Atlas-PS 2

David Atlas

Problem 1

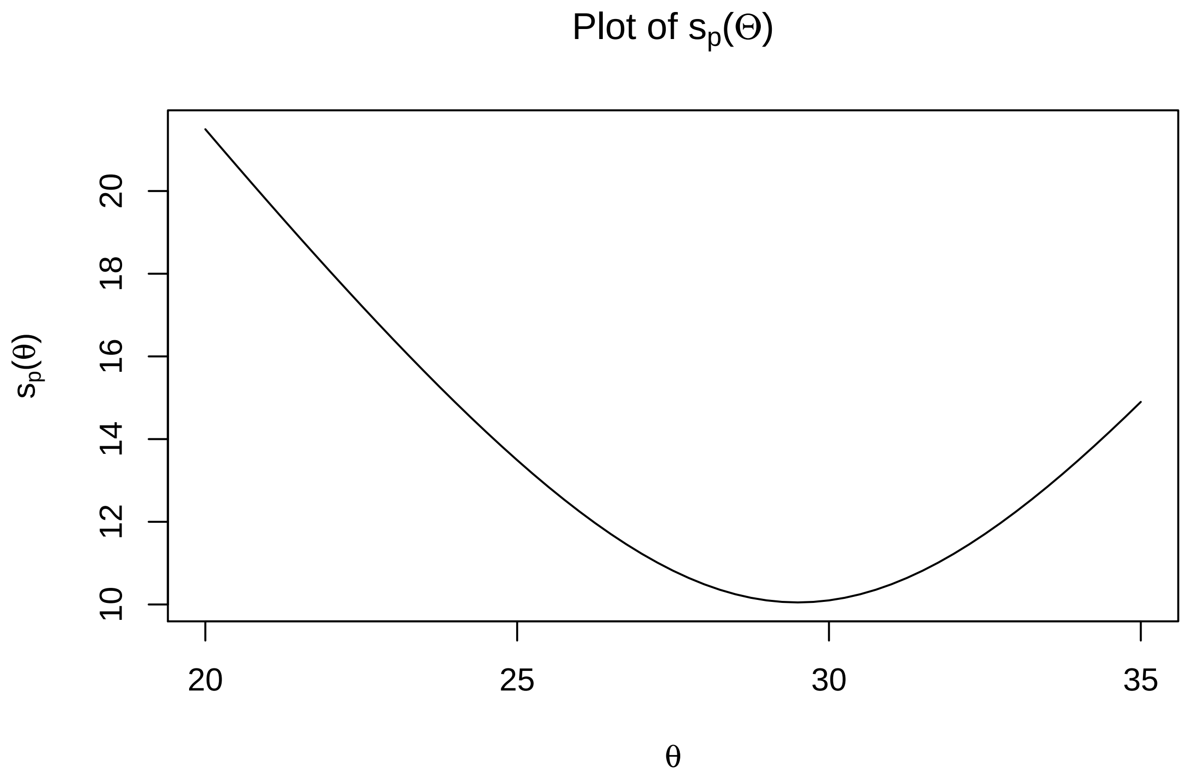
a)

Let $X = (28, 33, 22, 35)$ be our set of i.i.d data points. The function $s_p(\theta) = \sqrt{\Sigma_{x \in X}(\theta - x)^2}$, or the sum of squared residuals.

b)

$s_p(\theta)$ is plotted below, with the R code used to generate the plot.

```r
x <- c(28, 33, 22, 35)
s_p <- function(theta, x){
  # We implement our function to minimize
  return(sqrt(sum((theta - x) ^ 2)))
}

# We set up our domain for theta
theta_space <- seq(20, 35, .25)
# We calculate the function value over the space
s_p_theta_space <- sapply(theta_space, function(theta){s_p(theta, x)})

# We plot the function over the space
plot(theta_space, s_p_theta_space, 'l',
  main=TeX('Plot of $s_p(\\Theta)$'),
  xlab=TeX('$\\theta$'), ylab=TeX('$s_p(\\theta)$'))
```



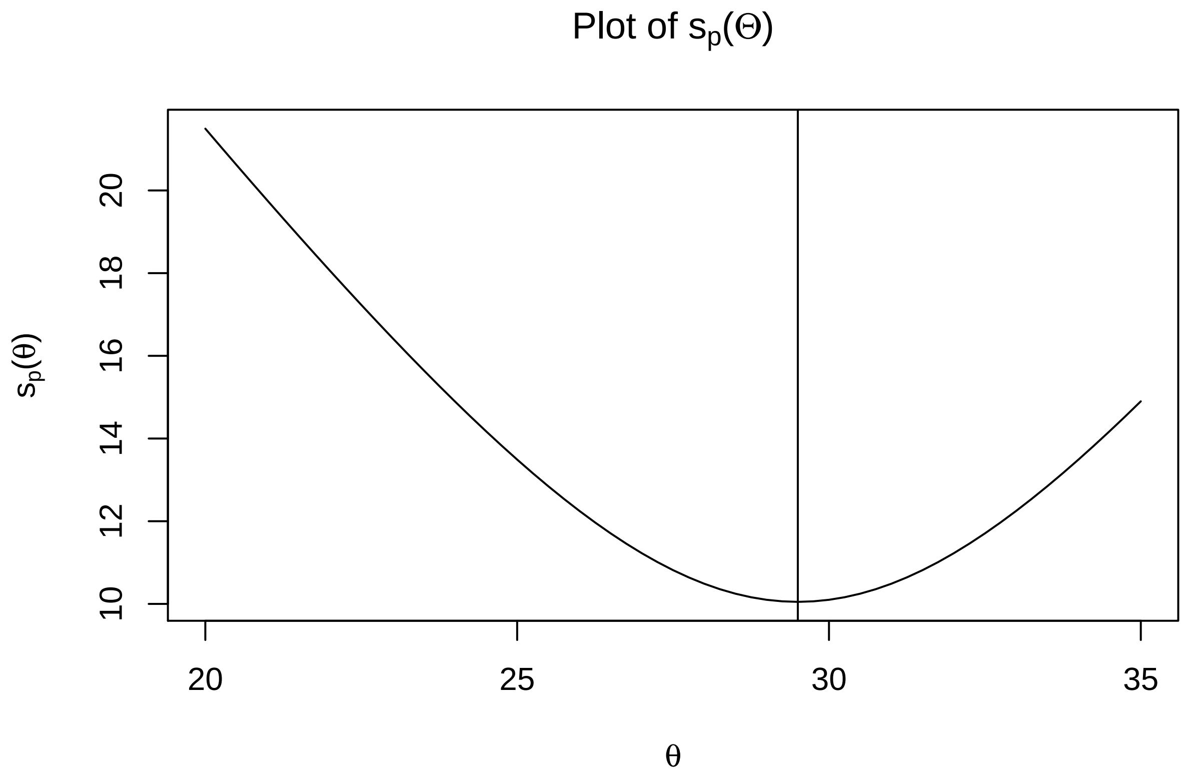
c)

To use the bisection method, we must first compute $s_p\prime(\theta)$.

$$\begin{aligned} s_p\prime(\theta) &= \frac{1}{2}(\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}} \times 2\Sigma_{x \in X}(\theta - x) \\ &= (\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}}\Sigma_{x \in X}(\theta - x). \end{aligned}$$

Next, we implement the bisection method, as well as $s_p(\theta)$ and $s_p\prime(\theta)$. We plot the $s_p(\theta)$ with the Minimum Residual Estimator as a vertical line. The solution to the optimization problem is $\hat{\theta} = 29.50$.

```r
x <- c(28, 33, 22, 35)

s_p <- function(theta, x){
  # We implement our function to minimize
  return(sqrt(sum((theta - x) ^ 2)))
}

s_p_prime <- function(theta, x){
  # This is the first derivative of the function
  return(((sum(theta - x) ^ 2) ^ -.5) * sum(theta - x))
}

bisection <- function(a, b, f_prime, tol=.0001, n=0){
  x_t <- .5 * (a + b)
  # Use conditioning to get the next interval
  if(f_prime(a, x) * f_prime(x_t, x) <= 0){
    new_interval <- c(a, x_t)
  }else{
    new_interval <- c(x_t, b)
  }

  # if interval is less than the tolerance, stop the recursion.
  if ((b - a) < tol){
    print(paste0("The solution is ", round(x_t, 3) , " and it was found in ", n, " iterations."))
    return(x_t)
  }else{
    # If not, call again on the new interval
    return(bisection(new_interval[1], new_interval[2], f_prime, n=n + 1))
  }
}

plot(theta_space, s_p_theta_space, 'l',
  main=TeX('Plot of $s_p(\\Theta)$'),
  xlab=TeX('$\\theta$'), ylab=TeX('$s_p(\\theta)$'))
abline(v=bisection(20, 35, s_p_prime, tol=.000001))
```



```
## [1] "The solution is 29.5 and it was found in 18 iterations."
```

d)

We already calculated $s'_p(\theta) = (\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}} \Sigma_{x \in X}(\theta - x)$. We find

$$s''_p(\theta) = -\frac{1}{2}(\Sigma_{x \in X}(\theta - x)^2)^{-\frac{3}{2}} \Sigma_{x \in X}(\theta - x) + (\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}}.$$

We can then find $h(\theta) = \frac{s'_p(\theta)}{s''_p(\theta)}$.

$$\begin{aligned} -\frac{s'_p(\theta)}{s''_p(\theta)} &= -\frac{(\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}} \Sigma_{x \in X}(\theta - x)}{-\frac{1}{2}(\Sigma_{x \in X}(\theta - x)^2)^{-\frac{3}{2}} \Sigma_{x \in X}(\theta - x) + (\Sigma_{x \in X}(\theta - x)^2)^{-\frac{1}{2}}} \\ &= 2\frac{\Sigma_{x \in X}(\theta - x)}{\Sigma_{x \in X}(\theta - x)^2 + 2} \end{aligned}$$

Problem 2

We maximize the function $f(x) = -\frac{x^4}{4} + \frac{x^2}{2} - x + 2$ using Newton's Method and starting points $x_0 = -1$ and $x_0 = 2$. We also print out the number of iterations needed to converge within 2 decimal places. We define the first 2 derivatives of the function below:

$$f(x) = -\frac{x^4}{4} + \frac{x^2}{2} - x + 2 \quad (1)$$
$$f'(x) = -x^3 + x - 1 \quad (2)$$
$$f''(x) = -3x^2 + 1 \quad (3)$$

We implement Newton's Method:

```
newtons <- function(xt, fprime, f2prime, n=1, tol=0.01){
  # Define the updating equation
  xt_update <- xt - (fprime(xt) / f2prime(xt))

  # If the adjustment value is less than the tolerance, end the iterations
  if(abs(xt_update - xt) < tol){
    print(paste0("The solution is ", round(xt_update, 3) , " and it was found in ", n, " iterations."))
    return(xt_update)
  }else{
    # If not, call the recursive formula again
    return(newtons(xt_update, fprime, f2prime, n=n+1, tol=tol))
  }
}

fprime <- function(x){
  return(-x^3 + x -1)
}

f2prime <- function(x){
  return(-3 * x ^ 2 + 1)
}
```

a)

We solve the optimization using $x_0 = -1$.

```
x0 <- -1
solution <- newtons(x0, fprime, f2prime)
```

```
## [1] "The solution is -1.325 and it was found in 4 iterations."
```

The solution is -1.325, and it took 4 iterations to find it.

b)

We solve the optimization using $x_0 = 2$.

```
x0 <- 2
solution <- newtons(x0, fprime, f2prime)
```

```
## [1] "The solution is -1.325 and it was found in 64 iterations."
```

The solution is -1.325, and it took 64 iterations to find it.

Problem 3

We solve exercise 2.1 from the textbook:

The following data are an i.i.d. sample from a Cauchy(θ, 1) distribution: 1.77, -.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24, -2.44, 3.29, 3.71, -2.40, 4.53, -.07, -1.05, -13.87, -2.53, -1.75, .27, 43.21.

a)

Graph the log likelihood function. Find the MLE for θ using the Newton-Raphson method. Try the following starting point: -11, -1, 0, 1.5, 4, 4.7, 7, 8, 38. Discuss your results. Is the mean of the data a good starting point?

The likelihood function of a Cauchy(θ, 1) distribution:

$$L(\theta) = \prod_{x\in X} \frac{1}{\pi(1+(x-\theta)^2)}.$$

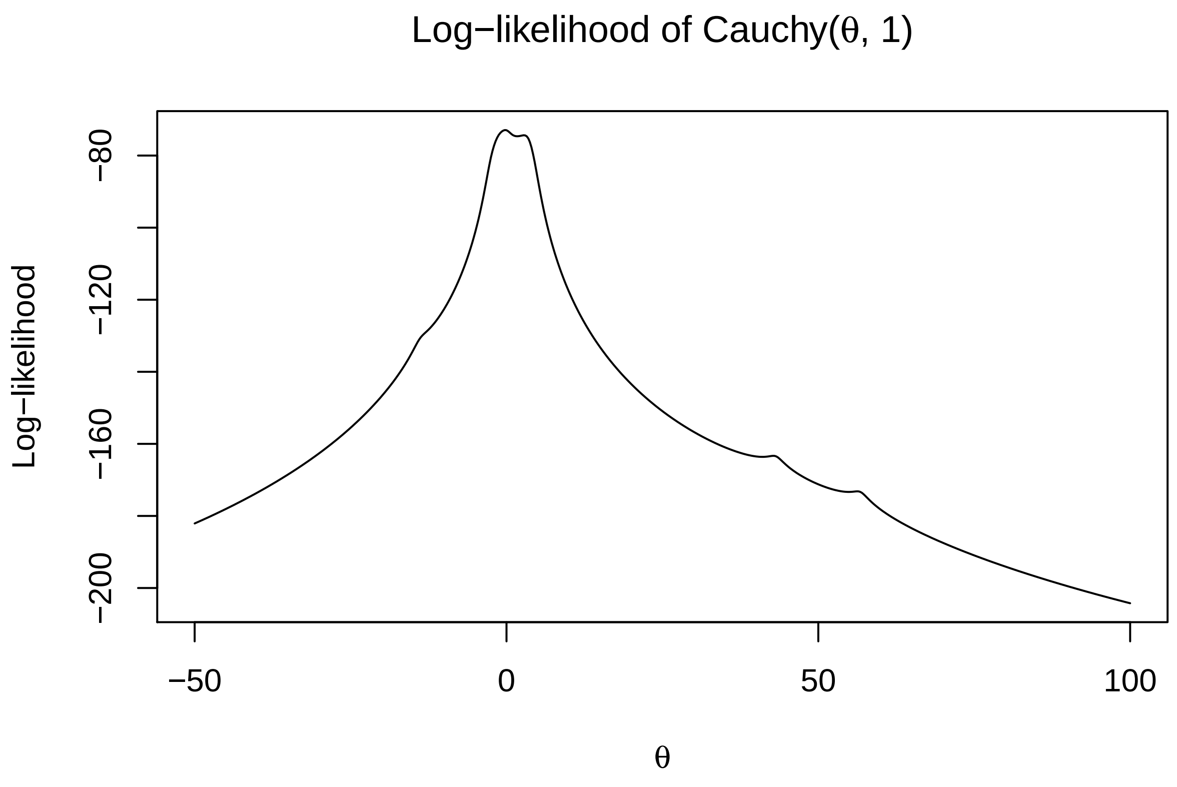
Therefore, the log-likelihood is

$$\begin{aligned} l(\theta) &= \Sigma_{x\in X} \ln\left(\frac{1}{\pi(1+(x-\theta)^2)}\right) \\ &= \Sigma_{x\in X} - \ln(\pi(1+(x-\theta)^2)) \\ &= -n\ln(\pi) - \Sigma_{x\in X} \ln(1+(x-\theta)^2), \end{aligned}$$

where n is the number of observations in X.

We plot the function below.

```r
X <- c(1.77, -.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24,
-2.44, 3.29, 3.71, -2.40, 4.53, -.07, -1.05, -13.87,
-2.53, -1.75, .27, 43.21)

log_likelihood <- function(theta, x){
  return (sum(dcauchy(x, location=theta, scale=1, log=TRUE)))
}

# Create a space for theta
theta_space <- seq(-50, 100, .25)

# Create the function results for theta
theta_f <- sapply(theta_space, function(theta){log_likelihood(theta, X)})

# Plot the likelihood function of theta
plot(theta_space, theta_f, 'l',
    main=TeX('Log-likelihood of Cauchy($\\theta$, 1)' ),
    xlab=TeX('$\\theta$'), ylab='Log-likelihood')
```



We note that the log-likelihood values can be negative, as they are not likelihoods, but rather the natural logarithms of those likelihoods.

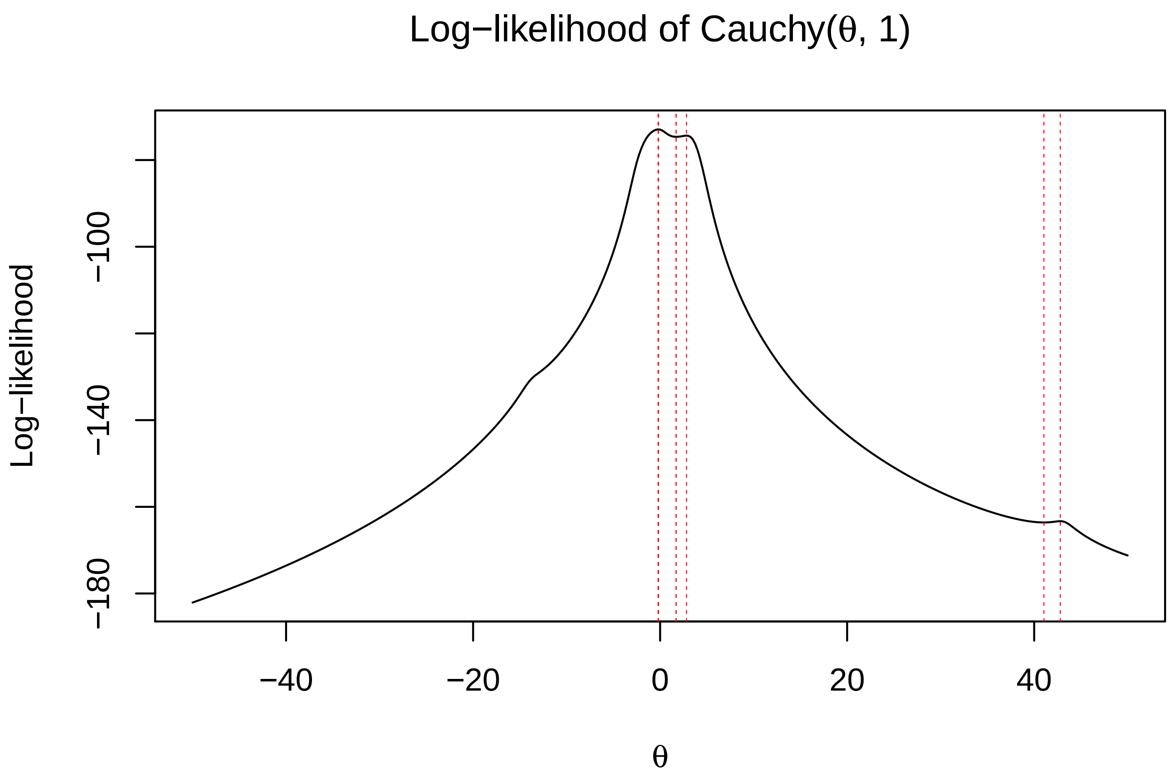
Next, we find the MLE for θ using Newton's Method for the set of starting values given above. We calculate the first two derivatives of the log-likelihood function.

$$
\begin{aligned}
l' &= -\Sigma_{x\in X} 2\frac{x-\theta}{1+(x-\theta)^2} \\
l'' &= -\Sigma_{x\in X} 2(1+(x-\theta)^2)^{-1} + -4(x-\theta)(1+(x-\theta)^2)^{-2}(x-\theta) \\
&= -\Sigma_{x\in X} \frac{2}{1+(x-\theta)^2} - \frac{4(x-\theta)^2}{(1+(x-\theta)^2)^2}
\end{aligned}
$$

```r
newtons <- function(xt, fprime, f2prime, tol=0.01){
  # Define the updating equation
  n <- 0
  xt_update <- xt + 100

  # While not below the tolerance level, continue updates
  while(abs(fprime(xt)) > tol){
    xt <- ifelse(n == 0, xt, xt_update)

    # Define the updating equation
    xt_update <- xt - (fprime(xt) / f2prime(xt))
    n <- n + 1
  }
  # If the adjustment value is less than the tolerance, end the iterations
  print(paste0("The solution is ", round(xt_update, 3) , " and it was found in ", n, " iterations."))
  return(xt_update)
}

# Define the first derivative
fprime <- function(theta) sum(2 * (X - theta) / (1 + (X - theta) ^ 2))

# Define the second derivative
f2prime <- function(theta){
  return(2 * sum (((X - theta) ^ 2 - 1) / (1 + (X - theta) ^ 2 ) ^ 2))
}

# Define the strating point, including mean and median
starting_points <- c(-11, -1, 0, 1.5, 4, 4.7, 7, 8, 38, mean(X), median(X))

# Call over all the starting points
solutions <- sapply(starting_points, function(x0){
  print(paste0("Starting Point: ", x0))
  newtons(x0, fprime=fprime, f2prime=f2prime, tol=.0001)
})
```

```
## [1] "Starting Point: -11"
## [1] "The solution is -821178.891 and it was found in 17 iterations."
## [1] "Starting Point: -1"
## [1] "The solution is -0.192 and it was found in 4 iterations."
## [1] "Starting Point: 0"
## [1] "The solution is -0.192 and it was found in 3 iterations."
## [1] "Starting Point: 1.5"
## [1] "The solution is 1.714 and it was found in 4 iterations."
## [1] "Starting Point: 4"
## [1] "The solution is 2.817 and it was found in 5 iterations."
## [1] "Starting Point: 4.7"
## [1] "The solution is -0.192 and it was found in 5 iterations."
## [1] "Starting Point: 7"
## [1] "The solution is 41.041 and it was found in 8 iterations."
## [1] "Starting Point: 8"
## [1] "The solution is -1145180.117 and it was found in 18 iterations."
## [1] "Starting Point: 38"
## [1] "The solution is 42.795 and it was found in 5 iterations."
## [1] "Starting Point: 5.106"
## [1] "The solution is 54.877 and it was found in 8 iterations."
## [1] "Starting Point: 1.02"
## [1] "The solution is 1.714 and it was found in 5 iterations."
```

```r
# Plot the likelihood function
theta_space <- seq(-50, 50, .25)
theta_f <- sapply(theta_space, function(theta){log_likelihood(theta, X)})
plot(theta_space, theta_f, 'l',
    main=TeX('Log-likelihood of Cauchy($\\theta$, 1)' ),
    xlab=TeX('$\\theta$'), ylab='Log-likelihood')

# Add in all solutions found
abline(v=solutions, col='red', lty=2, lwd=.5)
```



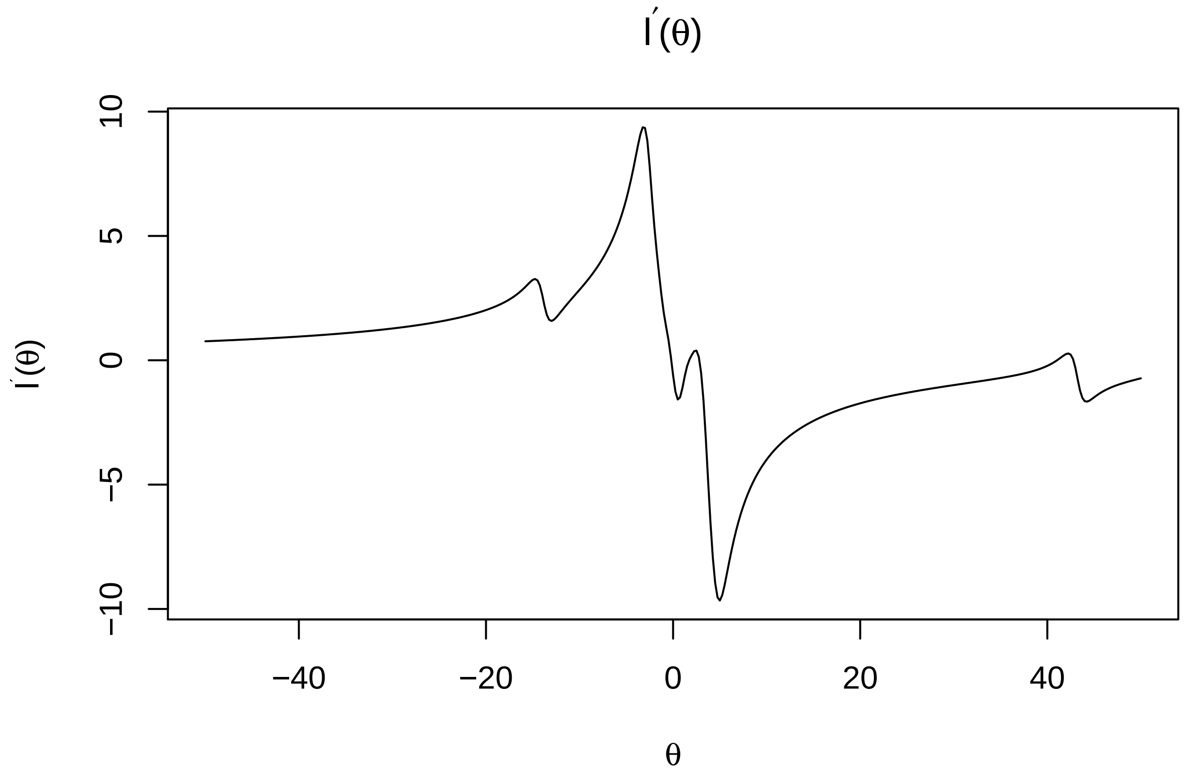
We get some strange results. Depending on where the search begins, the results are pretty wildly different. starting at 0, 1 and 4.7 yield the correct answer. Of the other ones, some find local maxima, and some just veer off to where the density is so low, that the derivative is basically zero and therefore below the tolerance.

We also try the mean and the median of the data sample as starting points. Neither of them find the actual global maximum. For a Cauchy dsitribution, the MLE for θ is actually the median, so the mean is somewhat lacking as an estimator here (in fact, the expected value of the distribution is undefined). Here, the median as a starting point actually outperforms the mean.

Here, we plot the first derivative, which gives some insight into the difficulty of solving this by Newton's method, which really just finds the roots of the first derivative.

```r
plot(theta_space, sapply(theta_space, fprime), 'l',
     main=TeX('$l^{\\prime}(\\theta)$'),
     xlab=TeX('$\\theta$'), ylab=TeX("$l^{\\prime}(\\theta)$")
```

```
    )
```



The log-liklihood function does not have a well behaving first and second derivative, and so Newton's method results in unstable solutions. With a likelihood function like this one, it's a good idea to run many processes over different starting points to find the correct solution.

b)

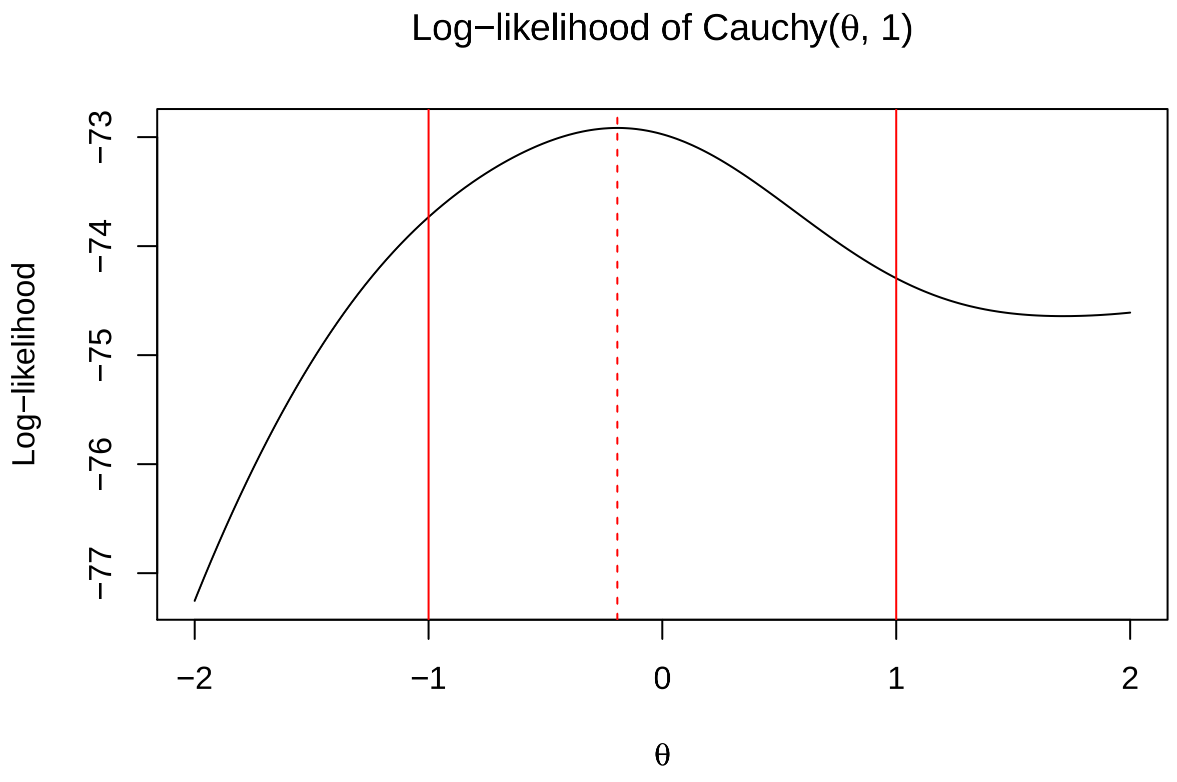
Next, we try the bisection method. We might expect this to work better, as we can bound the search to the part of the plot that we know contains the maximum.

```
bisection <- function(a, b, f_prime, tol=.0001, n=0){
  # Define the new split point
  x_t <- .5 * (a + b)
  # Use conditioning to get the next interval
  if(f_prime(a) * f_prime(x_t) <= 0){
    new_interval <- c(a, x_t)
  }else{
    new_interval <- c(x_t, b)
  }

  # if interval is less than the tolerance, stop the recursion.
  if ((b - a) < tol){
    print(paste0("The solution is ", round(x_t, 3) , " and it was found in ", n, " iterations."))
    return(x_t)
  }else{
    # If not, call again on the new interval
    return(bisection(new_interval[1], new_interval[2], f_prime, n=n + 1))
  }
}

# Define the observed values
X <- c(1.77, -.23, 2.76, 3.80, 3.47, 56.75, -1.34, 4.24,
       -2.44, 3.29, 3.71, -2.40, 4.53, -.07, -1.05,
```

```r
       -13.87, -2.53, -1.75, .27, 43.21)

# Define the derivative of the function
fprime <- function(theta) sum(2 * (X - theta) / (1 + (X - theta) ^ 2))

# Create the likelihood function over the space
theta_space <- seq(-2, 2, .01)
theta_f <- sapply(theta_space, function(theta){log_likelihood(theta, X)})

# Assign starting interval
a <- -1; b <- 1;

# Solve over that interval
solutions <- bisection(a, b, function(theta) fprime(theta), tol=.00001)
```

```
## [1] "The solution is -0.192 and it was found in 15 iterations."
```

```r
# Plot the likelihood function
plot(theta_space, theta_f, 'l',
    main=TeX('Log-likelihood of Cauchy($\\theta$, 1)' ),
    xlab=TeX('$\\theta$'), ylab='Log-likelihood')

# ADd the interval lines in black
abline(v=c(a, b), col='red')
# Add the solution lines in red
abline(v=solutions, col='red', lty=2)
```

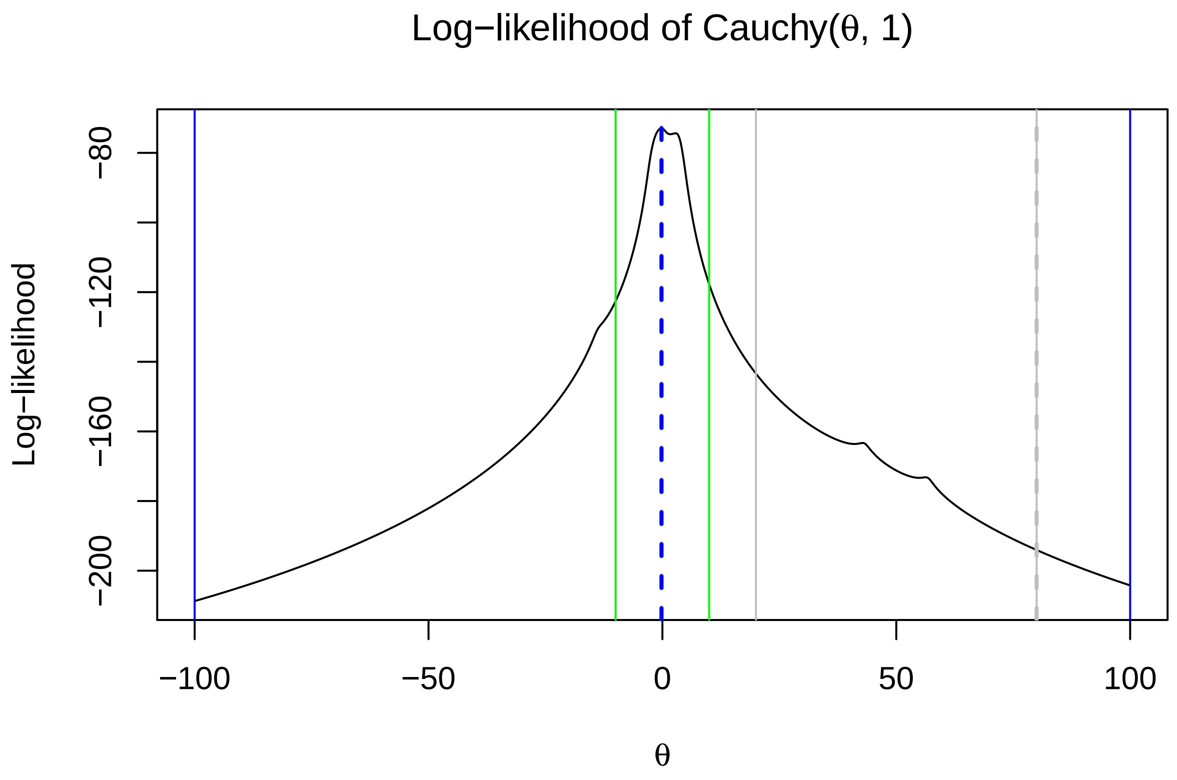


Using the bisection method yields much better results, as we converge on -.19. By visual inspection, this appears to be correct. It is worth noting that it takes many more iterations to converge than for the iterations of Newton's method that converged on the correct answer.

Next, we try several other intervals for the search, trying to trip up the algorithm.

```r
fprime <- function(theta) sum(2 * (X - theta) / (1 + (X - theta) ^ 2))
theta_space <- seq(-100, 100, .1)
theta_f <- sapply(theta_space, function(theta){log_likelihood(theta, X)})
```

```r
plot(theta_space, theta_f, 'l',
main=TeX('Log-likelihood of Cauchy($\\theta$, 1)'),
xlab=TeX('$\\theta$'), ylab='Log-likelihood')

# Iterate over the list of intervals
holder <- lapply(list(
        interval2 = c(-10, 10, 'green'),
        interval3 = c(-100, 100, 'blue'),
        interval4 = c(20, 80, 'gray')
     ),
     # Create a function to plot the results of the intervals
     function(interval){
       # Define the interval
       a <- as.numeric(interval[1]); b <- as.numeric(interval[2]);
       # Solve the problem
       solutions <- bisection(a, b, function(theta) fprime(theta), tol=.00001)

       # Add thin solid lines for interval bounds
       abline(v=c(a, b), col=interval[3])
       # Add thick dashed lines for solution
       abline(v=solutions, col=interval[3], lty=2, lwd=2)})
```



```
## [1] "The solution is -0.192 and it was found in 18 iterations."
## [1] "The solution is -0.192 and it was found in 21 iterations."
## [1] "The solution is 80 and it was found in 20 iterations."
```

Here, we plot the interval bounds as solid lines, and the solution as dashed lines. Each color represents a different interval. The intervals that include the solution solve the problem correctly. The interval that doesn't converges on one of the endpoints.

c)

We implement the fixed point iteration solver from starting point $x_0 = -1$, with scaling values $\alpha \in \{1, .64, .25\}$.

```
fixed_point_iteration <- function(x0, fprime, alpha=1, max_iter=100000, tol=.001){
  starting_point <- x0
  n <- 0
  xt <- x0
  while((n < 1 | abs(x0 - xt) > tol) & n < max_iter){
    # Update the point
    x0 <- xt

    # Define update equation
    xt <- x0 + (alpha * fprime(x0))

    n <- n + 1
  }
  print(paste0("The solution is: ", round(xt, 3), ". The algorithm converged in ", n, " iterations with start
  return(c(xt=xt,  n=n, starting_point=starting_point, alpha=alpha))
}

# Define the derivative of the function to optimize.

fprime <- function(theta) sum(2 * (X - theta) / (1 + (X - theta) ^ 2))

# start with x0 = -1.
results <- sapply(c(1, .64, .25), function(alpha){
  fixed_point_iteration(-1, fprime=fprime, alpha=alpha)
})
```

```
## [1] "The solution is: -0.193. The algorithm converged in 2294 iterations with starting point = -1 and alph
## [1] "The solution is: -0.193. The algorithm converged in 196 iterations with starting point = -1 and alpha
## [1] "The solution is: -0.192. The algorithm converged in 7 iterations with starting point = -1 and alpha =
```

All of the attempts converge on the correct solution, but the scaling parameters greatly affect the number of iterations needed to get there. We experiment with different starting points and scaling parameters below.

```
results <- lapply(c(-10, -5, -1, 1, 5, 10), function(x0){
  print(paste0("Starting Point: ", x0))
  return(lapply(c(.1, .3, .8, 1), function(alpha){
    fixed_point_iteration(x0, fprime=fprime, alpha=alpha, max_iter=100000)
  }))

})
```

```
## [1] "Starting Point: -10"
## [1] "The solution is: -0.194. The algorithm converged in 37 iterations with starting point = -10 and alpha
## [1] "The solution is: -0.192. The algorithm converged in 11 iterations with starting point = -10 and alpha
## [1] "The solution is: 0.372. The algorithm converged in 1e+05 iterations with starting point = -10 and alp
## [1] "The solution is: 2.642. The algorithm converged in 1e+05 iterations with starting point = -10 and alp
## [1] "Starting Point: -5"
## [1] "The solution is: -0.194. The algorithm converged in 24 iterations with starting point = -5 and alpha
## [1] "The solution is: -0.192. The algorithm converged in 5 iterations with starting point = -5 and alpha =
## [1] "The solution is: -0.802. The algorithm converged in 1e+05 iterations with starting point = -5 and alp
## [1] "The solution is: -0.192. The algorithm converged in 7 iterations with starting point = -5 and alpha =
## [1] "Starting Point: -1"
## [1] "The solution is: -0.194. The algorithm converged in 18 iterations with starting point = -1 and alpha
## [1] "The solution is: -0.192. The algorithm converged in 5 iterations with starting point = -1 and alpha =
## [1] "The solution is: -0.802. The algorithm converged in 1e+05 iterations with starting point = -1 and alp
## [1] "The solution is: -0.193. The algorithm converged in 2294 iterations with starting point = -1 and alph
## [1] "Starting Point: 1"
## [1] "The solution is: -0.19. The algorithm converged in 20 iterations with starting point = 1 and alpha =
## [1] "The solution is: -0.192. The algorithm converged in 6 iterations with starting point = 1 and alpha =
## [1] "The solution is: -0.802. The algorithm converged in 1e+05 iterations with starting point = 1 and alph
## [1] "The solution is: -0.192. The algorithm converged in 9344 iterations with starting point = 1 and alpha
## [1] "Starting Point: 5"
## [1] "The solution is: 2.82. The algorithm converged in 20 iterations with starting point = 5 and alpha = 0
## [1] "The solution is: 2.817. The algorithm converged in 13 iterations with starting point = 5 and alpha =
## [1] "The solution is: 2.817. The algorithm converged in 22 iterations with starting point = 5 and alpha =
## [1] "The solution is: 2.929. The algorithm converged in 1e+05 iterations with starting point = 5 and alpha
## [1] "Starting Point: 10"
## [1] "The solution is: 2.82. The algorithm converged in 29 iterations with starting point = 10 and alpha =
## [1] "The solution is: 2.818. The algorithm converged in 9 iterations with starting point = 10 and alpha =
## [1] "The solution is: 0.372. The algorithm converged in 1e+05 iterations with starting point = 10 and alph
## [1] "The solution is: -0.193. The algorithm converged in 21769 iterations with starting point = 10 and alp
```

We can see that the starting point and scaling parameters heavily impacts the convergence of the algorithm to the correct solution. Smaller scaling parameters tend to work better, with with those around .3 usually converging the fastest. Additionally, a scaling parameter of .8 reliably fails from all starting points. This reiterates the need to run parallel processes across various parameters to get a sense for the true result.

d)

Using starting values $(\theta^{(0)}, \theta^{(1)}) \in \{(-2, -1), (-3, 3)\}$, we apply the secant method to estimate θ.

```r
secant_method <- function(x0, x1, fprime, max_iter=100000, tol=.001){
  n <- 0

  while((n < 1 | abs(x0 - x1) > tol) & n < max_iter){
    # Define the updating equation
    xt <- x1 - fprime(x1) * (x1 - x0) / (fprime(x1) - fprime(x0))

    # Iterate on update
    x0 <- x1; x1 <- xt;

    n <- n + 1
  }
  print(paste0("The solution is: ", xt, ". The algorithm converged in ", n, " iterations"))

  return(xt)
}

print("Starting points of (-2, 1)")
```

```
## [1] "Starting points of (-2, 1)"
```

```r
results1 <- secant_method(-2, -1, fprime=fprime)
```

```
## [1] "The solution is: -0.192286493714121. The algorithm converged in 5 iterations"
```

```r
print("Starting points of (-3, 3)")
```

```
## [1] "Starting points of (-3, 3)"
```

```r
results2 <- secant_method(-3, 3, fprime=fprime)
```

```
## [1] "The solution is: 2.81746622667055. The algorithm converged in 5 iterations"
```

```r
print("Starting Points of (-2, 3)")
```

```
## [1] "Starting Points of (-2, 3)"
```

```r
results3 <- secant_method(-2, 3, fprime=fprime)
```

```
## [1] "The solution is: 2.81747656517513. The algorithm converged in 6 iterations"
```

```
print("Starting Points of (-4, -4.5)")
```

```
## [1] "Starting Points of (-4, -4.5)"
```

```
results3 <- secant_method(-4, -4.5, fprime=fprime)
```

```
## [1] "The solution is: -1.94676563893071e+154. The algorithm converged in 738 iterations"
```

We see a pretty wide variety of results, entirely dependant on the parameterization of the starting points. The takeaway here may be that this method requires a very careful selection of starting points, based on the plot of the function being maximized. The first set of points converges quickly and to the correct answer. The rest do not. The 2nd pair of points given in the text find a different root of the derivative. Some other points tried find the wrong root, as well as run off to divergence before stopping when the likelihood function levels off.

e)

To compare the results takes some nuance, as different algorithms excel in different ways.

1. The bisection method has stability, but comes with a performance drawback.
2. Fixed point iteration is the most unstable, as various starting points and scaling parameters simply fail, with small changes in inputs leading to large changes in outputs.
3. Newton's Method is extremely fast when it starts in the right place and isn't tripped up by unusual function behavior. Again, a win in performance, but less so in stability.
4. The secant method performs well when it finds the right thing, but it often doesn't, and the starting points appear to really tank the performance. It seems difficult to pick starting points that work, even when examining the function manually.

Problem 4

a) We find the MLE for λ, given a Poisson distribution. The PDF of a Poisson distribution is

$$f(x;\lambda) = \frac{\lambda^x \mathrm{e}^{-\lambda}}{x!}.$$

Therefore, the likelihood function $L(\lambda)$ is the product of the likelihods for each data sample:

$$L(\lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!}.$$

and the log-likelihood can be written as the sum of the logs:

$$\begin{aligned} l(\lambda) &= \log \prod_{i=1}^{n} \frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!} \\ &= \Sigma_{i=1}^{n} \log \lambda^{x_i} - \Sigma_{i=1}^{n} \log X_i! - \Sigma_{i=1}^{n} \lambda \\ &= \Sigma_{i=1}^{n} x_i \log \lambda - \Sigma_{i=1}^{n} \log X_i! - n\lambda. \end{aligned}$$

We find the first derivative of the log-likelihood function:

$$l'(\lambda) = \Sigma_{i=1}^{n} \frac{x_i}{\lambda} - n,$$

and then find the root:

$$0 = \Sigma_{i=1}^{n} \frac{x_i}{\lambda} - n \implies \lambda = \Sigma_{i=1}^{n} \frac{x_i}{n}.$$

Therefore, the MLE for λ is $\Sigma_{i=1}^{n}(x_i)/n$, which is the mean of the data sample. This is as expected, as the expected value of a Poisson distribution is λ.

b)

Next, we look for the MLE of θ in an Exponential Distribution. The PDF of an Exponential distribution is

$$f(x;\theta) = \theta \mathrm{e}^{-\theta \mathrm{x}}.$$

We can write the likelihood function for θ as

$$L(\theta) = \prod_{i=1}^{n} \theta \mathrm{e}^{-\theta \mathrm{x_i}},$$

with the log-likelihood as

$$\begin{aligned} l(\theta) &= \log \prod_{i=1}^{n} \theta \mathrm{e}^{-\theta \mathrm{x_i}} \\ &= \Sigma_{i=1}^{n} \log \theta + \Sigma_{i=1}^{n} -\theta x_i \\ &= n \log \theta - \theta \Sigma_{i=1}^{n} x_i. \end{aligned}$$

To find the MLE, we find the root of the first derivative:

$$0 = l'(\theta) = \frac{n}{\theta} - \Sigma_{i=1}^{n} x_i \implies \theta = \frac{n}{\Sigma_{i=1}^{n} x_i}.$$

Solving this equation gives us the MLE for θ as $\theta = \frac{n}{\Sigma_{i=1}^{n} x_i}$. This is as expected, as the expected value of an Exponential(θ) distribution is $\frac{1}{\theta}$, and so the MLE for θ is the reciprocal of the mean.

Problem 5

To find the Fisher Information $I(\theta)$, we use the formula

$$I(\theta) = E[(l'(\theta))^2].$$

First, we find $L(\theta)$ for a set of n independent Bernoulli trials with parameter θ. This is simply the likelihood function for a Binomial distribution:

$$L(\theta; x, n) = \binom{n}{x} \theta^x (1-\theta)^{n-x},$$

with log-likelihood

$$l(\theta) = \log \binom{n}{k} + x \log \theta + (n-x) \log (1-\theta).$$

We first take the first derivative of the log-likelihood function,

$$l'(\theta) = \frac{x}{\theta} - \frac{n-x}{1-\theta} = \frac{x(1-\theta)}{\theta(1-\theta)} - \frac{n\theta - x\theta}{\theta(1-\theta)} = \frac{x-n\theta}{\theta(1-\theta)},$$

and then square it

$$\left(\frac{x-n\theta}{\theta(1-\theta)}\right)^2 = \frac{x^2}{\theta^2(1-\theta)^2} - \frac{2nx}{\theta(1-\theta)^2} - \frac{n^2}{(1-\theta)^2}.$$

We then take the expected value of this. Using the algebraic properties of the Expected Value function, we can write

$$E((l'(\theta)^2) = \frac{1}{\theta^2(1-\theta)^2} E(X^2) - \frac{2n}{\theta(1-\theta)^2} E(X) + \frac{n^2}{(1-\theta)^2}.$$

We site the textbook's table of distribution to define the following:

$$E(X) = n\theta$$
$$Var(X) = n\theta(1-\theta).$$

Note that $Var(X) = E(X^2) - E(X)^2$, and so $E(X^2) = Var(X) + E(X)^2 = n\theta(1-\theta) + n^2\theta^2$. We can then plug these into the formulas above, such that

$$\begin{aligned}
E((l'(\theta)^2) &= \frac{1}{\theta^2(1-\theta)^2}E(X^2) - \frac{2n}{\theta(1-\theta)^2}E(X) + \frac{n^2}{(1-\theta)^2} \\
&= \frac{n\theta(1-\theta) + n^2\theta^2}{\theta^2(1-\theta)^2} - \frac{2n^2\theta^2}{\theta^2(1-\theta)^2} + \frac{n^2\theta^2}{\theta^2(1-\theta)^2} \\
&= \frac{n\theta(1-\theta)}{\theta^2(1-\theta)^2} \\
&= \frac{n}{\theta(1-\theta)}.
\end{aligned}$$

Therefore, the Fisher Information can be described as $I(\theta) = \frac{n}{\theta(1-\theta)}$.