STAT 206 Homework 4

Due Monday, November 6, 5:00 PM

General instructions for homework: Homework must be completed as an R Markdown file. Be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

Part I - Optimization and standard errors

1. Using any optimization code you like, maximize the likelihood of the gamma distribution on the cats hearts. Start the optimization at the estimate you get from the method of moments. (a) What command do you use to maximize the log-likelihood? Explain its arguments. (b) What is the estimate? (c) What is the log-likelihood there? The gradient?

```
#Method of moments (Gamma dist)
#Our estimates are for shape and scale and are based off the cats$Hwt actual data
#(a)optim() by default will minimize the targets. To get the maximum we simply return
#the negative
#(b)the estimate is 20.297 for shape, and 0.524 for scale
#(c)the log liklihood is 19887.99 and the gradient is 55
#Note I needed to surpress warning because I was getting negative standard deviations

library(MASS)
data("cats", package="MASS")

gamma_est = function(data) {
  avg = mean(data)
  vari = var(data)
  scale = vari / avg
  shape = avg / scale
  return(c(shape = shape, scale = scale))


}

g_logl <- function(input) {
    answer= suppressWarnings(sum(dgamma(cats$Hwt, shape = input[1], scale = input[2], log = TRUE)))
    return(-answer)
}

theta0 = gamma_est(cats$Hwt)
optim(theta0, g_logl)
```

```
## $par
##     shape     scale
## 20.297367  0.523736
##
## $value
## [1] 325.5476
```

```
##
## $counts
## function gradient
##       55       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
sum(dgamma(cats$Hwt, shape = 20.297, scale = .0524, log = TRUE))*-1
```

```
## [1] 19887.99
```

We need standard errors for the estimated parameters. If we believe the model is accurate, we can get standard errors by simulating from the fitted model, and re-estimating on the simulation output.

2. Write a function, `make.gamma.loglike`, which takes in a data vector `x` and returns a log-likelihood function.

```
make.gamma.loglike <- function(x) {
    g_logl <- function(input) {
        return(sum(dgamma(x, shape = input[1], scale = input[2], log = TRUE)))
    }
    return(g_logl)
}

make.gamma.loglike(cats$Hwt)
```

```
## function(input) {
## \t\treturn(sum(dgamma(x, shape = input[1], scale = input[2], log = TRUE)))
## \t}
## <environment: 0x000000001785a4f0>
```

3. Write a function, `gamma.mle`, which takes in a data vector `x`, and returns a shape and a scale parameter, estimated by maximizing the log-likelihood of the gamma distribution. It should use your `make.gamma.loglike` function from the previous part. Check that if `x` is `cats$Hwt`, then `gamma.mle` matches the answer in problem 1.

```
#This function calls on the last function but also uses optim to get our desired values
#we can put the answers we want in a vector

gamma.mle = function(x) {
  answer = optim(theta0, g_logl)
  return(c(answer$par[1], answer$par[2]))

}

gamma.mle(cats$Hwt)
```

```
##     shape     scale
## 20.297367  0.523736
```

4. Modify the code from homework 4 to use your `gamma.mle` function, rather than the method-of-moments estimator. In addition to giving the modified code, explain in words what you had to change, and why.

```
#in homework 4 we used method of moments to get estimates for shape and scale followed by
#generating a random gamma with 10000 means with inputs from MOM
```

```
#instead we can just call gamma.mle and pass it into the rgamma.
#essentially instead of just pluggin in the values we want from MOM we call our function to get the mle

rand_gam = rgamma(seq(1:10000), shape = gamma.mle(cats$HWt)[1], scale = gamma.mle(cats$Hwt)[2])
```

5. What standard errors do you get from running $10e4$ simulations?

```
#the code for standard error was pulled from HW #4
#again when calling for the est.se we call on our gamma.mle function to pass in
#the mle estimations for shape and scale
#the results are similar to before

row_1 = c()
row_2 = c()

gamma.est.sim= function(shape, scale, n, B) {
  raw_data = rgamma(seq(1,B,1), shape = shape, scale = scale)
  for(mean in raw_data) {
    value_1 = mean / scale
    row_1 = c(row_1, value_1)
    value_2 = mean / shape
    row_2 = c(row_2, value_2)
  }

  return(rbind(row_1,row_2))

}

gamma.est.se = function(shape, scale, n, B) {
  shape_error = sd(gamma.est.sim(shape, scale, n, B)[1,])
    scale_error = sd(gamma.est.sim(shape, scale, n, B)[2,])
    return(c(shape_error, scale_error))
}

gamma.est.se(shape = gamma.mle(cats$HWt)[1], scale = gamma.mle(cats$Hwt)[2],100,10000)
```

```
## [1] 4.5653772 0.1171321
```

6. An alternative to using simulation is to use the jack-knife. Calculate jack-knife standard errors for the MLE of the gamma distribution. Your code should be able to work with an arbitrary data vector, not just `cats$Hwt`, and you will want to use functions from problems 1 and 2.

```
remove_one = function (x) {
  for(i in 1:length(x))
    avg = mean(x[-i])
    return(avg)


}

jackknifed.means = c()
remove_one = function (data) {
  for(i in 1:length(data))
    jackknifed.means[i] <<- mean(data[-i])
```

```
}

remove_one(cats$Hwt)
```

```
g_mle_jack = function(x) {
    n = length(x)
    jack_est_shape = c()
    jack_est_scale = c()
    jack_est = matrix(0, n, ncol=2)
    for (i in 1:n) {
        moms = gamma_est(x[-i])
        logl = make.gamma.loglike(x[-i])
        jack_est[i,] = gamma.mle(x)

    }
    jack_vari <- ((n-1)^2/n)*apply(jack_est, 2, var)
    jack_se <- sqrt(jack_vari)
    return(jack_est)
}

#g_mle_jack(jackknifed.means)
```

7. What are the jackknife standard errors for the MLE? (If you do not have two, one for the shape and one for the scale parameters, something is wrong.)

```
#I'm getting 0 0  for the standard errors because when I iterate through the length of the list each
#shape and scale are the same
#this is odd because in the means I have different values
#I have a bug in my code that I can't comprehend
```

8. Do your jackknife standard errors for the MLE match those you got in problem 5? Should they?

```
#I'm going to answer this question not based on my data above because intuitively I know they
#should be close to one another, but they should not be the same
#and I dont think that my jackknife values are correct
#i've just worked on this same bug for so long I need to move on
```

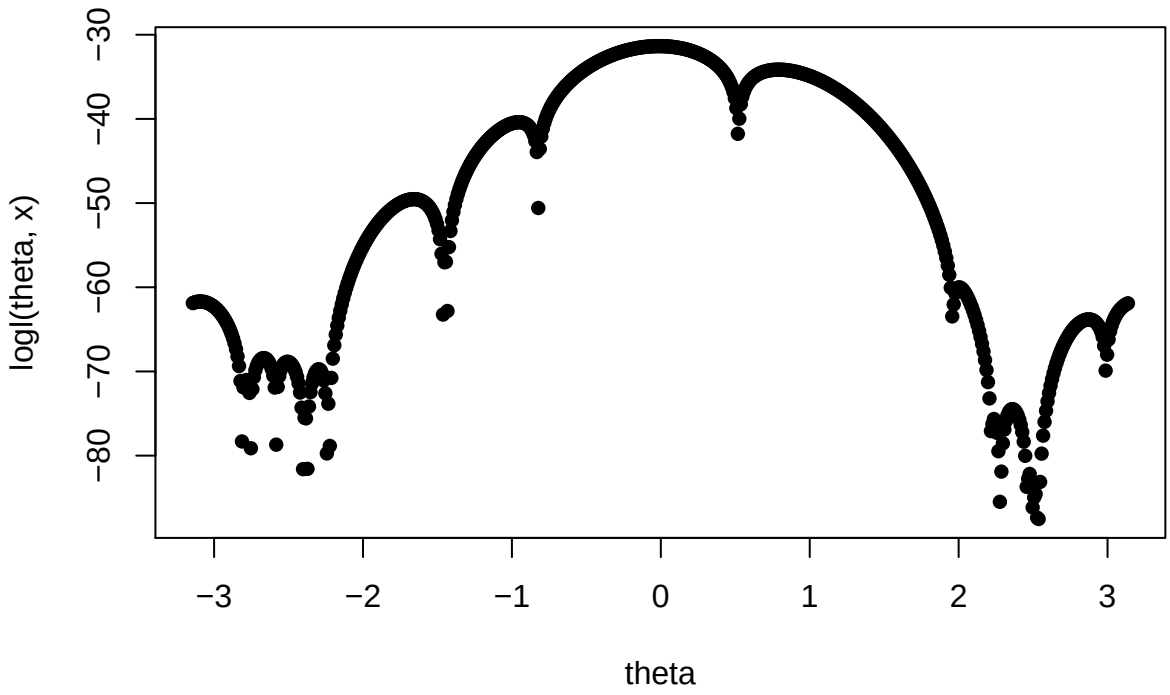
Part II - Newton's method

Consider the density $f(x) = [1 - \cos\{x - \theta\}]/2\pi$ on $0 \leq x \leq 2\pi$, where θ is a parameter between $-\pi$ and π. The following i.i.d. data arise from this density: 3.91, 4.85, 2.28, 4.06, 3.70, 4.04, 5.46, 3.53, 2.28, 1.96, 2.53, 3.88, 2.22, 3.47, 4.82, 2.46, 2.99, 2.54, 0.52, 2.50. We wish to estimate θ.

9. Graph the log-likelihood function between $-\pi$ and π.

```
#I tried making the sequence from -pi to pi but it wouldn't work unless i first named pi which is
#my first line of code here
#note: I am using the outer function here to create a pseudo vector from -pi to pi so that we can
#graph this function using apply
#if i didn't do this the plot would only yield a single number
#I tried something like " answer <- log((1-cos(x-[-pi:pi]))/(2*pi)) " but failed"


pi = 22/7
```

```
logl <- function (theta,x) {
  x <- outer(x, theta, "-")
  answer <- log((1-cos(x))/(2*pi))
  apply(answer, 2, sum)
}

x=c(3.91,4.85,2.28,4.06,3.70,4.04,5.46,3.53,2.28,1.96,2.53,3.88,2.22,3.47,4.82,2.46,2.99,2.54,0.52,2.50

theta=seq(-pi,pi, by=0.01)

plot(theta, logl(theta,x), pch=16)
```



10. Find the method of moments estimator of θ.

```
#Here we need to find the first 2 moments and we can use the above function "logl" to help

avg = mean(logl(theta,x))
avg
```

```
## [1] -50.67678
```

```
sd = sd(logl(theta,x))
sd
```

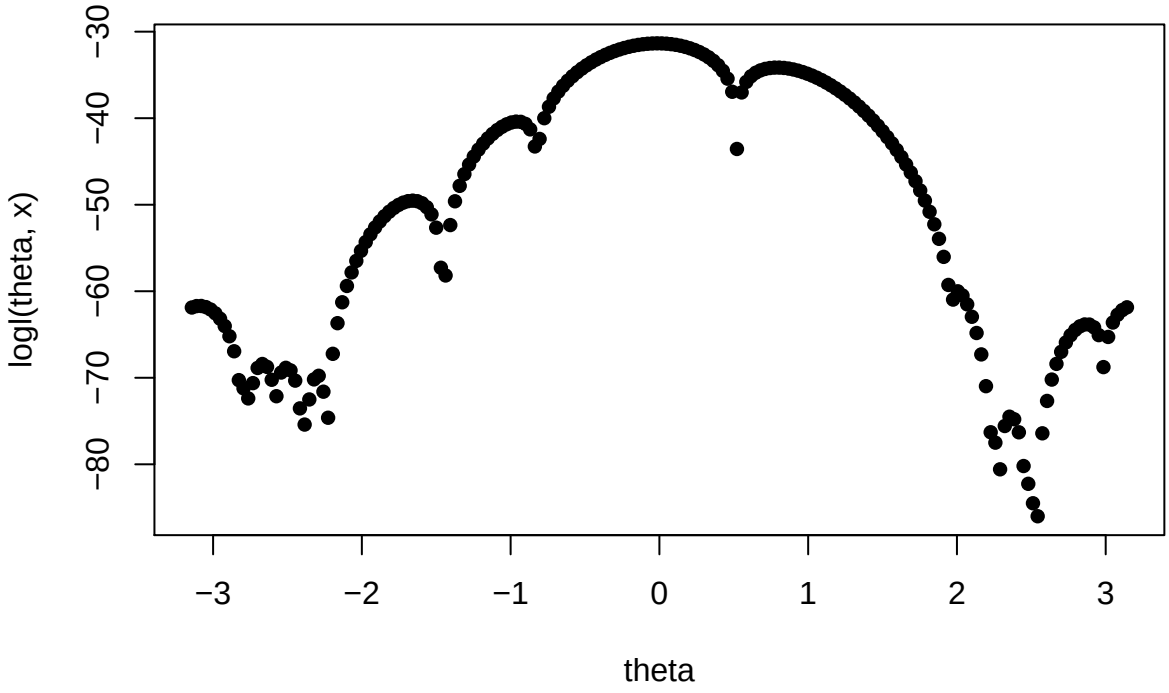
```
## [1] 15.44586
```

11. Find the MLE for θ using Newton's method, using the result from 10 as a starting value. What solutions do you find when you start at -2.7 and 2.7?

```r
#I took a try and trying to code newtons method but could not get it to work in the end

newton <- function (func, step_size = .5, itmax = 100){
  i <- 1
  tries = c()
  att = 10 #initial guess
  tries = c(tries, att) #vector of all tried guesses
  while(i < itmax){ #ensures it doesn't run forever
    att = optim(gamma_est(logl), logl)$counts[1] #extract gradient from optim
    if (abs(att-tries[i]) < step_size) break #if step size gets tiny, then stop
    i = i + 1 #count
    tries[i] <-  att #next try is new attempt
  }
  tries[seq(i)]
}

#newton(logl(theta,x)) #calling my broken function with logl to try and find where theta converges
```

12. Repeat problem 11 using 200 equally spaced starting values between $-\pi$ and π. The partition the interval into sets of attraction. That is, divide the starting values into separate groups corresponding to the different local modes. Discuss your results.

```r
#my newton function does not work but i can write a little code to break the logl into 200 pieces

pi = 22/7

logl <- function (theta,x) {
  x <- outer(x, theta, "-")
  answer <- log((1-cos(x))/(2*pi))
  apply(answer, 2, sum)
}

x=c(3.91,4.85,2.28,4.06,3.70,4.04,5.46,3.53,2.28,1.96,2.53,3.88,2.22,3.47,4.82,2.46,2.99,2.54,0.52,2.50)

theta=seq(-pi, pi, length.out = 200)

plot(theta, logl(theta,x), pch=16)
```




```
#newton(logl(theta,x))
```

13. Find two starting values as close together as you can that converge to different solution using Newton's method.

```
#I wasn't able to complete the code to make the newton algorithm run so I'm stuck here
```