial = -9	5.169572	10000	1.9815503
Initial = -8.5	4.953209	10000	1.5762712
Initial = -8	6.243344	10000	0.7527119
Initial = -7.5	5.327966	10000	2.1146908
Initial = -7	7.027978	10000	1.7809897
Initial = -6.5	7.084503	10000	1.8561853
Initial = -6	4.985057	10000	1.2546729
Initial = -5.5	5.170596	10000	0.8014345
Initial = -5	5.047966	10000	1.8343950
Initial = -4.5	7.402185	10000	2.2890758
Initial = -4	5.273161	10000	2.0805175
Initial = -3.5	5.493783	10000	2.1508381
Initial = -3	5.073243	10000	1.0017347
Initial = -2.5	7.302259	10000	2.1504548
Initial = -2	7.377826	10000	2.2550049
Initial = -1.5	5.085442	10000	0.9740904
Initial = -1	5.125818	10000	1.9320688
Initial = -0.5	5.207631	10000	2.0219897
Initial = 0	7.399890	10000	2.2858575
Initial = 0.5	6.501876	10000	1.0912332
Initial = 1	7.257221	10000	2.0888442
Initial = 1.5	7.142420	10000	1.9336602
Initial = 2	5.221411	10000	2.0356843
Initial = 2.5	6.319499	10000	0.8527238
Initial = 3	5.541442	10000	2.1492014
Initial = 3.5	5.201300	10000	2.0155025
Initial = 4	5.183698	10000	1.9969271
Initial = 4.5	6.229508	10000	0.7345032

Table 4: Table for alpha =0.64

	Root	# iter.	Difference b/w new and current
Initial = 5	5.110902	10000	1.9146160
Initial = 5.5	5.488538	10000	2.1507780
Initial = 6	7.101392	10000	1.8787289
Initial = 6.5	5.989930	10000	0.4165361
Initial = 7	6.499335	10000	1.0879170
Initial = 7.5	7.475178	10000	2.3924783
Initial = 8	5.082166	10000	1.8797523
Initial = 8.5	5.579203	10000	2.1455001
Initial = 9	4.981127	10000	1.7078425
Initial = 9.5	5.057975	10000	1.8482999
Initial = 10	7.595380	10000	2.5684871
Initial = 10.5	7.207182	10000	2.0209055
Initial = 11	7.229123	10000	2.0506339
Initial = 11.5	6.009757	10000	0.4430885
Initial = 12	7.243113	10000	2.0696384
Initial = 12.5	5.153909	10000	1.9641251
Initial = 13	4.950411	10000	1.4994315
Initial = 13.5	5.582250	10000	2.1451212
Initial = 14	6.009468	10000	0.4427021
Initial = 14.5	6.382688	10000	0.9354866
Initial = 15	5.126622	10000	1.9330000
Initial = 15.5	5.276002	10000	2.0826338
Initial = 16	7.119801	10000	1.9033460
Initial = 16.5	7.218504	10000	2.0362344
Initial = 17	5.090256	10000	0.9634434
Initial = 17.5	5.577203	10000	2.1457425
Initial = 18	5.976268	10000	0.3982092
Initial = 18.5	5.434719	10000	2.1468451
Initial = 19	7.684688	10000	2.7063690
Initial = 19.5	5.285052	10000	2.0891208
Initial = 20	5.286778	10000	2.0903144

Table 5: Table for alpha =0.25

	Root	# iter.	Difference b/w new and current
Initial = -10	5.685425	41	1.28e-05
Initial = -9.5	5.685417	39	1.87e-05
Initial = -9	5.685415	37	2.37e-05
Initial = -8.5	5.685425	36	1.33e-05
Initial = -8	5.685425	35	1.36e-05
Initial = -7.5	5.685424	33	1.08e-05
Initial = -7	5.685425	32	1.31e-05
Initial = -6.5	5.685425	30	1.24e-05
Initial = -6	5.685418	28	1.43e-05
Initial = -5.5	5.685425	28	1.35e-05
Initial = -5	5.685425	24	1.15e-05
Initial = -4.5	5.685425	25	1.25e-05
Initial = -4	5.685417	23	1.67e-05
Initial = -3.5	5.685418	22	1.41e-05
Initial = -3	5.685417	21	1.63e-05
Initial = -2.5	5.685417	20	1.58e-05
Initial = -2	5.685418	19	1.37e-05
Initial = -1.5	5.685416	18	2.07e-05
Initial = -1	5.685425	18	1.36e-05
Initial = -0.5	5.685418	17	1.27e-05
Initial = 0	5.685425	17	1.24e-05
Initial = 0.5	5.685417	15	1.67e-05
Initial = 1	5.685416	13	2.15e-05
Initial = 1.5	5.685416	14	2.34e-05
Initial = 2	5.685425	14	1.20e-05
Initial = 2.5	5.685415	13	2.49e-05
Initial = 3	5.685427	11	2.09e-05
Initial = 3.5	5.685418	12	1.39e-05
Initial = 4	5.685425	13	1.35e-05
Initial = 4.5	5.685425	12	1.22e-05

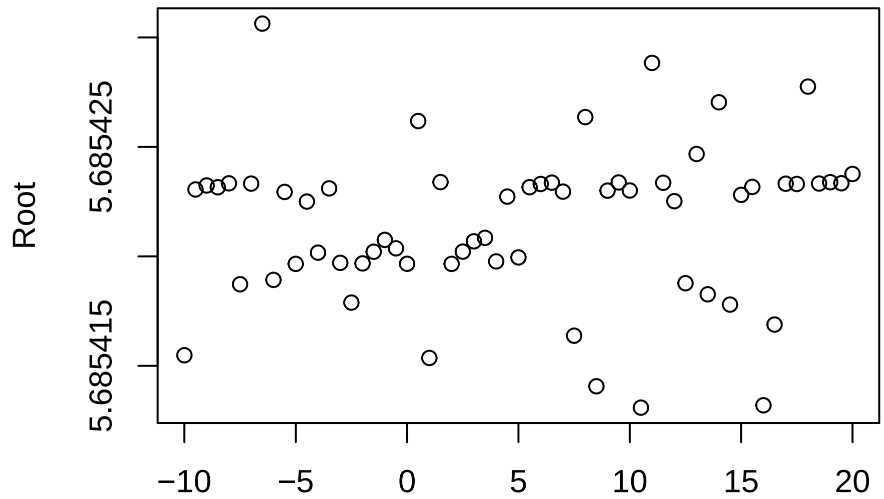
Table 6: Table for alpha =0.25

	Root	# iter.	Difference b/w new and current
Initial = 5	5.685418	11	1.43e-05
Initial = 5.5	5.685414	11	2.77e-05
Initial = 6	5.685416	10	2.30e-05
Initial = 6.5	5.685425	12	1.35e-05
Initial = 7	5.685416	12	2.29e-05
Initial = 7.5	5.685416	12	2.14e-05
Initial = 8	5.685425	14	1.32e-05
Initial = 8.5	5.685416	14	2.14e-05
Initial = 9	5.685415	12	2.49e-05
Initial = 9.5	5.685425	16	1.35e-05
Initial = 10	5.685415	16	2.45e-05
Initial = 10.5	5.685428	15	2.39e-05
Initial = 11	5.685425	18	1.22e-05
Initial = 11.5	5.685417	14	1.75e-05
Initial = 12	5.685425	19	1.35e-05
Initial = 12.5	5.685417	19	1.70e-05
Initial = 13	5.685418	19	1.36e-05
Initial = 13.5	5.685425	21	1.30e-05
Initial = 14	5.685425	22	1.33e-05
Initial = 14.5	5.685425	23	1.24e-05
Initial = 15	5.685425	24	1.27e-05
Initial = 15.5	5.685425	25	1.36e-05
Initial = 16	5.685414	25	2.75e-05
Initial = 16.5	5.685419	27	1.12e-05
Initial = 17	5.685425	29	1.36e-05
Initial = 17.5	5.685417	30	1.68e-05
Initial = 18	5.685417	30	1.86e-05
Initial = 18.5	5.685429	33	2.61e-05
Initial = 19	5.685425	37	1.36e-05
Initial = 19.5	5.685425	40	1.14e-05
Initial = 20	5.685428	43	2.33e-05

1.5 First use Fisher scoring to find the MLE for θ

When you look at the plot and table, the values are converged around 5.68 for all initial values. But the iterations for it are different with each other.

```
set.seed(20180909)
n = 10
sample = rcauchy(n, location = 5, scale = 1)
F_D = function(theta)
{
   First = -2 * sum( (theta-sample)/(1+(theta-sample)^2) )
   return(First)
}

N_R = function(initial, max = 100, tol = 1e-5)
{
   current = initial
   for(i in 1:max)
   {
      new = current + 2 * F_D(current)/n
      if(abs(new -current) < tol) break
      current1 = current
      current = new
   }
   return( c(current, i, diff = abs(current -current1) ) )
}
initial = seq(-10, 20, by = 0.5)
result = matrix(0, length(initial), 3)
for(i in 1:length(initial))
{
   result[i,] = N_R(initial[i],200)
}
colnames(result) = c('Root', '# iter.', 'Difference b/w new and current')
rownames(result) = paste('Initial =',seq(-10, 20, by = 0.5))

plot(seq(-10, 20, by = 0.5), result[,1], ylab = 'Root', xlab = '')
```




```
knitr::kable(result[1:30,], booktabs = TRUE,align='c',
             row.names = 1)
```

	Root	# iter.	Difference b/w new and current
Initial = -10	5.685416	42	7.70e-05
Initial = -9.5	5.685423	40	1.75e-05
Initial = -9	5.685423	38	1.98e-05
Initial = -8.5	5.685423	36	1.88e-05
Initial = -8	5.685423	34	2.10e-05
Initial = -7.5	5.685419	32	3.65e-05
Initial = -7	5.685423	31	2.09e-05
Initial = -6.5	5.685431	28	1.12e-04
Initial = -6	5.685419	26	3.40e-05
Initial = -5.5	5.685423	26	1.61e-05
Initial = -5	5.685420	24	2.49e-05
Initial = -4.5	5.685422	23	1.06e-05
Initial = -4	5.685420	21	1.85e-05
Initial = -3.5	5.685423	20	1.81e-05
Initial = -3	5.685420	18	2.43e-05
Initial = -2.5	5.685418	16	4.70e-05
Initial = -2	5.685420	16	2.46e-05
Initial = -1.5	5.685420	15	1.80e-05
Initial = -1	5.685421	14	1.12e-05
Initial = -0.5	5.685420	13	1.60e-05
Initial = 0	5.685420	12	2.49e-05
Initial = 0.5	5.685426	10	5.65e-05
Initial = 1	5.685415	9	7.86e-05
Initial = 1.5	5.685423	10	2.18e-05
Initial = 2	5.685420	9	2.49e-05
Initial = 2.5	5.685420	8	1.79e-05
Initial = 3	5.685421	8	1.20e-05
Initial = 3.5	5.685421	7	1.00e-05
Initial = 4	5.685420	7	2.35e-05
Initial = 4.5	5.685423	7	1.35e-05

```
knitr::kable(result[31:61,], booktabs = TRUE,align='c',
             row.names = 1)
```

	Root	# iter.	Difference b/w new and current
Initial = 5	5.685420	6	0.0000212
Initial = 5.5	5.685423	6	0.0000188
Initial = 6	5.685423	6	0.0000207
Initial = 6.5	5.685423	7	0.0000214
Initial = 7	5.685423	8	0.0000163
Initial = 7.5	5.685416	8	0.0000659
Initial = 8	5.685426	8	0.0000588
Initial = 8.5	5.685414	9	0.0000947
Initial = 9	5.685423	11	0.0000169
Initial = 9.5	5.685423	12	0.0000215
Initial = 10	5.685423	13	0.0000169
Initial = 10.5	5.685413	13	0.0001069
Initial = 11	5.685429	12	0.0000896
Initial = 11.5	5.685423	15	0.0000214
Initial = 12	5.685422	16	0.0000109
Initial = 12.5	5.685419	16	0.0000359
Initial = 13	5.685425	16	0.0000378
Initial = 13.5	5.685418	17	0.0000423
Initial = 14	5.685427	18	0.0000672
Initial = 14.5	5.685418	20	0.0000481
Initial = 15	5.685423	22	0.0000145
Initial = 15.5	5.685423	23	0.0000190
Initial = 16	5.685413	24	0.0001056
Initial = 16.5	5.685417	25	0.0000595
Initial = 17	5.685423	28	0.0000208
Initial = 17.5	5.685423	30	0.0000207
Initial = 18	5.685428	31	0.0000762
Initial = 18.5	5.685423	35	0.0000209
Initial = 19	5.685423	38	0.0000217
Initial = 19.5	5.685423	42	0.0000211
Initial = 20	5.685424	46	0.0000264

1.6 Comment on the results from different methods

In the conclusion, the roots could be converged or not according to intial values in the case of Newton-Raphson with second derivative. It means that the algorithm might be unstable. But when it is converged, the converging speed very fast compared to fixed point algorithm. Also, the roots could be converged or not according to alpha in the case of the Fixed point algorithm. It means that we can say that it is not stable for alpha even though it is converged for all initial values, but the converging speed is quite fast. For Newton-Raphson with fisher scoring, the values are converged for all initial values and also, the speed is fast compared to two other algoritm. Thus, we can conclude that Newton-Raphson with fisher scoring is more constant and faster than the others in this example.