# ASS x Cainz R Workshop 2021 - Day 2

Functions used:

- `uniroot()`
- `maxLik()`
- `lm()`
- `paste()`
- `with()`
- `summary()`
- `plot()`
- `sapply()`
- `read.csv()`

Packages:

- `maxLik` — `install.packages("maxLik")`

### Exercise 1 - Solving 1D equations example

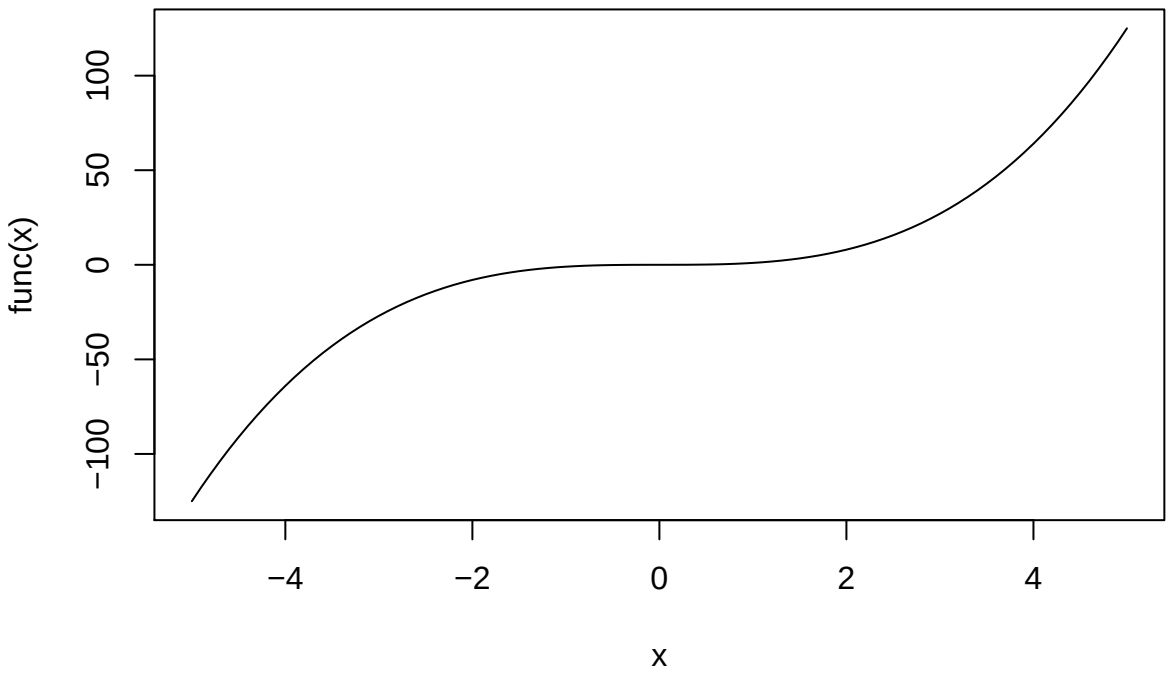
Often we need to use software ($R$) to solve equations of the form $f(x) = 0$ for $x$. This can be done using the `uniroot()` function, applications of this could be something as simple as finding the $x$-intercepts of a curve, to finding the yield to maturity a dividend paying bond or finding a maximum likelihood estimate. We will do each of these in turn:

1. Solve $x^3 = 0$:

```
func <- function(x) {x^3}
uniroot(func, lower = -10, upper = 10)$root
```

```
## [1] 0
```

```
curve(func, from = -5, to = 5)
```




2. Find the yield to maturity of a dividend paying bond maturing in 4 years, paying coupons annually with 4% interest p.a. and a face value of $100 bought at $95:

The yield to maturity will be $x$, where $x$ is given by:

$$95 = \frac{4}{(1+x)} + \frac{4}{(1+x)^2} + \frac{4}{(1+x)^3} + \frac{104}{(1+x)^4} \text{ or equivalently } 95 = 4 \times \frac{1-(1+x)^{-4}}{x} + \frac{100}{(1+x)^4}$$

```
func <- function(x) {4*((1-(1+x)^(-4))/(x))+(100/((1+x)^(4))) - 95}
#uniroot(func, lower = 0, upper = 1) # will not work due to divide by 0
uniroot(func, lower = 0.0000001, upper = 1)$root #instead enter a very low number for the lower bound
```

```
## [1] 0.05422504
```

3. Use the observed heights from `heights.csv` to find the an estimate of the mean through maximum likelihood estimation assuming that we know the data follows $\mathcal{N}(\mu, 10)$.

```
#Maybe use uniroot instead w/ derivation at end
library(maxLik)
#Importing data
heights <- read.csv('heights.csv')
variance <- 10

my.dnorm <- function(x, mu, var) {
  (1/sqrt(2*pi*var)) * exp((-0.5/var)*(x - mu)^2)
}
my.norm.loglik <- function(mu, var){
  loglik <- log(prod(my.dnorm(heights, mu, var)))
```

```
  return(loglik)
}

(mle <- paste("MLE of mu:", maxLik(my.norm.loglik, start = 170, var = variance)$estimate))
```

```
## [1] "MLE of mu: 175.060000006531"
```

```
(mean <- paste("Mean:", mean(heights[,1])))
```

```
## [1] "Mean: 175.06"
```

### Exercise 2 - CAPM example

```
CAPM <- read.csv("CAPM_BAC.csv")
```

Using the with() function create a data set which contains the following data points: 1. Excess returns of BAC over the Risk Free (BAC.Returns - RF) 2. Market premium (Mkt.RF)

```
dataset <- with(CAPM, data.frame("Excess.Rtn.BAC" = BAC.Returns - RF, "Mkt.Prem" = Mkt.RF))
```

Using lm(), run a linear regression of the excess BAC returns on the market premium.

```
CAPM.model <- lm(Excess.Rtn.BAC ~ Mkt.Prem, dataset)
summary(CAPM.model)
```

```
##
## Call:
## lm(formula = Excess.Rtn.BAC ~ Mkt.Prem, data = dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -22.2835  -0.8048  -0.0533   0.7213  28.6059
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.02943    0.03319  -0.887    0.375
## Mkt.Prem     1.71140    0.02722  62.864   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.185 on 4338 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.4767, Adjusted R-squared:  0.4766
## F-statistic:  3952 on 1 and 4338 DF,  p-value: < 2.2e-16
```

Recall back to Principles of Finance, the regression of a stocks excess returns on the excess returns of the market finds the Beta of the stock. Accordingly, through this simple linear regression we have found the Beta of Bank of America (BAC) across the give time horizon.

### Exercise 3 - Dealing with missing data (Imputation)

```
#Creating dataset
set.seed(1)
iris.missing <- iris
```

```r
#creating data with missing values
missing_matrix <- matrix(as.logical(rbinom(5*150, 1, 0.05)), nrow = 150)
iris.missing[missing_matrix] <- NA

#The idea is to replace missing values in the first 4 columns
#with the mean of that species
iris.cleaned <- iris.missing[!is.na(iris.missing$Species),]

#Group the dataset by unique values of species
#I will only do this for one species for the sake of explanability
species <- "setosa"
species.data <- iris.cleaned[iris.cleaned$Species == species,]
for (i in 1:(ncol(species.data)-1)) {
  #select the column we want to manipulate
  column <- species.data[,i]
  column[which(is.na(column))] = mean(column, na.rm = T)
  #replace the column in the dataset with the manipulated column
  species.data[,i] <- column
}
species.data
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1      5.100000         3.5     1.400000   0.2431818  setosa
## 2      4.900000         3.0     1.400000   0.2000000  setosa
## 3      4.700000         3.2     1.300000   0.2000000  setosa
## 4      4.600000         3.1     1.500000   0.2000000  setosa
## 7      4.600000         3.4     1.400000   0.3000000  setosa
## 8      5.000000         3.4     1.500000   0.2431818  setosa
## 10     4.900000         3.1     1.500000   0.1000000  setosa
## 11     5.400000         3.7     1.500000   0.2000000  setosa
## 12     4.800000         3.4     1.600000   0.2000000  setosa
## 13     4.800000         3.0     1.400000   0.1000000  setosa
## 14     4.300000         3.0     1.100000   0.1000000  setosa
## 15     5.800000         4.0     1.200000   0.2000000  setosa
## 16     5.700000         4.4     1.500000   0.4000000  setosa
## 17     5.400000         3.9     1.300000   0.4000000  setosa
## 18     5.008696         3.5     1.400000   0.3000000  setosa
## 19     5.700000         3.8     1.443902   0.3000000  setosa
## 20     5.100000         3.8     1.500000   0.3000000  setosa
## 21     5.400000         3.4     1.700000   0.2000000  setosa
## 22     5.100000         3.7     1.500000   0.2431818  setosa
## 23     4.600000         3.6     1.000000   0.2000000  setosa
## 24     5.100000         3.3     1.443902   0.5000000  setosa
## 25     4.800000         3.4     1.900000   0.2000000  setosa
## 26     5.000000         3.0     1.443902   0.2000000  setosa
## 27     5.000000         3.4     1.600000   0.4000000  setosa
## 28     5.200000         3.5     1.500000   0.2000000  setosa
## 29     5.200000         3.4     1.400000   0.2000000  setosa
## 30     4.700000         3.2     1.600000   0.2000000  setosa
## 31     4.800000         3.1     1.443902   0.2000000  setosa
## 32     5.400000         3.4     1.500000   0.4000000  setosa
## 33     5.200000         4.1     1.500000   0.1000000  setosa
## 34     5.500000         4.2     1.400000   0.2000000  setosa
## 35     4.900000         3.1     1.500000   0.2000000  setosa
```

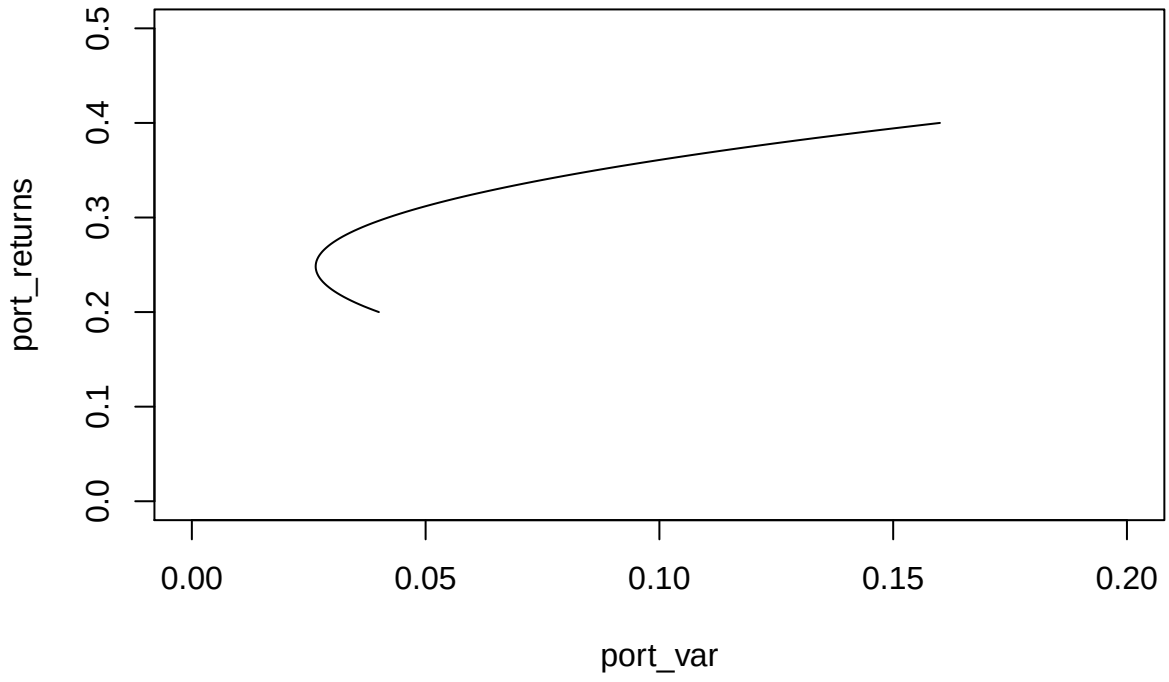
```
## 36     5.000000         3.2     1.200000   0.2000000  setosa
## 37     5.500000         3.5     1.300000   0.2000000  setosa
## 38     4.900000         3.6     1.400000   0.1000000  setosa
## 39     4.400000         3.0     1.300000   0.2000000  setosa
## 40     5.100000         3.4     1.443902   0.2000000  setosa
## 41     5.000000         3.5     1.443902   0.3000000  setosa
## 42     4.500000         2.3     1.300000   0.3000000  setosa
## 43     4.400000         3.2     1.300000   0.2000000  setosa
## 44     5.000000         3.5     1.600000   0.6000000  setosa
## 45     5.100000         3.8     1.900000   0.4000000  setosa
## 46     4.800000         3.0     1.400000   0.3000000  setosa
## 47     5.100000         3.8     1.600000   0.2000000  setosa
## 48     4.600000         3.2     1.400000   0.2000000  setosa
## 49     5.300000         3.7     1.500000   0.2000000  setosa
## 50     5.000000         3.3     1.400000   0.2000000  setosa
```

```r
#we can put another for loop over the whole code block
#to repeat this for each species
```

**Exercise 4 - Portfolio optimisation example**

1. Create a plot of the minimum variance frontier of portfolios that can be formed from security $A$ with $E[r_A] = 0.2, \sigma_A^2 = 0.04$ and security $B$ with $E[r_B] = 0.4, \sigma_B^2 = 0.16$.and $Cov(A, B) = -0.016$

```r
exp_returns <- c(0.2, 0.4)
vars_returns <- c(0.04, 0.16)
covar <- -0.016

get_portfolio_returns <- function(w1, exp_returns) {
  w2 <- 1 - w1
  port_returns <- w1*exp_returns[1] + w2*exp_returns[2]
  return(port_returns)
}
get_portfolio_variances <- function(w1, vars_returns, covar) {
  w2 <- 1 - w1
  cov <- covar
  port_variance <- w1^2*vars_returns[1] + 2*w1*w2*covar + w2^2*vars_returns[2]
  return(port_variance)
}
w1 <- seq(0,1,0.01)
port_returns <- sapply(w1, get_portfolio_returns, exp_returns = exp_returns)
port_var <- sapply(w1, get_portfolio_variances, vars_returns = vars_returns, covar = covar)
plot(port_var, port_returns, type = 'l', xlim = c(0,0.2), ylim=c(0, 0.5))
```




**Exercise 5 - Data fitting example**

Packages:

- fitdistrplus
- maxLik
- stats4

Main Functions:

- hist()
- fitdist()
- maxLik()
- mle()
- dlnorm()

Exercise Outcomes:

1. Be able to conduct simple data fitting using the Method of Maximum likelihood Estimation (MLE)
2. Be able to plot a histogram with the probability density function superimposed
3. Develop critical thinking ability comparing different packages and associated functions

Assumed Knowledge:

1. the Method of Moments Estimation
2. the Method of Maximum Likelihood Estimation
3. Basic R knowledge - write a function (will be taught in the 1st workshop)

```r
# If you don't already have these installed
# install.packages("fitdistrplus")
# install.packages("maxLik")
# install.packages("stats4")
library(fitdistrplus)
```

```
## Warning: package 'fitdistrplus' was built under R version 3.6.3
```

```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```r
library(maxLik)
library(stats4)
```

```r
#Initial data exploration
getwd()
```

```
## [1] "C:/Users/antho/Desktop/CAINZ/Documents/2021/Sem 1/R Workshop"
```

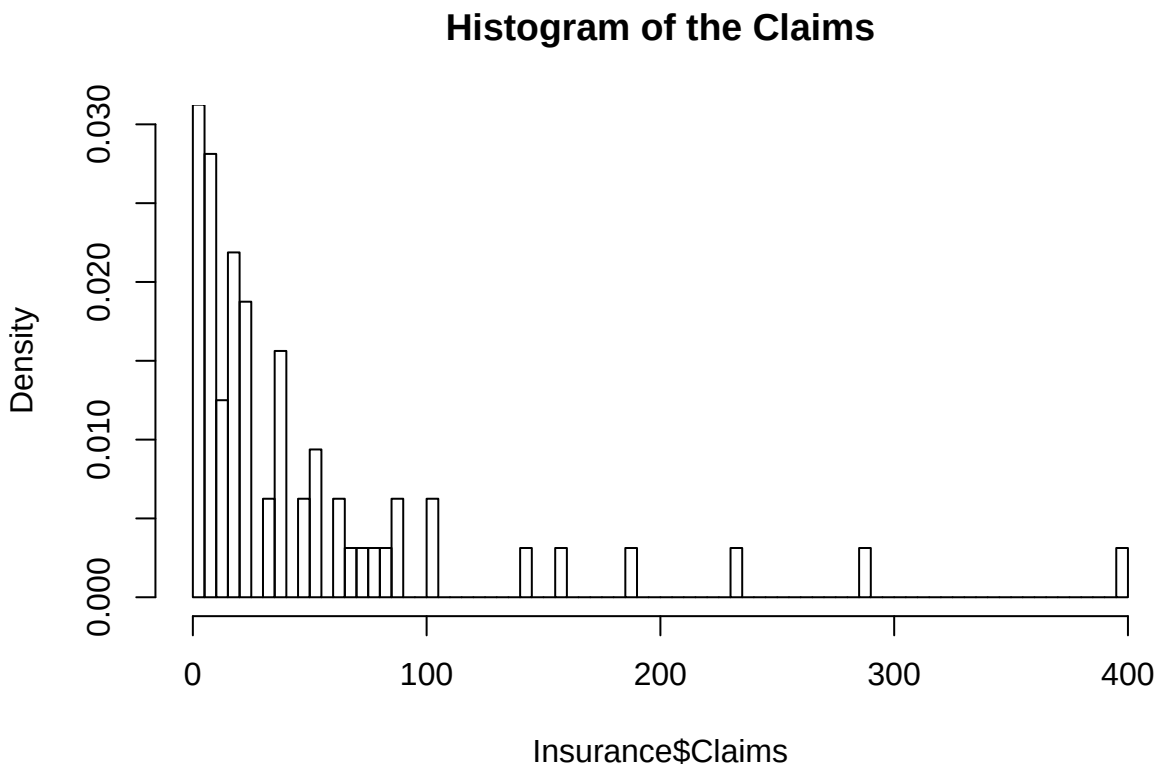
```r
Insurance <- read.csv("insurance.csv")

str(Insurance)
```

```
## 'data.frame':    64 obs. of  5 variables:
##  $ District: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Group   : Factor w/ 4 levels "<1l",">2l","1-1.5l",..: 1 1 1 1 3 3 3 3 4 4 ...
##  $ Age     : Factor w/ 4 levels "<25",">35","25-29",..: 1 3 4 2 1 3 4 2 1 3 ...
##  $ Holders : int  197 264 246 1680 284 536 696 3582 133 286 ...
##  $ Claims  : int  38 35 20 156 63 84 89 400 19 52 ...
```

```r
summary(Insurance$Claims)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.50   22.00   49.23   55.50  400.00
```

```r
hist(Insurance$Claims, main= "Histogram of the Claims", breaks=100, freq = FALSE, xlim = c(0,400), ylim
```




```
# Comment: Based on the histogram, we take a wild guess that the LOSS data follows a lognormal distribu
# Issue: zero claim has high density

non.zero.Insurance <- Insurance[-which((Insurance$Claims == 0)),]
```

```
#Fit a statistical distribution to the claims data using different packages

#fitdistrplus - fitdist function

MLE1 <- fitdist(non.zero.Insurance$Claims,"lnorm",method="mle")
(estimates.mle <- summary(MLE1)$estimate)
```

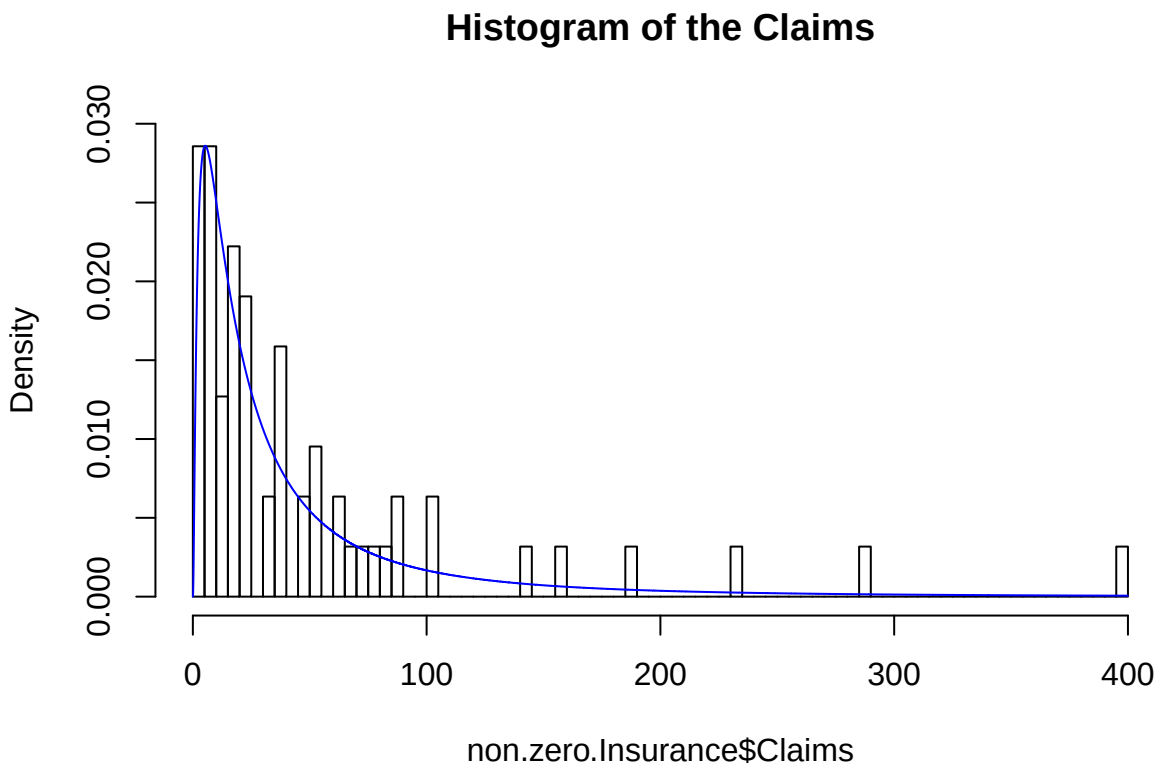
```
##  meanlog    sdlog
## 3.185221 1.231108
```

```
MLEmm <- fitdist(non.zero.Insurance$Claims,"lnorm",method="mme")
(estimates.mme <- summary(MLEmm)$estimate)
```

```
##  meanlog    sdlog
## 3.361582 1.049531
```

```
# the estimates from the method of moments are used as the initial values in maxLik() and mle()
#Plotting - histogram

hist(non.zero.Insurance$Claims, main= "Histogram of the Claims", breaks = 100,
     freq = FALSE, xlim = c(0,400), ylim = c(0,0.03))
x_axis <- seq(0,400, 0.01)
y_lnorm <- dlnorm(x_axis, mean = estimates.mle[1], sd = estimates.mle[2])
```

```
lines(x_axis,y_lnorm, col="blue",lwd =1)
legend(250,0.2, legend="dlnorm", col="blue",lty=1, cex=0.8)
```




```
#maxLik

logLikFun1 =function(param) {
  mu=param[1]
  sigma=param[2]
  sum(dlnorm(non.zero.Insurance$Claims, mean = mu, sd = sigma, log = TRUE))
  }
MLE2=maxLik(logLik = logLikFun1,
            start = c(mu = estimates.mme[1], sigma = estimates.mme[2]))

summary(MLE2)
```

```
## --------------------------------------------
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 4 iterations
## Return code 8: successive function values within relative tolerance limit (reltol)
## Log-Likelihood: -303.1606
## 2  free parameters
## Estimates:
##             Estimate Std. error t value Pr(> t)
## mu.meanlog    3.1852     0.1550   20.55  <2e-16 ***
## sigma.sdlog   1.2311     0.1097   11.23  <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ---------------------------------------------
```

```
#stats4

logLikFun2 = function(mu, sigma){
  sum(-dlnorm(non.zero.Insurance$Claims, meanlog = mu, sdlog = sigma, log = TRUE))
   }
 MLE3 = mle(minuslog=logLikFun2,
            start = list(mu = estimates.mme[[1]], sigma = estimates.mme[[2]]), method="L-BFGS-B")
summary(MLE3)
```

```
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = logLikFun2, start = list(mu = estimates.mme[[1]],
##     sigma = estimates.mme[[2]]), method = "L-BFGS-B")
##
## Coefficients:
##       Estimate Std. Error
## mu    3.185221  0.1551050
## sigma 1.231108  0.1096756
##
## -2 log L: 606.3212
```

After exercise question: As illustrated in the above exercise using three different packages and associated functions, it is now obvious to you that the maximum likelihood estimate is unique if it does exist. Thus, these three methods achieve mathematically-equivalent results. Which method would you prefer and why?