Stat 135 Lab 1 - Jin Kweon

Jin Kweon

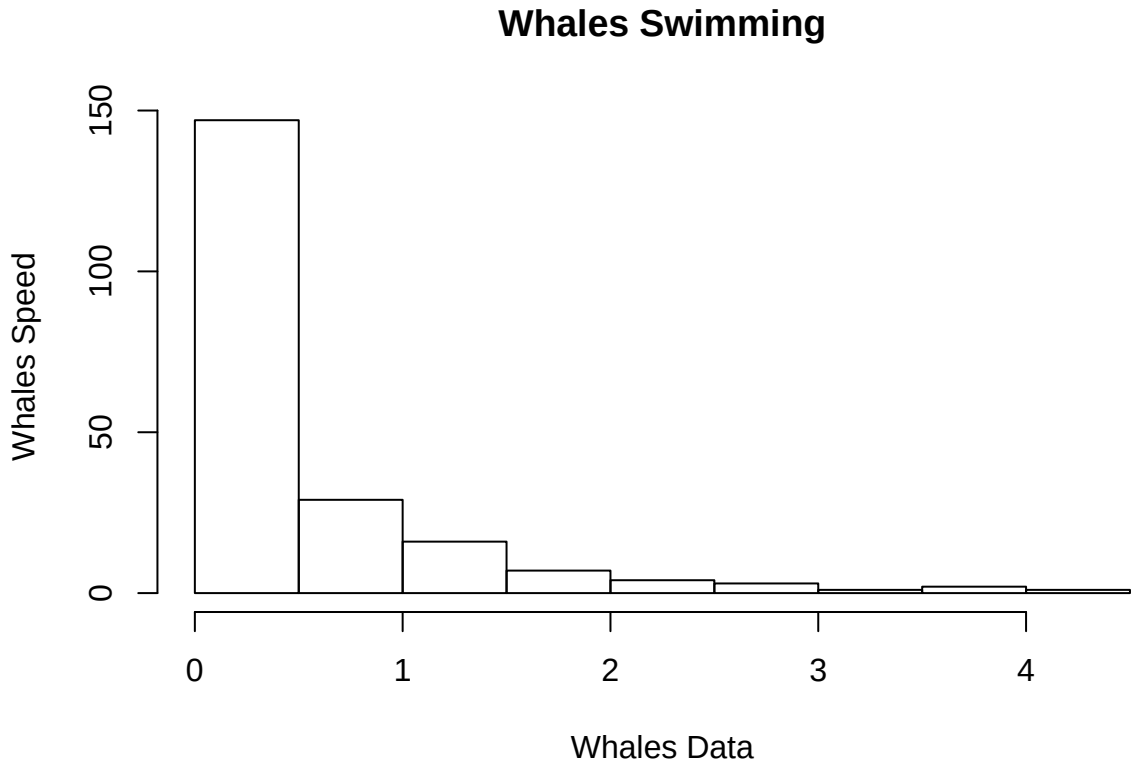
2/13/2017

Data import

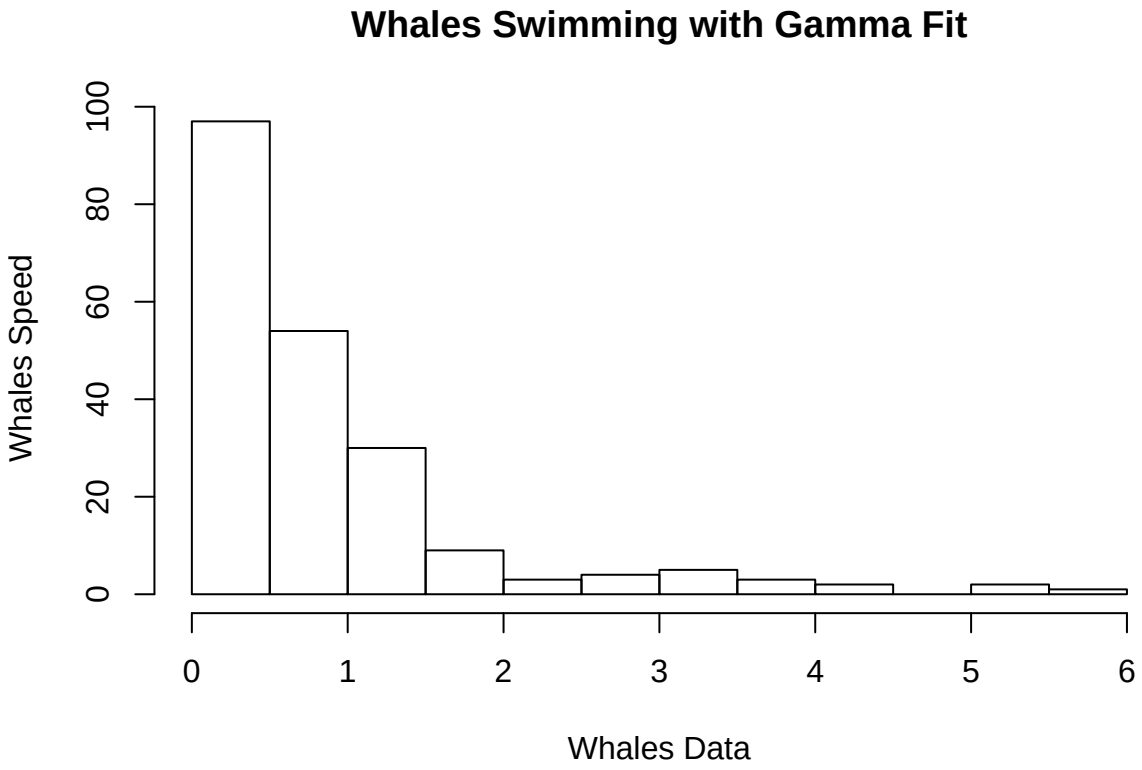
```
whales <- read.csv("/Users/yjkweon24/Desktop/Cal/2017Spring/Stat 135/Data/whales.txt", header = FALSE, c
colnames(whales) <- "Data"
options(digits=9, stringsAsFactors = FALSE)
whales$Data <- as.numeric(whales$Data)
newdata <- as.numeric(whales[[1]])
```

A

```
hist(newdata, main = "Whales Swimming", xlab = "Whales Data", ylab = "Whales Speed")
```



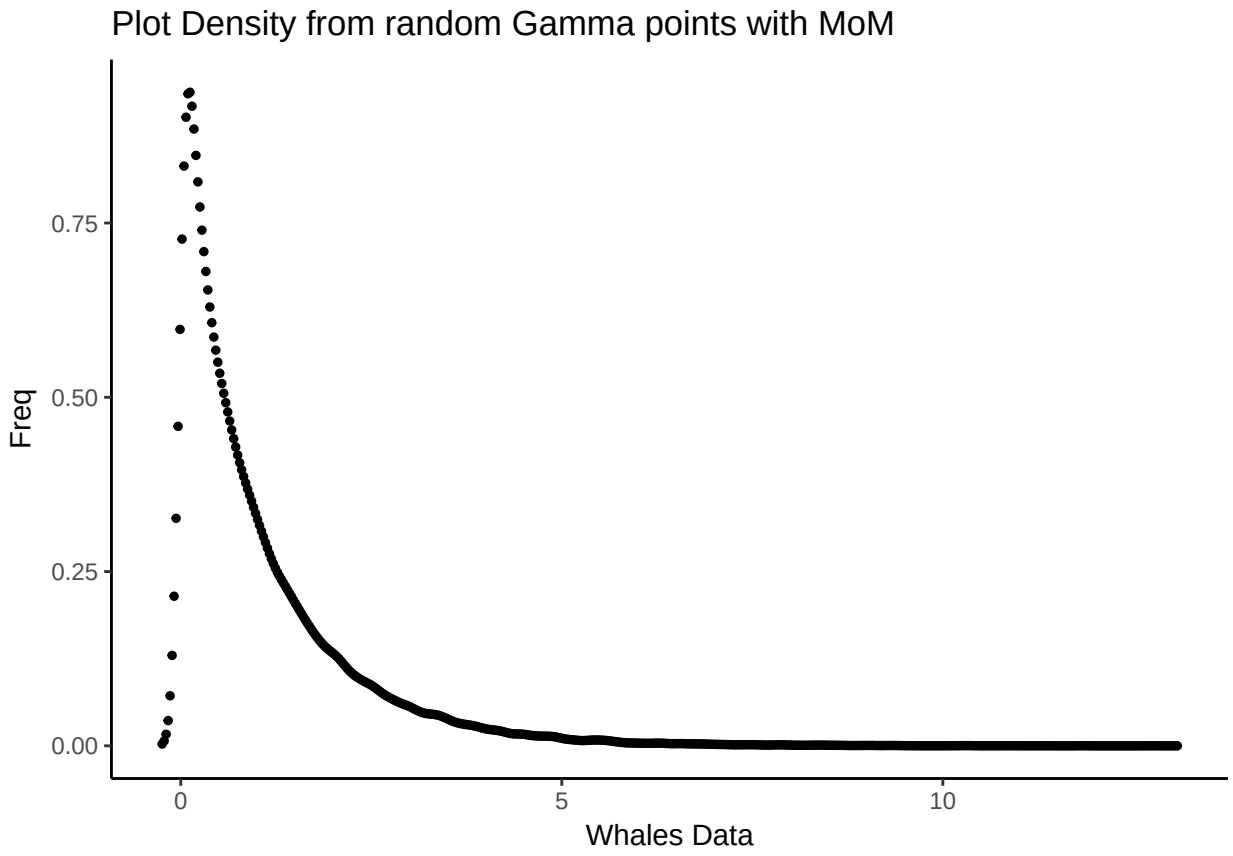

```
hist(rgamma(newdata, 0.9), main = "Whales Swimming with Gamma Fit", xlab = "Whales Data", ylab = "Whales
```



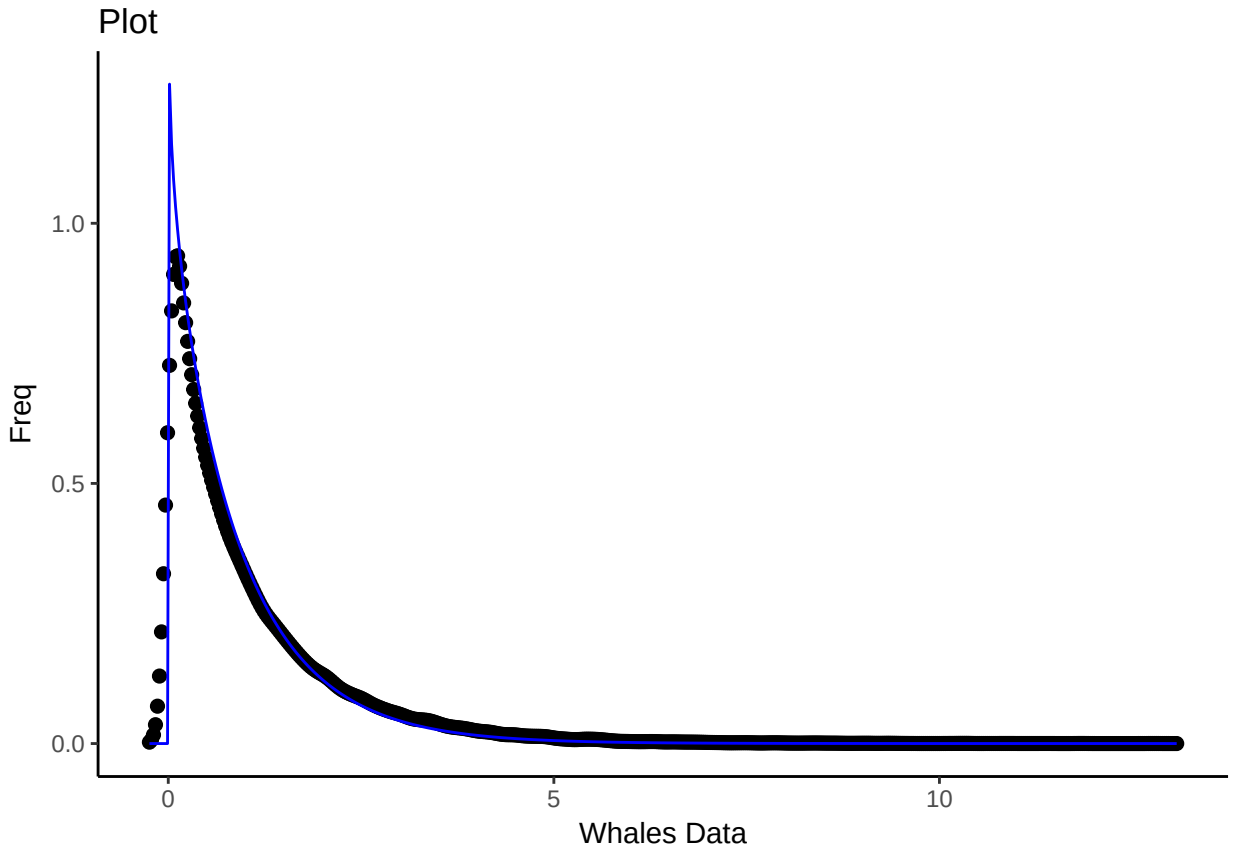

```
# So, I can say the histogram looks like it fits into the gamma distribution.
# The left portion is much bigger compared to the rightmost portions. The frequency gets smaller rapidl
# To prove my statement, I used rgamma function and get the data, and plot them as histogram.
```

B

```
# Method of Moments Way
# when I say x ~ gamma(alpha, lambda), and x_bar = mean of sample space (210), expectation(x^2) = get2,
# alph(a) = (x_bar)^2 / [get2 - (x_bar)^2]  and
# lambd(a) = (x_bar) / [get2 - (x_bar)^2]
x_bar = mean(newdata)
get2 = (x_bar)^2 + var(newdata)
alph = (x_bar)^2 / (get2 - (x_bar)^2) # Get Alpha.
lambd = (x_bar) / (get2 - (x_bar)^2) # Get Lambda

x <- rgamma(100000, shape=alph, scale=lambd) # Randomly get 100000 data from the alpha and lambda I got
densit <- density(x)
datas <- data.frame(x = densit$x, y = densit$y)

# Plot density from Rgamma points with MoM
ggplot(data = datas, aes(x = x, y = y)) +
  geom_point(size = 1) +
  theme_classic() + xlab("Whales Data") + ylab("Freq") + ggtitle("Plot Density from random Gamma points
```



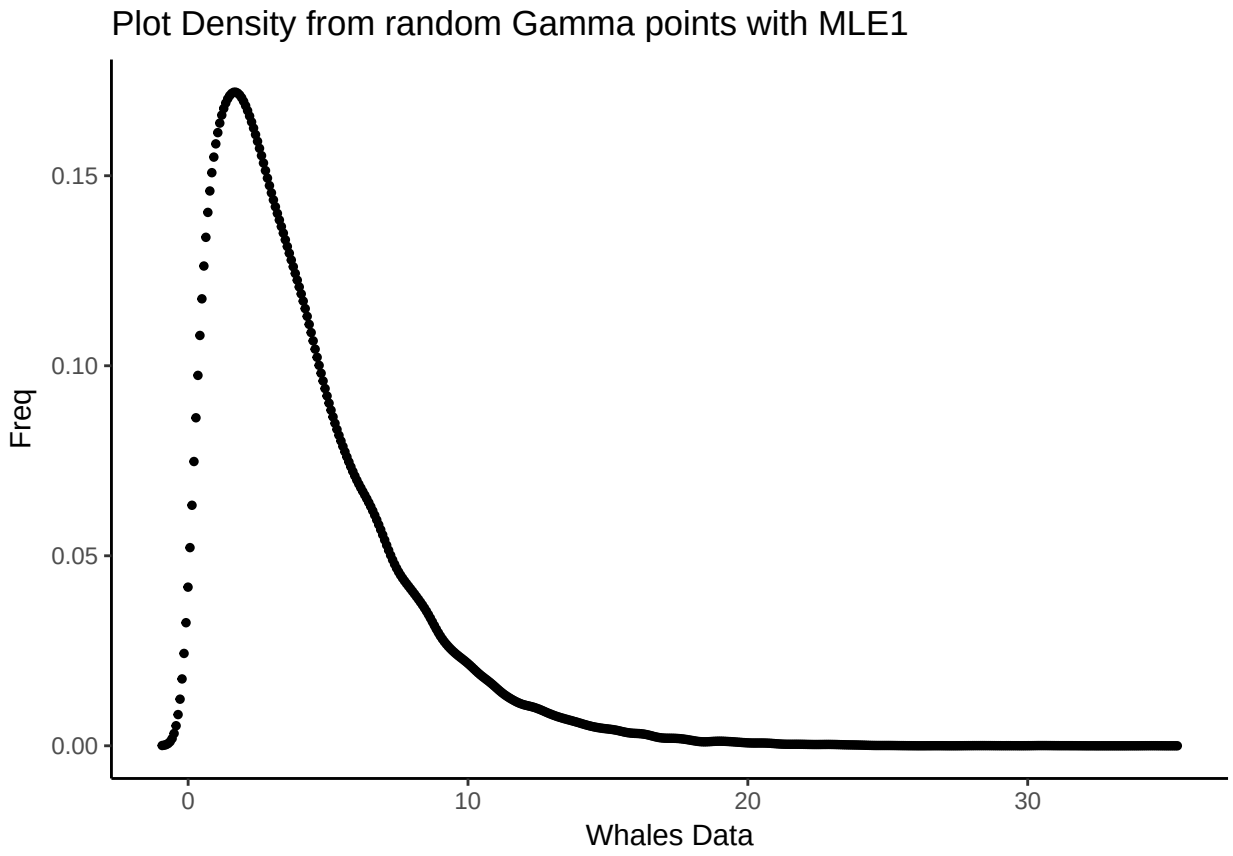

```
#Fit Parameters - check Fit
fit.params <- fitdistr(x, "gamma", lower = c(0, 1))

# Plot with density points - Fitparams into the MOM density
ggplot(data = datas, aes(x = x,y = y)) +
  geom_point(size = 2) +
  geom_line(aes(x=datas$x, y=dgamma(datas$x, fit.params$estimate["shape"], fit.params$estimate["rate"])))
  theme_classic() + xlab("Whales Data") + ylab("Freq") + ggtitle("Plot")
```



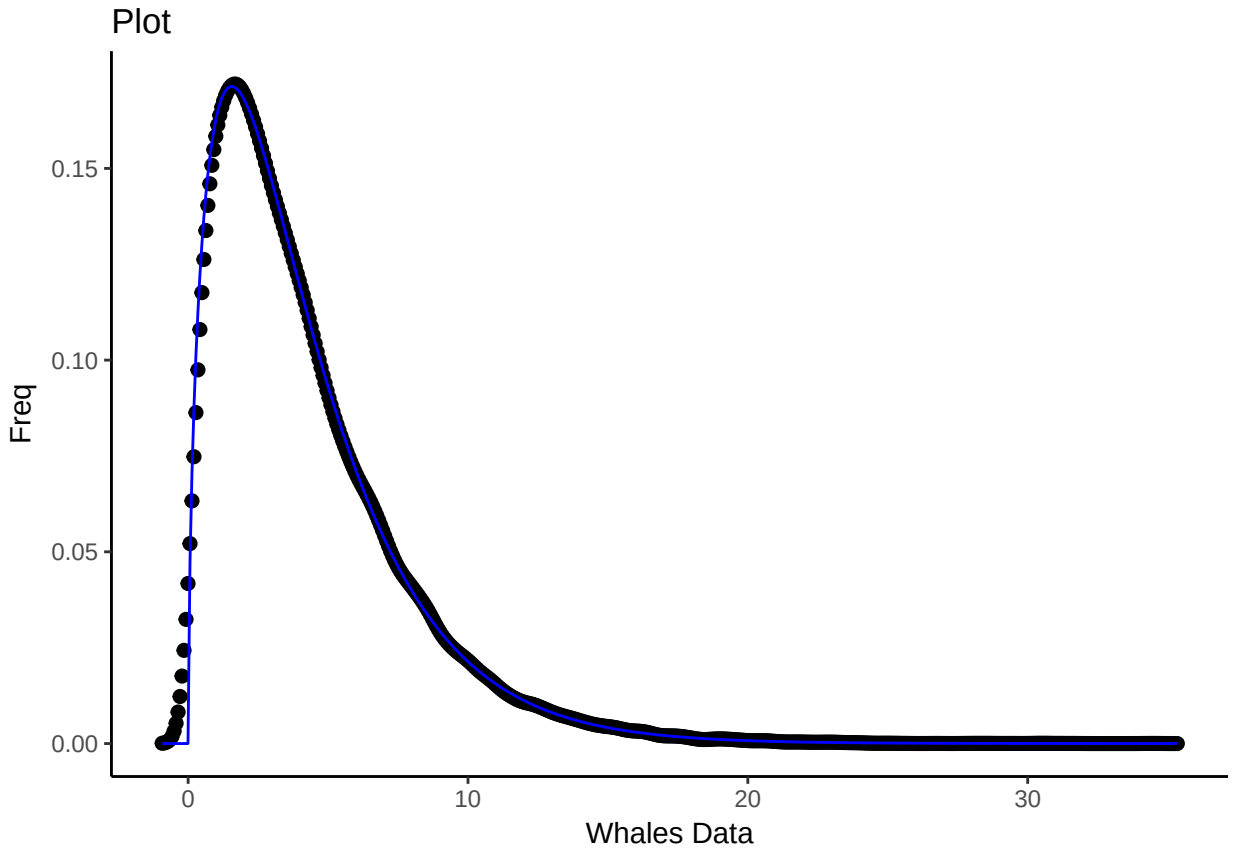
So, basically, I conclude that alpha and lambda from MLE are nearly twice as the ones from MoM. $alpha, lambda(MLE) = 2 * alpha, lambda(Mom)$ ### C

```
# Maximum Likehoold Way
# As the text book mentions "MLE are not given in closed form, obtaining their exact sampling distribut
# So I went over "https://pdfs.semanticscholar.org/306d/d2f94d4e2a4e460ba26d07aba05c6f0b587a.pdf" and "
# lambda = alpha / x_bar
# 1. alpha ~ ((3-s + sqrt((s-3)^2 + 24*s)) / 12*s) where s = log(x_bar) - mean(log(newdata)) -> within
# 2. alpha ~ (0.5 / s) -> (https://pdfs.semanticscholar.org/306d/d2f94d4e2a4e460ba26d07aba05c6f0b587a.p
# 3. alpha -> from the textbook
# 4. alpha -> from mle() function -> lambda: 2.63269 and alpha: 1.59541

# Textbook way
loglikehood <- function(x){
  answer <- 210*log(x) - 210*log(x_bar) + sum(log(x))-n*digamma(x)
  return(answer)
}

#z <- nleqslv(x, loglikehood)[[1]]


# MLE function using package of stats4
MLEf <- function(alpha, lambda) {
     R = dgamma(whales$Data, alpha, lambda)
     -sum(log(R))
 }
```

```r
mle1 = mle(MLEf, start = list(alpha = 1, lambda = 1), method = "L-BFGS-B", lower = c(-Inf, 0), upper = c

alph_mle1 = mle1@coef[[1]]
lambd_mle1 = mle1@coef[[2]]


x2 <- rgamma(50000, shape=alph_mle1, scale=lambd_mle1)
densit2 <- density(x2)
datas2 <- data.frame(x_mle1 = densit2$x, y_mle1 = densit2$y)

# Plot density from Rgamma points with MLE2
ggplot(data = datas2, aes(x = x_mle1, y = y_mle1)) +
  geom_point(size = 1) +
  theme_classic() + xlab("Whales Data") + ylab("Freq") + ggtitle("Plot Density from random Gamma points
```



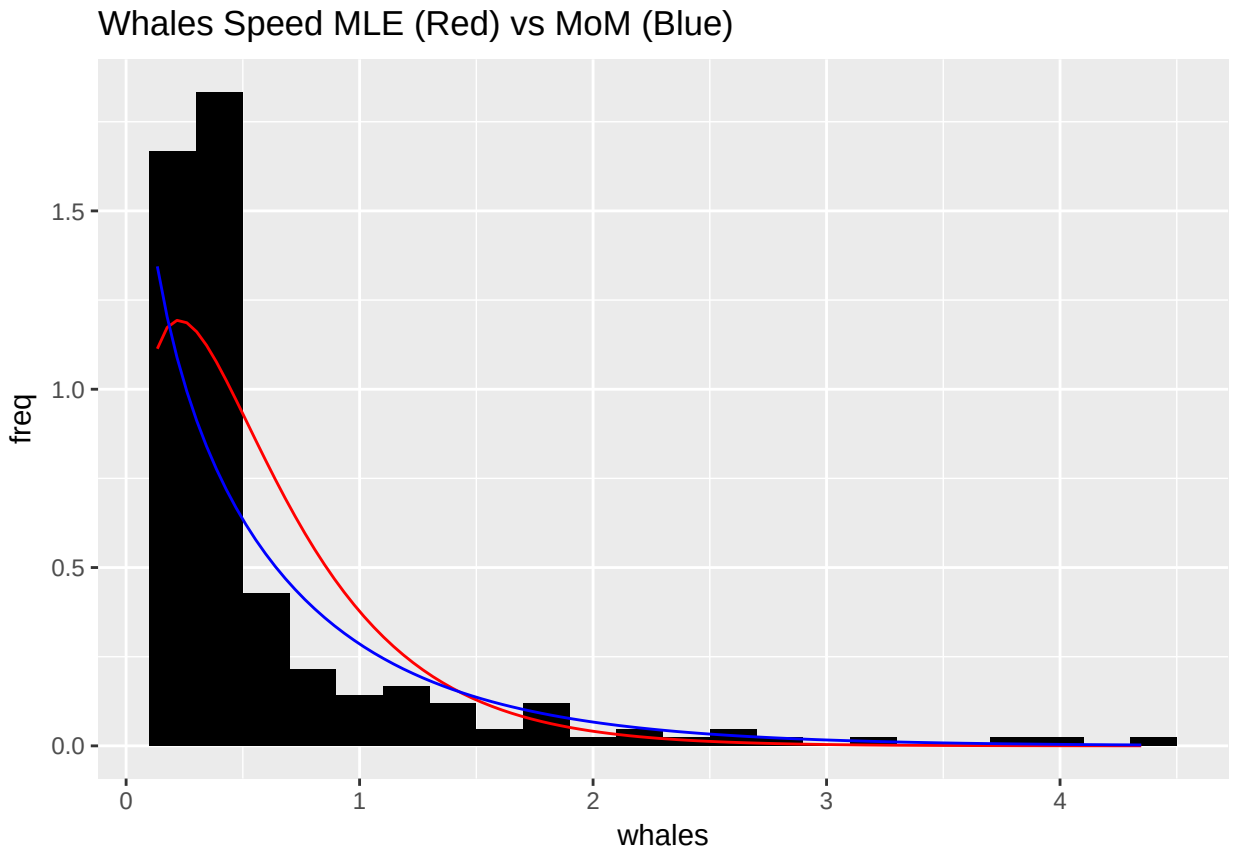

```r
#Fit Parameters - check Fit
fit.params2 <- fitdistr(x2, "gamma")

# Plot with density points - Fitparams into the MLE2 density
ggplot(data = datas2, aes(x = x_mle1,y = y_mle1)) +
  geom_point(size = 2) +
  geom_line(aes(x=datas2$x, y=dgamma(datas2$x, fit.params2$estimate["shape"], fit.params2$estimate["rat
  theme_classic() + xlab("Whales Data") + ylab("Freq") + ggtitle("Plot")
```




```
# The second way (Extra) - This might be wrong, although I got closed to alph_mle1 and lambd_mle1.
s = log(x_bar) - mean(log(newdata))
alph_mle2 = 0.5 / s
lambd_mle2 = alph_mle2 / mean(newdata)
```

D

```
hist1 <- ggplot(whales, aes(x = Data)) +
  geom_histogram(binwidth = 0.2, fill = "black", aes(y = ..density..), position = "identity", size = 1)
  xlab("whales") +ylab("freq") + ggtitle("Whales Speed MLE (Red) vs MoM (Blue)")+
  stat_function(fun = dgamma, args= c(shape=alph_mle1, scale=1/lambd_mle1), color = "red")+
  stat_function(fun = dgamma, args= c(shape=alph, scale=1/lambd), color = "blue")


hist1
```



E

```
lambdhats <- numeric(0)
alphhats <- numeric(0)
for (i in 1:1000){
  do <- rgamma(210, shape = alph, scale = lambd)
  xbar <- mean(do)
  s <- var(do)
  lambdhat <- (xbar / s)
  alphhat <- (xbar^2 / s)
  lambdhats <- c(lambdhats, lambdhat)
  alphhats <- c(alphhats, alphhat)
}
print("Mean for lambda hat - MoM")
```

```
## [1] "Mean for lambda hat - MoM"
```

```
mean(lambdhats)
```

```
## [1] 0.78377298
```

```
print("sdv for lambda hat - MoM")
```

```
## [1] "sdv for lambda hat - MoM"
```

```
sd(lambdhats)
```

```
## [1] 0.119863645
```

```r
print("Mean for alpha hat - MoM")
```

```
## [1] "Mean for alpha hat - MoM"
```

```r
mean(alphhats)
```

```
## [1] 0.816031257
```

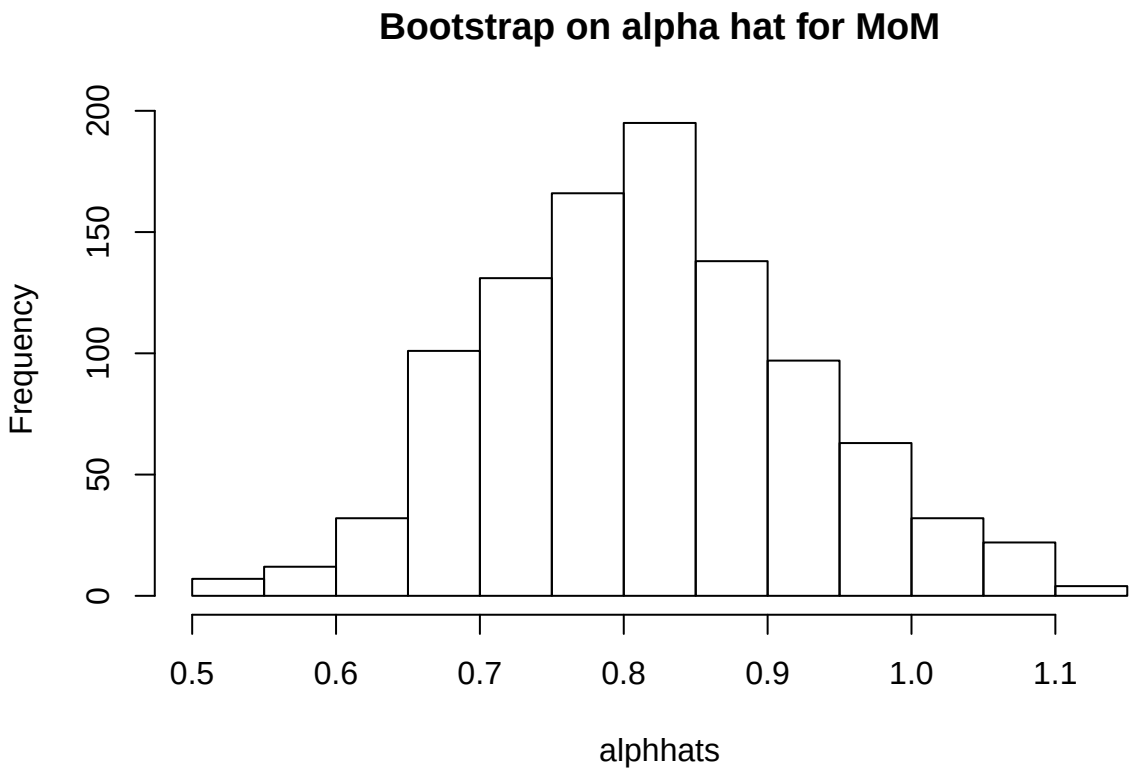
```r
print("sdv for lambda hat - MoM")
```

```
## [1] "sdv for lambda hat - MoM"
```

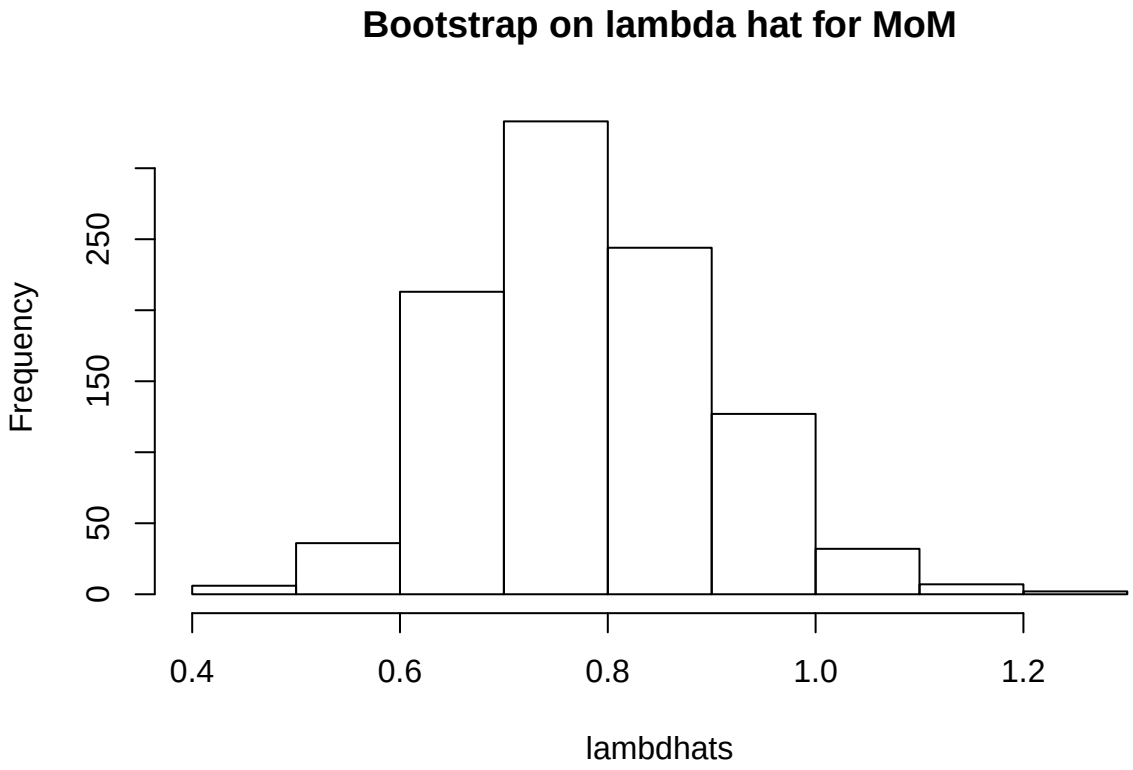
```r
sd(alphhats)
```

```
## [1] 0.11024179
```

```r
hist(alphhats, main = "Bootstrap on alpha hat for MoM")
```




```r
hist(lambdhats, main = "Bootstrap on lambda hat for MoM")
```




```
# Looks normal
```

F

```
lambdhats2 <- numeric(0)
alphhats2 <- numeric(0)


for(i in 1:1000){
  do2 <- rgamma(210, shape = alph_mle1, rate = lambd_mle1)

  LL2 <- function(alpha, lambda) {
     R = dgamma(do2, alpha, lambda)
     -sum(log(R))
  }
  mle2 = mle(LL2, start = list(alpha = alph_mle1, lambda=lambd_mle1), method =
               "L-BFGS-B", lower = c(-Inf, 0), upper = c(Inf, Inf))

  lambdhat2 <- mle2@coef[[2]]
  alphhat2 <- mle2@coef[[1]]
  lambdhats2 <- c(lambdhats2, lambdhat2)
  alphhats2 <- c(alphhats2, alphhat2)
}

print("Mean for lambda hat - MLE")
```

```
## [1] "Mean for lambda hat - MLE"
```

```r
mean(lambdhats2)
```

```
## [1] 2.65634167
```

```r
print("sdv for lambda hat - MLE")
```

```
## [1] "sdv for lambda hat - MLE"
```

```r
sd(lambdhats2)
```

```
## [1] 0.273366666
```

```r
print("Mean for alpha hat - MLE")
```

```
## [1] "Mean for alpha hat - MLE"
```

```r
mean(alphhats2)
```

```
## [1] 1.60839455
```

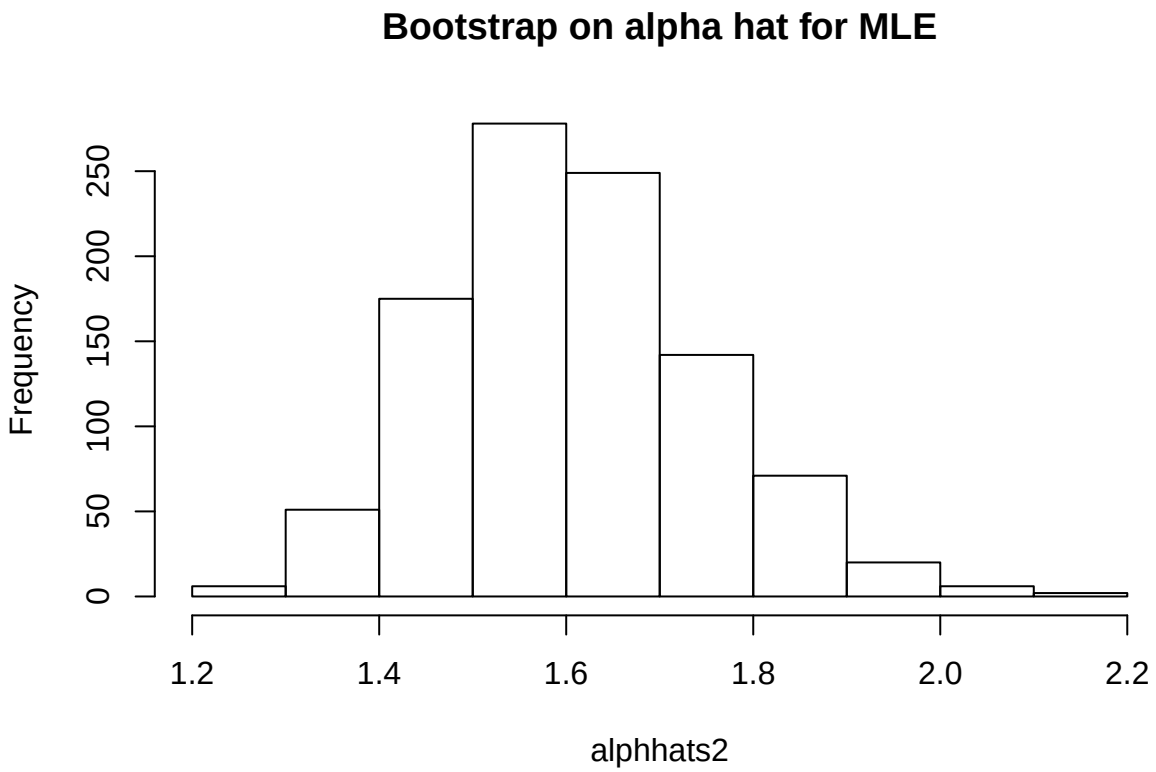
```r
print("sdv for lambda hat - MLE")
```

```
## [1] "sdv for lambda hat - MLE"
```

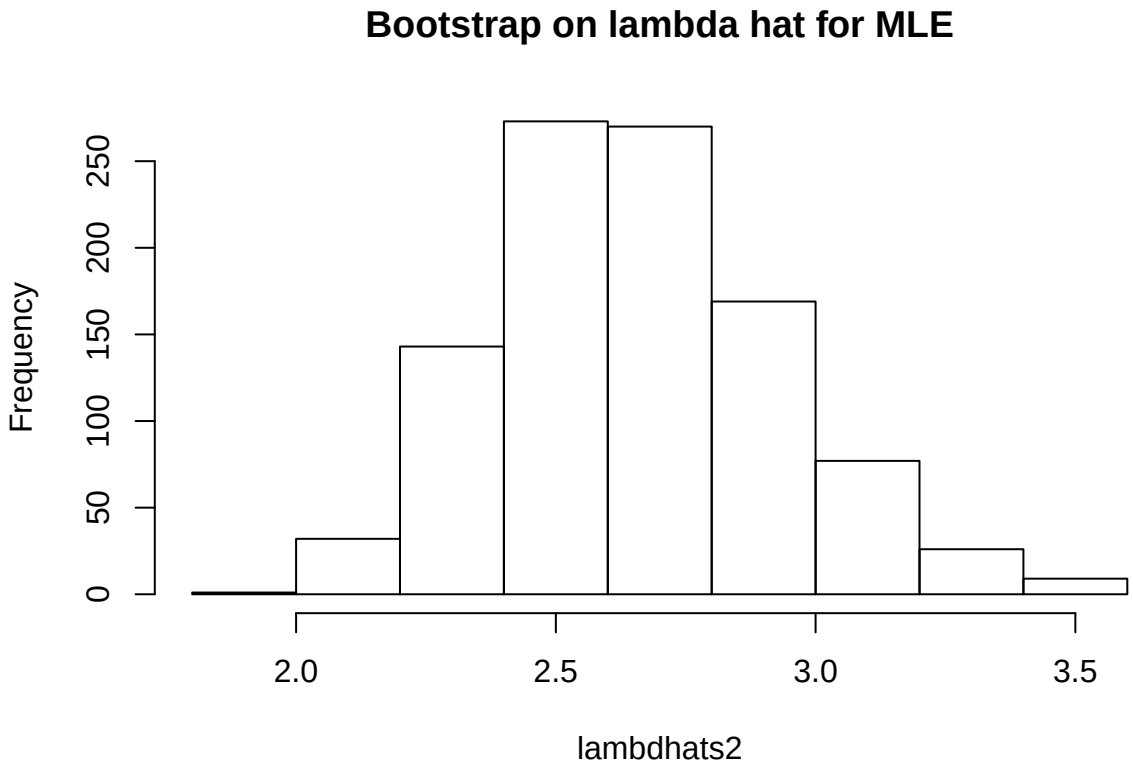
```r
sd(alphhats2)
```

```
## [1] 0.142807792
```

```r
hist(alphhats2, main = "Bootstrap on alpha hat for MLE")
```




```r
hist(lambdhats2, main = "Bootstrap on lambda hat for MLE")
```




```
# Looks normal
```

Since large degree of freedom, go for $Xp^2 = 1/2\ (zp + sqrt(2v-1))^2$. ### G

```
# Three methods for confidence interval for MLE: exact method, approximations based on the large sample
#Since Bootstrap uses a lot of data, by the central theorem, we say the distribution is likely to be a

# alpha
Lower <-function(t){
  return(mean(alphhats2) - (sd(alphhats2)/sqrt(1000))*t)
}
Upper <- function(t2){
  return(mean(alphhats2) + (sd(alphhats2)/sqrt(1000))*t2)
}
print("Alpha")
```

```
## [1] "Alpha"
```

```
# For 90%
print("Lower Boundary for 90%")
```

```
## [1] "Lower Boundary for 90%"
```

```
Lower(1.645)
```

```
## [1] 1.60096577
```

```
print("Upper Boundary for 90%")
```

```
## [1] "Upper Boundary for 90%"
```

```
Upper(1.645)
```

```
## [1] 1.61582334
```

```r
# For 95%
print("Lower Boundary for 95%")
```

```
## [1] "Lower Boundary for 95%"
```

```r
Lower(1.96)
```

```
## [1] 1.59954323
```

```r
print("Upper Boundary for 95%")
```

```
## [1] "Upper Boundary for 95%"
```

```r
Upper(1.96)
```

```
## [1] 1.61724587
```

```r
# For 99%
print("Lower Boundary for 99%")
```

```
## [1] "Lower Boundary for 99%"
```

```r
Lower(2.576)
```

```
## [1] 1.59676139
```

```r
print("Upper Boundary for 99%")
```

```
## [1] "Upper Boundary for 99%"
```

```r
Upper(2.576)
```

```
## [1] 1.62002771
```

```r
# Lambda
Boundary <- function(chi){
  return((1000*sd(lambdhats2)^2) / chi)
}

print("=====================================")
```

```
## [1] "====================================="
```

```r
print("=====================================")
```

```
## [1] "====================================="
```

```r
print("=====================================")
```

```
## [1] "====================================="
```

```r
print("Lambda")
```

```
## [1] "Lambda"
```

```r
# For 90%
print("Lower Boundary for 90%")
```

```
## [1] "Lower Boundary for 90%"
```

```r
Boundary(0.5 * (-1.645 + sqrt(2 * 1000 - 1) )^2) #Lower
```

```
## [1] 0.0805876737
```

```r
print("Upper Boundary for 90%")
```

```
## [1] "Upper Boundary for 90%"
```

```r
Boundary(0.5 * (1.645 + sqrt(2 * 1000 - 1) )^2) # Upper
```

```
## [1] 0.0695543996
```

```r
# For 95%
print("Lower Boundary for 95%")
```

```
## [1] "Lower Boundary for 95%"
```

```r
Boundary(0.5 * (-1.96 + sqrt(2 * 1000 - 1) )^2) #Lower
```

```
## [1] 0.0817796519
```

```r
print("Upper Boundary for 95%")
```

```
## [1] "Upper Boundary for 95%"
```

```r
Boundary(0.5 * (1.96 + sqrt(2 * 1000 - 1) )^2) # Upper
```

```
## [1] 0.0686186544
```

```r
# For 99%
print("Lower Boundary for 99%")
```

```
## [1] "Lower Boundary for 99%"
```

```r
Boundary(0.5 * (-2.575 + sqrt(2 * 1000 - 1) )^2) #Lower
```

```
## [1] 0.0841843662
```

```r
print("Upper Boundary for 99%")
```

```
## [1] "Upper Boundary for 99%"
```

```r
Boundary(0.5 * (2.575 + sqrt(2 * 1000 - 1) )^2) # Upper
```

```
## [1] 0.0668453275
```