SDS_HW03_G40

Trina Sahoo(1901254) and Debodeep Banerjee(1901253)

```
knitr::opts_chunk$set(echo = TRUE)
# Import important packages
library(heavy)
library(Gmedian)
```

```
## Warning: package 'Gmedian' was built under R version 3.6.2
```

```
library(mnormt)
library(rgl)
```

```
## Warning: package 'rgl' was built under R version 3.6.2
```

```
library(rglwidget)
```

```
## Warning: package 'rglwidget' was built under R version 3.6.2
```

```
library(ModelMetrics)
```

```
## Warning: package 'ModelMetrics' was built under R version 3.6.2
```

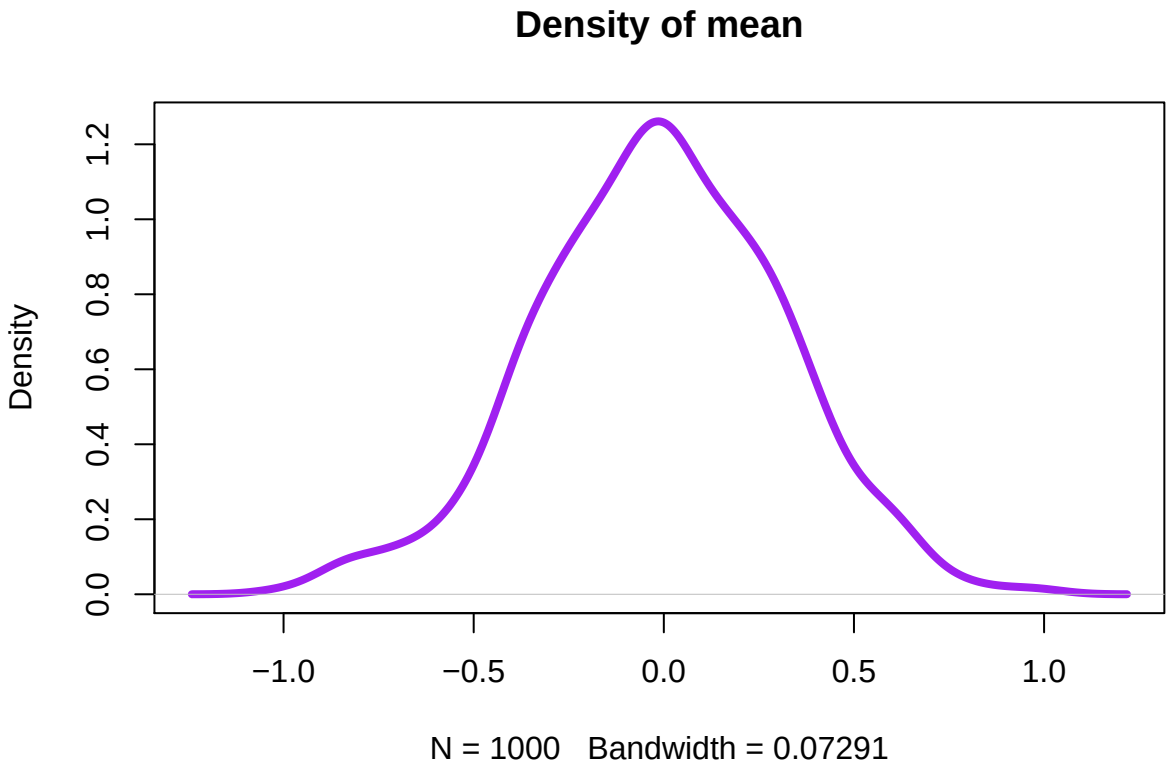
```
library(corrplot)
library(mvtnorm)
```

Question 1

1.1 Light tailed: Normal distribution

Random sample has been generated from normal distribution, MSE of mean and MSE of MOM has been obtained.

```
knitr::opts_chunk$set(echo = TRUE)

#Generating Random Sample
pop=rnorm(10000)
#Bootstrapping the sample
rep=100
results=vector()
M<-1000
for(i in 1:M){
  samp=sample(pop,10, replace=T)
  results[i]=mean(samp)
}
#Plot of density of means
plot(density(results), col="purple", lwd=4, main="Density of mean")
```

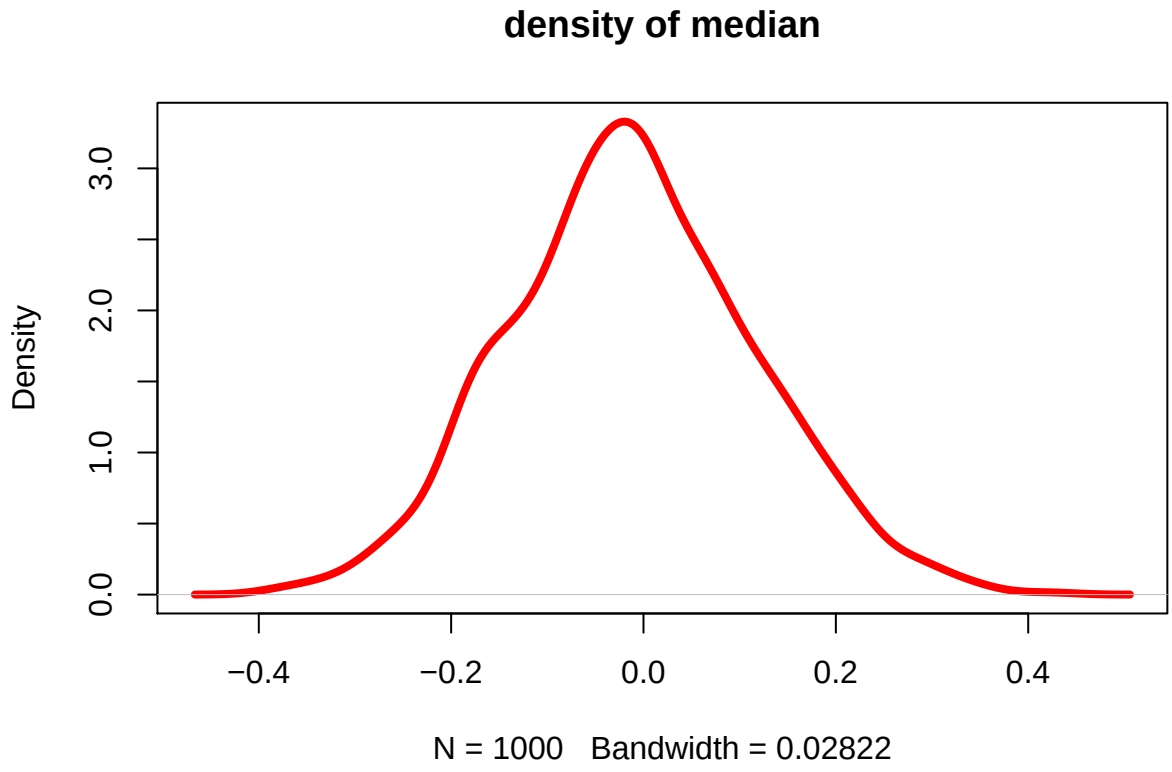



```
#Function to calculate the MSE

mse_uni=function(var,bias){
  return(var+bias^2)
}

means=mean(results)
mu.bias=means-0
var=var(results)
mu.bias
```

```
## [1] -0.01203221
```

```
mu.mse=mse_uni(var,mu.bias)

#The theoritical optimal block number k

thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}


# Sampling distribution of the MSE

k.exp=thres(0.05)
med=vector()
```

```r
for(i in 1:1000){
  samp=sample(pop,100,replace=T)
  x <- seq_along(samp)
  d1 <- split(samp, (x/(100/k.exp)))
  df=data.frame(d1)
  means=colMeans(df[sapply(df, is.numeric)])
  med[i]=median(means)
}

#Plot the density of medians
plot(density(med),col="red",lwd=4,main = "density of median")
```




```r
#Function to obtain the difference between the mean mse and median mse.

means=vector()
med=vector()
obj_uni=function(k){
  for(i in 1:1000){
    samp<-sample(pop,100, replace=T)
    x <- seq_along(samp)
    d1 <- split(samp, (x/(100/k)))
    df=data.frame(d1)
    means=colMeans(df[sapply(df, is.numeric)])
    med[i]=median(means)
  }
  med.bias=mean(med)-0
```

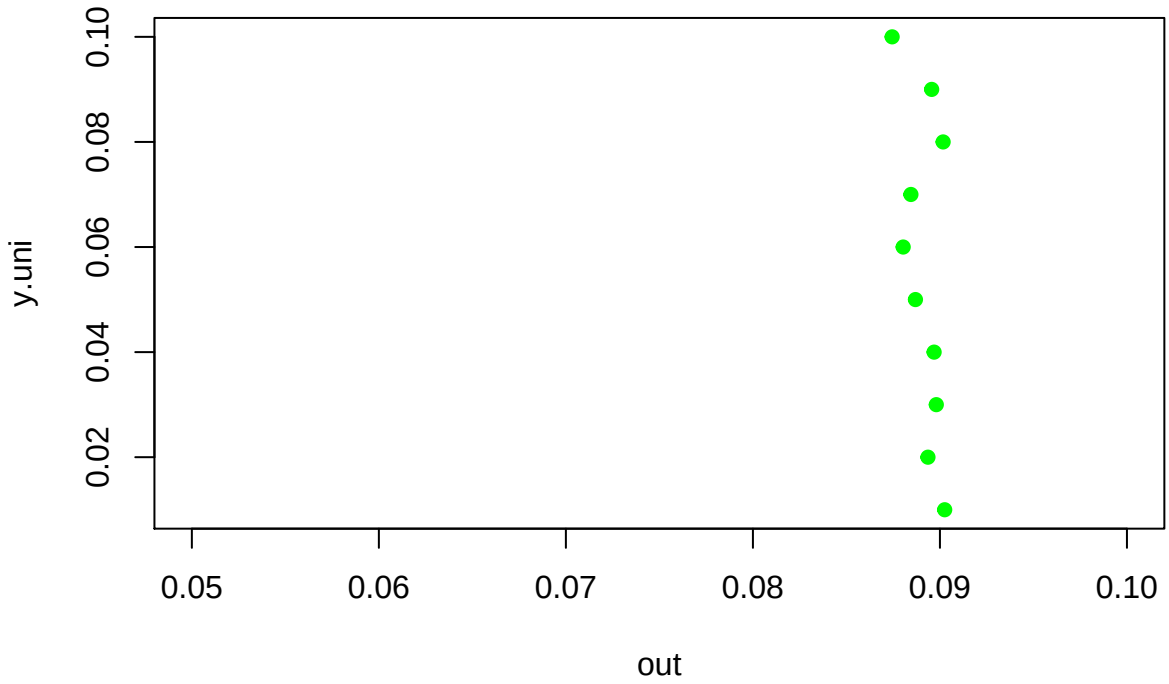
```
  med.var=var(med)
  med.mse=mse_uni(med.var,med.bias)
  return(mu.mse-med.mse)
}

#For different values
alpha=alpha=seq(0.01,0.1,0.01)
out=vector()
for (i in 1:length(alpha)){
  out[i]=obj_uni(thres(alpha[i]))
}
out
```

```
##  [1] 0.09025558 0.08936013 0.08980668 0.08968872 0.08869066 0.08803510
##  [7] 0.08844937 0.09017156 0.08955874 0.08744520
```

```
#Taking differnt values of alpha to plot the graph.
y.uni=vector()
for(i in 1:length(alpha)){
  y.uni[i]=alpha[i]
}
y.uni
```

```
##  [1] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10
```

```
#PLot to obtain the difference of mean mse and median mse against differnt values of alpha.
plot(y.uni~out, col="green", cex= 0.5, lwd=4,xlim=c(0.05,0.1))
```

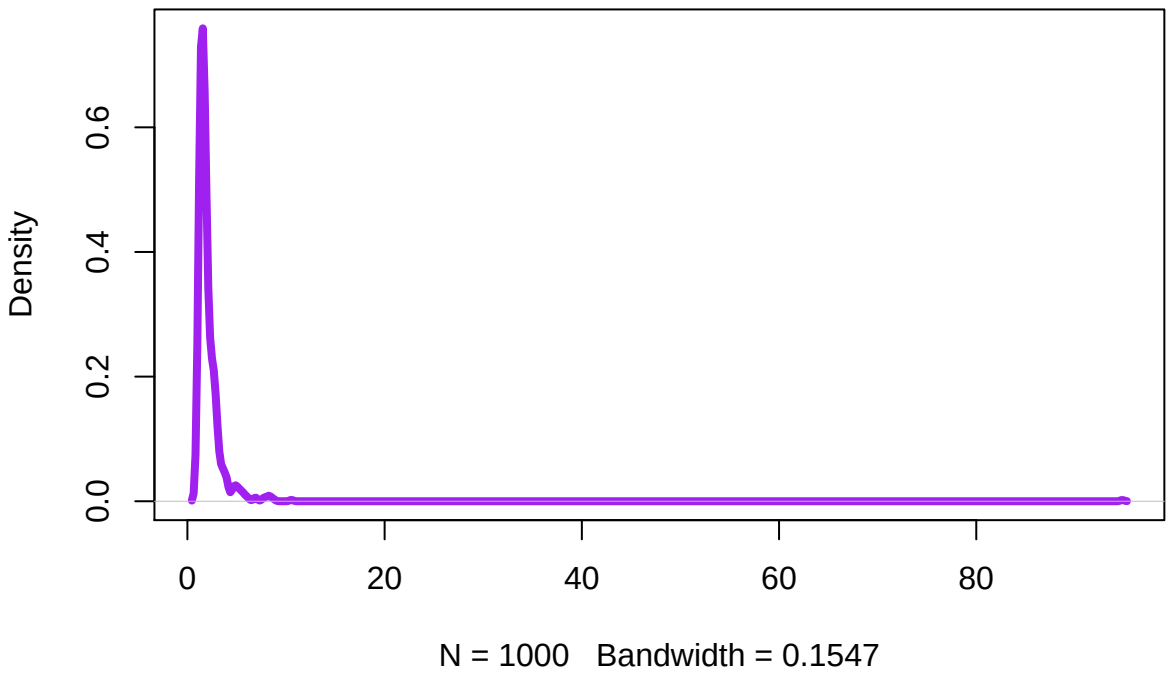


Interpretation

We have taken a normal distrubution which is a light tailed distribution, calculated the MSE of mean and the MSE of MoM. Then, we have selected various number of blocks over a sequencce of alpha. From the output and the plot we have observed that the MSE of the MoM is not always smaller than the the MSE of mean. Thus, we can claim that MOM is more robust statistic.

1.2. Heavy tailed: Pareto distribution

```
knitr::opts_chunk$set(echo = TRUE)

#Function to generate random number from Pareto distribution

rpar<-function(n, x, a){
  v<-runif(n)
  x<-x/v^(1/a)
  if (a>1)
    result1<-a*x/(a-1)
  result1
  if(a>2)
    result2<-x*x*a/((a-1)^2*(a-2))
  result2
}

pop=rpar(10000, 1, 3)
#Bootstrapping the sample
rep=100
results=vector()
M<-1000
for(i in 1:M){
  samp=sample(pop,10, replace=T)
  results[i]=mean(samp)
}
#Plot of density of means
plot(density(results), col="purple", lwd=4, main="Density of mean")
```

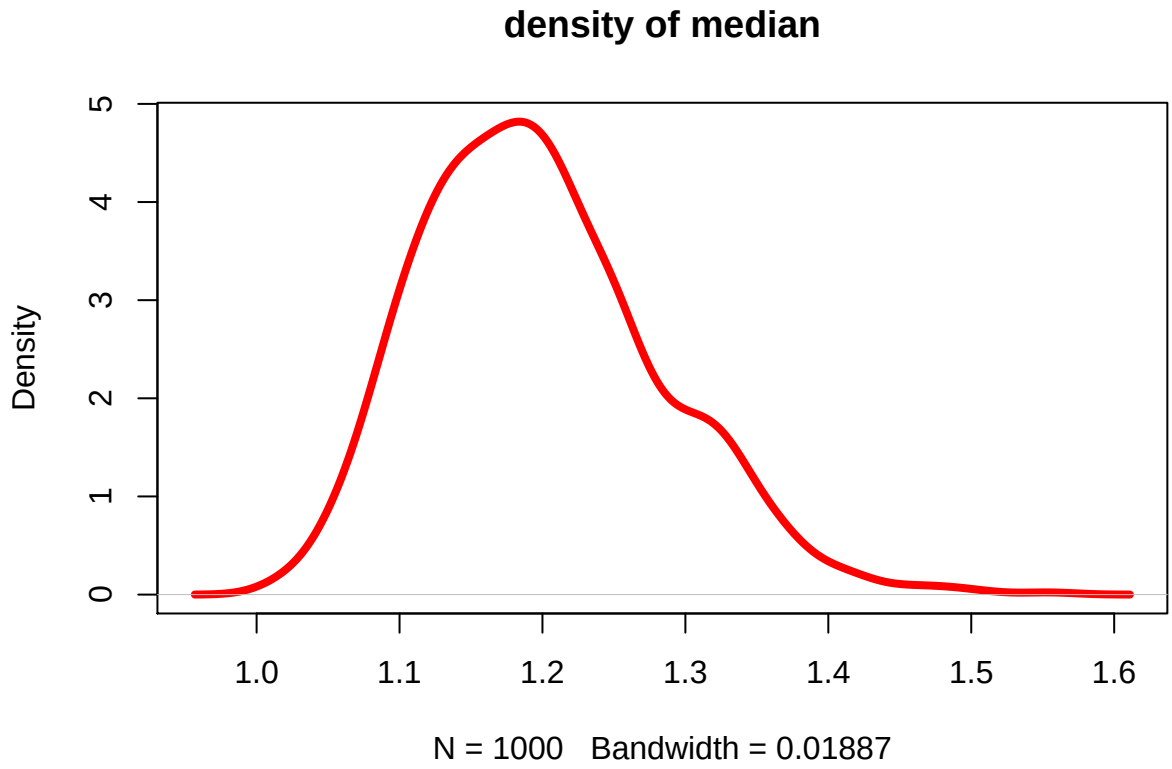



```
#Function to calculate the MSE

mse_uni=function(var,bias){
  return(var+bias^2)
}

means=mean(results)
mu.bias=means-0
var=var(results)
mu.bias
```

```
## [1] 2.160558
```

```
mu.mse=mse_uni(var,mu.bias)

#The theoritical optimal block number k

thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}


# Sampling distribution of the MSE

k.exp=thres(0.05)
med=vector()
```

```r
for(i in 1:1000){
  samp=sample(pop,100,replace=T)
  x <- seq_along(samp)
  d1 <- split(samp, (x/(100/k.exp)))
  df=data.frame(d1)
  means=colMeans(df[sapply(df, is.numeric)])
  med[i]=median(means)
}

#Plot the density of medians
plot(density(med),col="red",lwd=4,main = "density of median")
```




```r
#Function to obtain the difference between the mean mse and median mse.
means=vector()
med=vector()
obj_uni=function(k){
  for(i in 1:1000){
    samp<-sample(pop,100, replace=T)
    x <- seq_along(samp)
    d1 <- split(samp, (x/(100/k)))
    df=data.frame(d1)
    means=colMeans(df[sapply(df, is.numeric)])
    med[i]=median(means)
  }
  med.bias=mean(med)-0
  med.var=var(med)
```

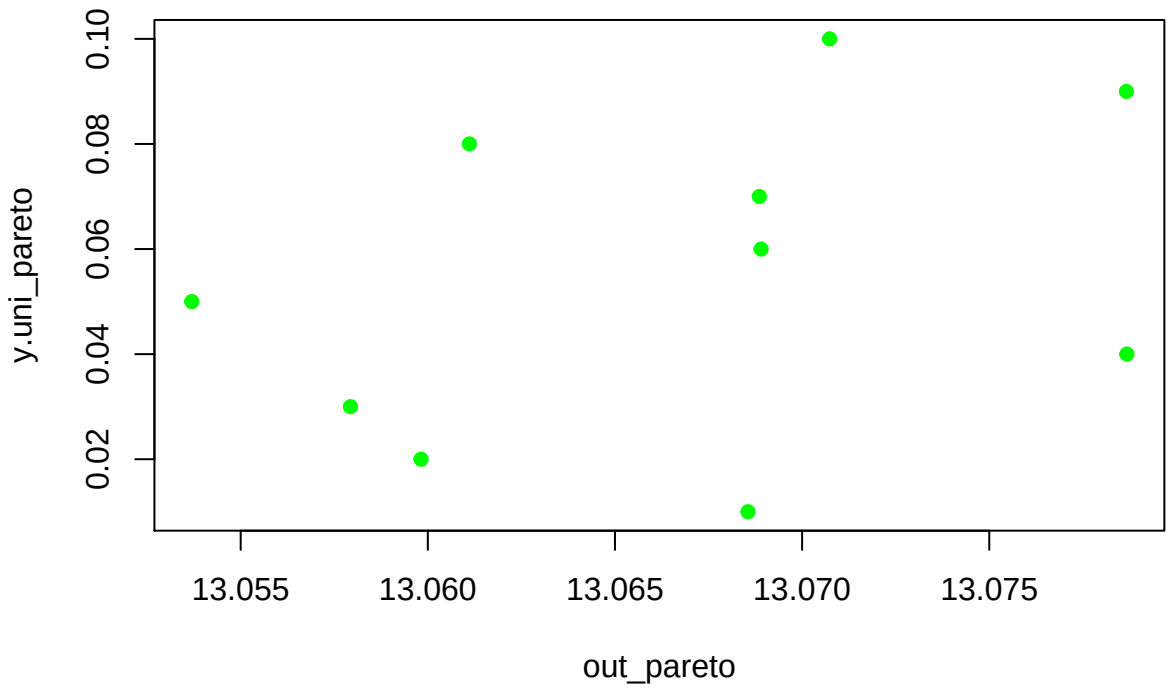
```r
  med.mse=mse_uni(med.var,med.bias)
  return(mu.mse-med.mse)
}

#For different values
alpha=seq(0.01,0.1,0.01)
out_pareto=vector()
for (i in 1:length(alpha)){
  out_pareto[i]=obj_uni(thres(alpha[i]))
}
out_pareto
```

```
##  [1] 13.06856 13.05982 13.05793 13.07868 13.05370 13.06890 13.06886
##  [8] 13.06111 13.07867 13.07074
```

```r
#Taking differnt values of alpha to plot the graph.
y.uni_pareto=vector()
for(i in 1:length(alpha)){
  y.uni_pareto[i]=alpha[i]
}
y.uni_pareto
```

```
##  [1] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10
```

```r
#PLot to obtain the difference of mean mse and median mse against differnt values of alpha.
plot(y.uni_pareto~out_pareto, col="green", cex= 0.5, lwd=4)
```

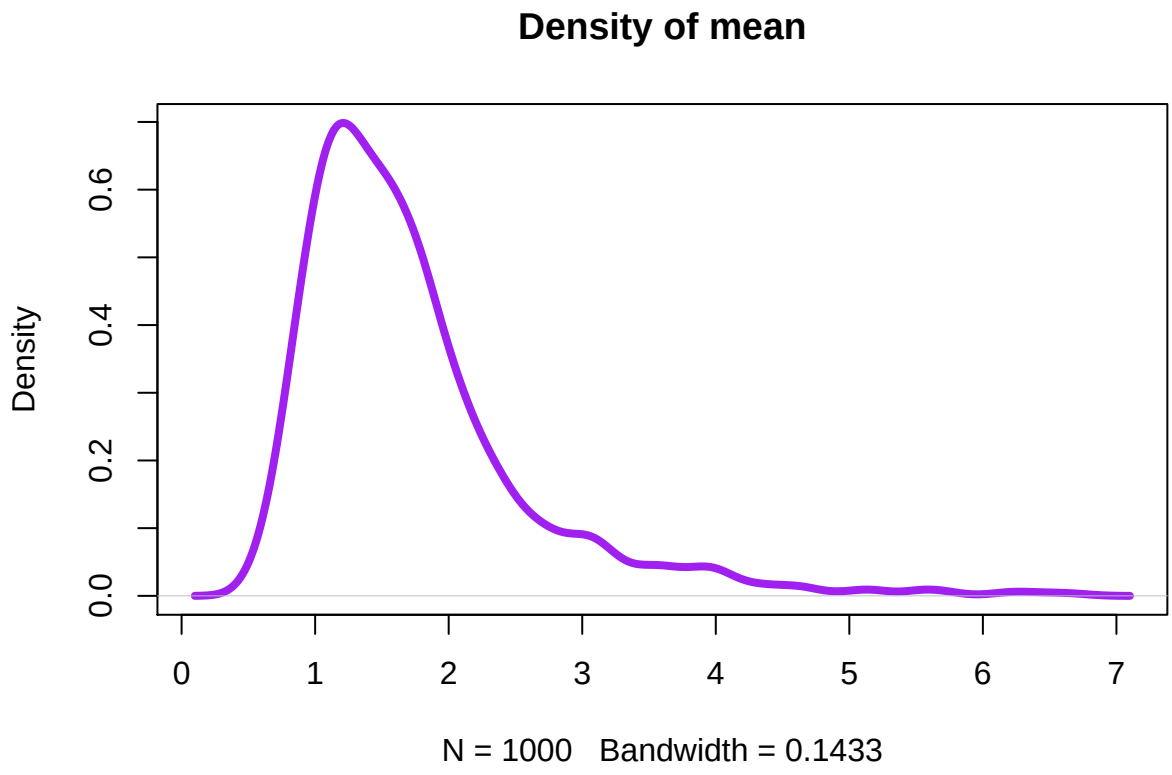


Interpretation

We have taken a pareto distribution which are heavy tailed distribution, calculated the MSE of mean and the MSE of MoM. Then, we have selected various number of blocks over a sequencce of alpha. From the output and the plot we have observed that the MSE of the MoM is not always smaller than the the MSE of mean. Thus, we can claim that MOM is more robust statistics.

1.3 Heavu tailed: Lognormal distribution

```
knitr::opts_chunk$set(echo = TRUE)

# Generate random number from Log-normal distribution
pop=rlnorm(1000, 0, 1)

#Bootstrapping the sample
rep=100
results=vector()
M<-1000
for(i in 1:M){
  samp=sample(pop,10, replace=T)
  results[i]=mean(samp)
}
#Plot of density of means
plot(density(results), col="purple", lwd=4, main="Density of mean")
```

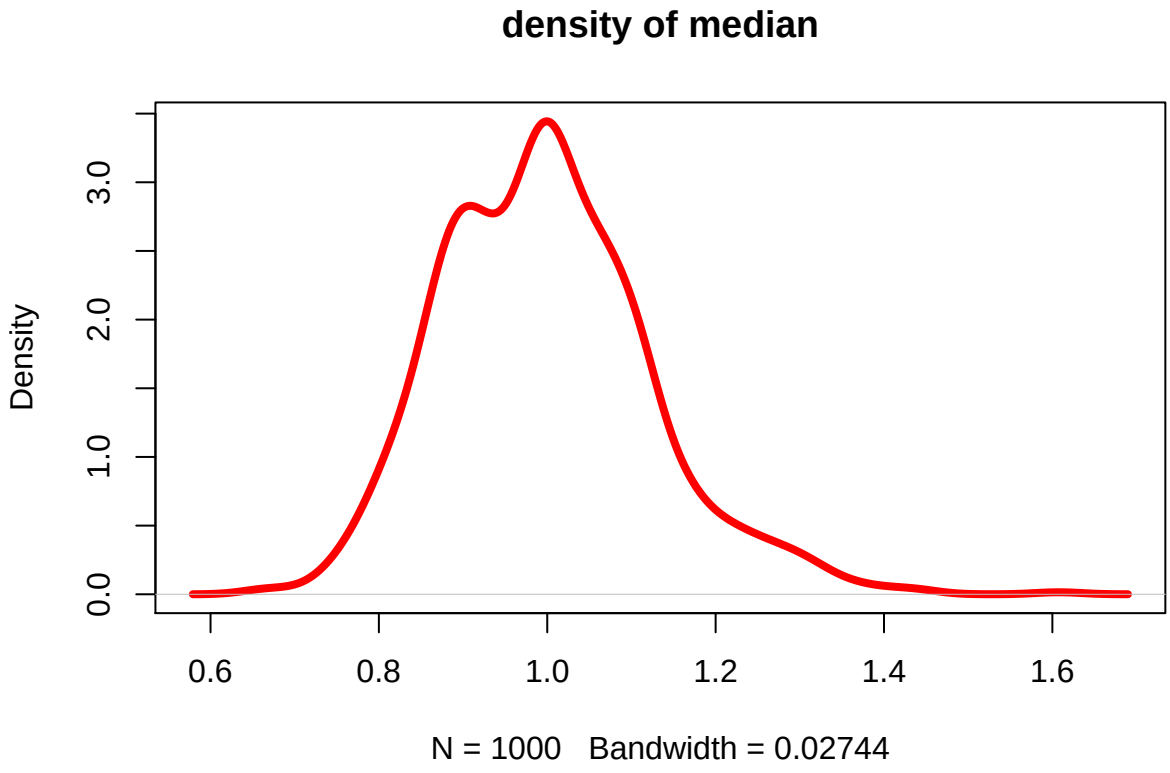



```
#Function to calculate the MSE
```

```r
mse_uni=function(var,bias){
  return(var+bias^2)
}

means=mean(results)
mu.bias=means-0
var=var(results)
mu.bias
```

```
## [1] 1.730396
```

```r
mu.mse=mse_uni(var,mu.bias)

#The theoritical optimal block number k

thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}


# Sampling distribution of the MSE

k.exp=thres(0.05)
med=vector()
for(i in 1:1000){
  samp=sample(pop,100,replace=T)
  x <- seq_along(samp)
  d1 <- split(samp, (x/(100/k.exp)))
  df=data.frame(d1)
  means=colMeans(df[sapply(df, is.numeric)])
  med[i]=median(means)
}

#Plot the density of medians
plot(density(med),col="red",lwd=4,main = "density of median")
```



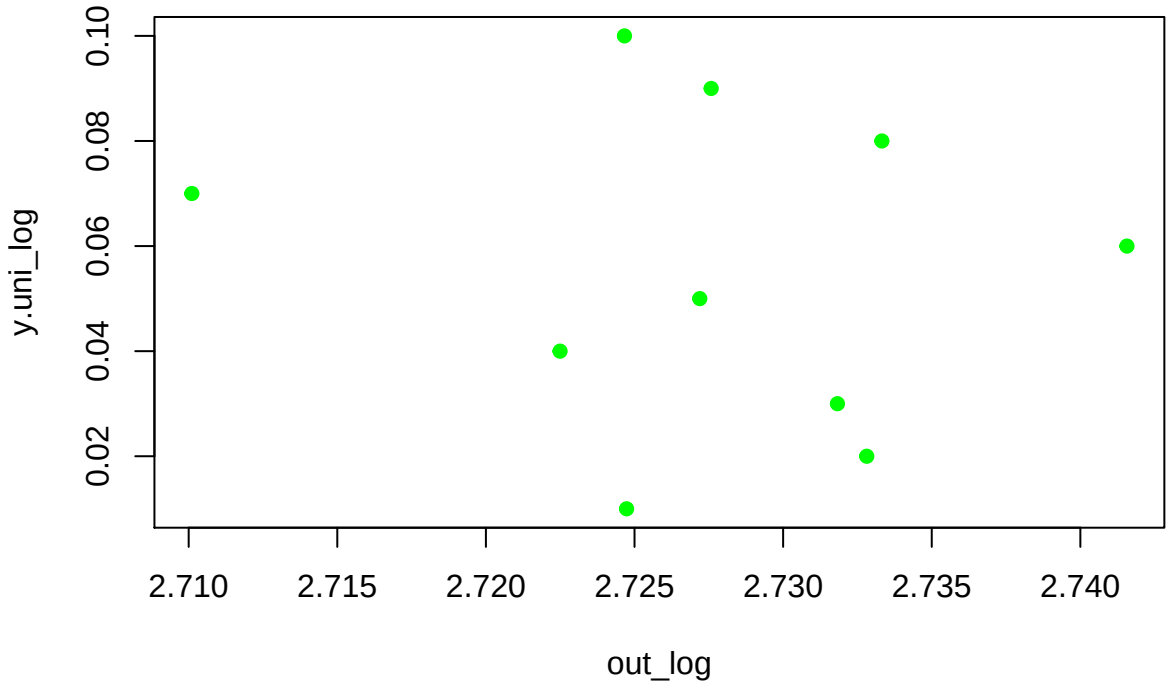

```
#Function to obtain the difference between the mean mse and median mse.
means=vector()
med=vector()
obj_uni=function(k){
  for(i in 1:1000){
    samp<-sample(pop,100, replace=T)
    x <- seq_along(samp)
    d1 <- split(samp, (x/(100/k)))
    df=data.frame(d1)
    means=colMeans(df[sapply(df, is.numeric)])
    med[i]=median(means)
  }
  med.bias=mean(med)-0
  med.var=var(med)
  med.mse=mse_uni(med.var,med.bias)
  return(mu.mse-med.mse)
}

#For different values
alpha=seq(0.01,0.1,0.01)
out_log=vector()
for (i in 1:length(alpha)){
  out_log[i]=obj_uni(thres(alpha[i]))
}
out_log
```

```
##  [1] 2.724734 2.732814 2.731828 2.722496 2.727196 2.741568 2.710106
##  [8] 2.733323 2.727579 2.724662
```

```
#Taking differnt values of alpha to plot the graph.
y.uni_log=vector()
for(i in 1:length(alpha)){
  y.uni_log[i]=alpha[i]
}
y.uni_log
```

```
##  [1] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10
```

```
#PLot to obtain the difference of mean mse and median mse against differnt values of alpha.
plot(y.uni_log~out_log, col="green", cex= 0.5, lwd=4)
```



Interpretation

We have taken a log-normal distribution which are heavy tailed distribution, calculated the MSE of mean and the MSE of MoM. Then, we have selected various number of blocks over a sequencce of alpha. From the output and the plot we have observed that the MSE of the MoM is not always smaller than the the MSE of mean. Thus, we can claim that MOM is more robut statistics.

Conclusion

We have taken Normal distribution as light tailed distribution and Pareto and Log-normal distribution as heavy tailed distribution. From the output and plot it is evident that both heavy and light tailed distribution the MSE of MOM is smaller than MSE of mean. But, compared to the light tailed distribution, for the heavy

tailed distribution the difference is much higher. Thus, we can conclude that MOM is more significant for heavy tailed distribution.

Question 2

2.1 Considering light tailed: Multivariate Normal population:

```
knitr::opts_chunk$set(echo = TRUE)

mu=rep(0,3)
mu
```

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

```
# We have to be careful of the fact that the corrlation matrix conventionally known as Sigma must be po
n <- 3
A <- matrix(runif(n^3)*2-1, ncol=n)
Sigma <- t(A) %*% A
sigma1 <- diag ( rep (1 , 9 ) )

# Now we can generate the random sample from the population

# Simple function to "normalize" a vector
norm_vec=function(x) x/ sqrt(sum(x^2))
sim=10000
sample=rmnorm(sim,mu,Sigma)
sample=t(apply(sample,1,norm_vec))

# Plotting the distribution
spheres3d (mu,lit=FALSE,col="bisque")
spheres3d (mu,radius=0.9,lit=FALSE,col="bisque", front="culled")
open3d()
```

```
## wgl
##   2
```

```
spheres3d(sample,col=blues9,radius = 0.02)
#rglwidget()

# Estimate the population mean vector
mean.ml=vector()
for (i in 1:3){
  mean.ml[i]<-mean(sample[,i])
}
mean.ml
```

```
## [1] -0.003799767 -0.002353852  0.002603139
```

```
# Calculate bias
bias<-mean.ml-mu
bias
```

```
## [1] -0.003799767 -0.002353852  0.002603139
```

```
# Calculate variance.
var=vector()
for (i in 1:3){
```

```
var[i]<-var(sample[,i])/10000
}
var
```

```
## [1] 2.824498e-05 2.634656e-05 4.541578e-05
```

```
mse.ml<-sum(var+bias^2)
mse.ml
```

```
## [1] 0.0001267625
```

```
## Function of split the matrix according to the block number

mat_split <- function(M, r, c){
  nr <- ceiling(nrow(M)/r)
  nc <- ceiling(ncol(M)/c)
  newM <- matrix(NA, nr*r, nc*c)
  newM[1:nrow(M), 1:ncol(M)] <- M

  div_k <- kronecker(matrix(seq_len(nr*nc), nr, byrow = TRUE), matrix(1, r, c))
  matlist <- split(newM, div_k)
  N <- length(matlist)
  mats <- unlist(matlist)
  dim(mats)<-c(r, c, N)
  return(mats)
}
```

Here we compare the MSE of the mean with the MSE of the geometric median. Our concern is check whether the MSE of mean is greater than the MSE of the geometric median. In order to compare the the above concern, we write a function which splits the matrix, then calculate the mean vector of each matrix, combine them and finally gives the geometric median. Once we get the geometric median, it is easy to make the comparison. Note that, there is a problem, the mat_split function allows to split the matrix iff the value of k is a factor of the number of rows on the parent matrix.

```
knitr::opts_chunk$set(echo = TRUE)

obj.ml=function(k,df){
  n=nrow(df)
  m=ncol(df)

  # We run a if loop where if k is a factor of row number, it directly yields the matrices, else, take
  if(nrow(df)%%k==0){
    testmat=mat_split(df,n/k,m)
    mean.mat=t(colMeans(testmat))
  }else{
    a=n-(n%%k)
    testmat=mat_split(df[1:a,],(a/k),3)
    a=testmat[,,k]
    testmat=testmat[,,-1]
    sep=tail(df,n%%k)
    p=rbind(a,sep)
    c=t(colMeans(p))
    d=t(colMeans(testmat))
    mean.mat=rbind(c,d)
  }
# Calculating the geometric median
```

```
gmed=Gmedian(mean.mat)
return(mse.ml-mse(gmed,mu))
}

obj.ml(20,sample)
```

```
## [1] 6.358394e-05
```

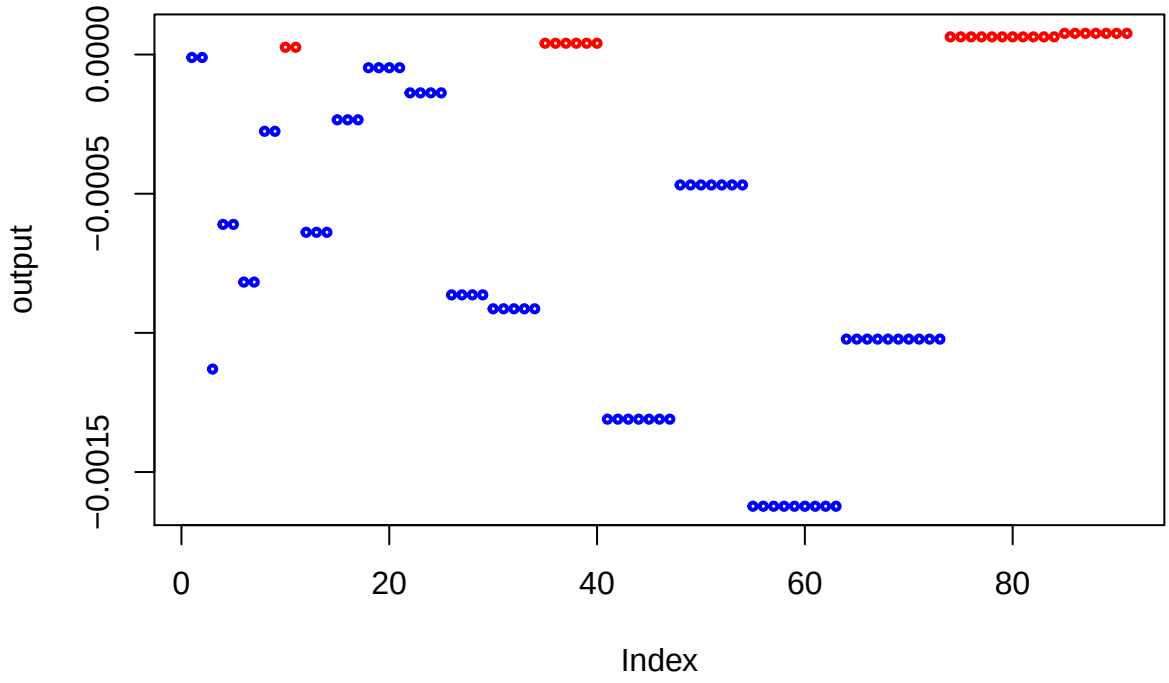
```
## considering the values of alpha
## We recall the function thres
thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}
alpha=seq(0.01,0.1,0.001)


k.ml=thres(alpha)
k.ml
```

```
##  [1] 37 37 36 35 35 34 34 33 33 32 32 31 31 31 30 30 30 29 29 29 29 28 28
## [24] 28 28 27 27 27 27 26 26 26 26 26 25 25 25 25 25 25 24 24 24 24 24 24
## [47] 24 23 23 23 23 23 23 23 22 22 22 22 22 22 22 22 22 21 21 21 21 21 21
## [70] 21 21 21 21 20 20 20 20 20 20 20 20 20 20 20 19 19 19 19 19 19 19
```

```
output=vector()
for (j in 1:length(alpha)){
  output[j]=obj.ml(thres(alpha[j]),sample)
}
output
```

```
##  [1] -1.049479e-05 -1.049479e-05 -1.130059e-03 -6.102676e-04 -6.102676e-04
##  [6] -8.175489e-04 -8.175489e-04 -2.757805e-04 -2.757805e-04  2.612039e-05
## [11]  2.612039e-05 -6.386454e-04 -6.386454e-04 -6.386454e-04 -2.344274e-04
## [16] -2.344274e-04 -2.344274e-04 -4.727859e-05 -4.727859e-05 -4.727859e-05
## [21] -4.727859e-05 -1.376454e-04 -1.376454e-04 -1.376454e-04 -1.376454e-04
## [26] -8.634786e-04 -8.634786e-04 -8.634786e-04 -8.634786e-04 -9.134092e-04
## [31] -9.134092e-04 -9.134092e-04 -9.134092e-04  4.056919e-05  4.056919e-05
## [36]  4.056919e-05  4.056919e-05  4.056919e-05  4.056919e-05  4.056919e-05
## [41] -1.310015e-03 -1.310015e-03 -1.310015e-03 -1.310015e-03 -1.310015e-03
## [46] -1.310015e-03 -1.310015e-03 -4.685156e-04 -4.685156e-04 -4.685156e-04
## [51] -4.685156e-04 -4.685156e-04 -4.685156e-04 -4.685156e-04 -1.622922e-03
## [56] -1.622922e-03 -1.622922e-03 -1.622922e-03 -1.622922e-03 -1.622922e-03
## [61] -1.622922e-03 -1.622922e-03 -1.622922e-03 -1.022403e-03 -1.022403e-03
## [66] -1.022403e-03 -1.022403e-03 -1.022403e-03 -1.022403e-03 -1.022403e-03
## [71] -1.022403e-03 -1.022403e-03 -1.022403e-03  6.358394e-05  6.358394e-05
## [76]  6.358394e-05  6.358394e-05  6.358394e-05  6.358394e-05  6.358394e-05
## [81]  6.358394e-05  6.358394e-05  6.358394e-05  6.358394e-05  7.617585e-05
## [86]  7.617585e-05  7.617585e-05  7.617585e-05  7.617585e-05  7.617585e-05
## [91]  7.617585e-05
```

```
plot(output, col= ifelse(output >= 0, "red", ifelse(output < 0,"blue", "black")),lwd=2,cex=0.5)
```




```
alpha_sig=alpha[which(output>0)]
alpha_sig
```

```
##  [1] 0.019 0.020 0.044 0.045 0.046 0.047 0.048 0.049 0.083 0.084 0.085
## [12] 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096
## [23] 0.097 0.098 0.099 0.100
```

```
#now we will see for the which k-value yields the lower MoM MSE than MSE of mean. For that we will use

k.ml_sig=thres(alpha_sig)
k.ml_sig
```

```
##  [1] 32 32 25 25 25 25 25 25 20 20 20 20 20 20 20 20 20 20 19 19 19 19
## [24] 19 19 19
```

```
# Finally we want to see for which value of k we get the lowest MSE
k.ml_min=thres(alpha[min(which(output>0))])
k.ml_min
```

```
## [1] 32
```

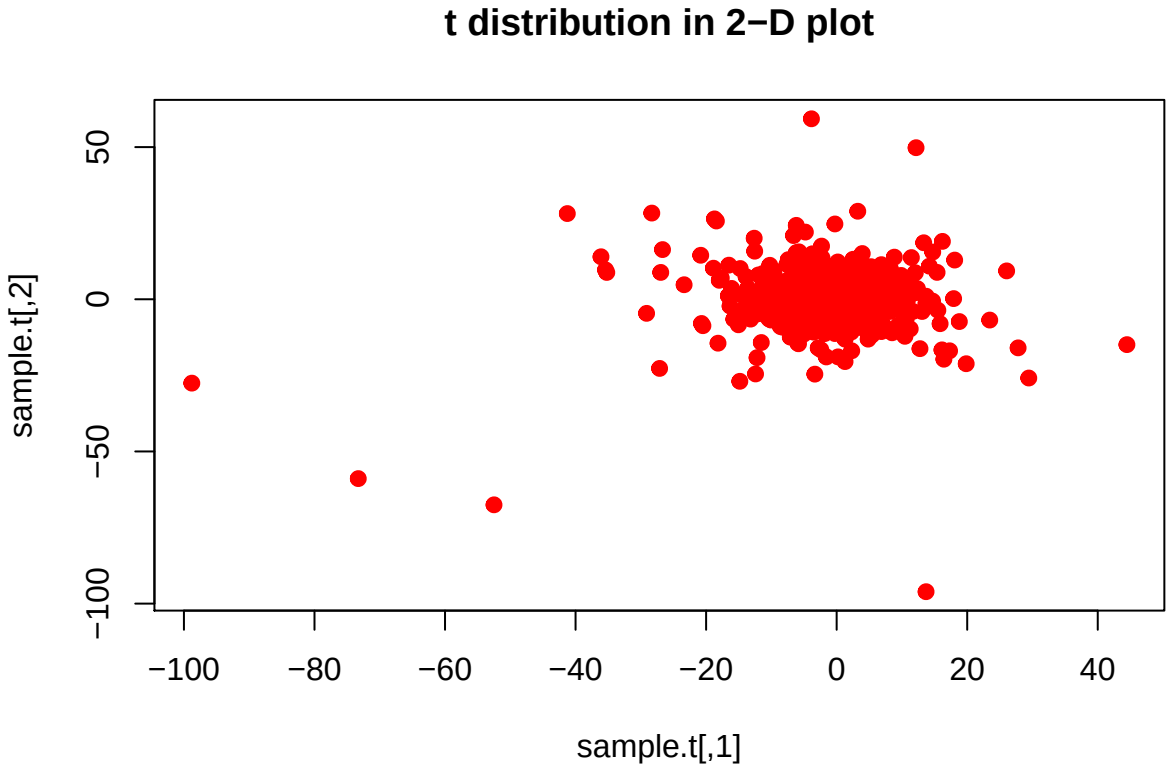
Interpretation

So, in a nutshell, we have taken a multivariate normal distrubution which is a light tailed distribution, calculated the MSE of mean and the MSE of the geometric median (MoM). Leter, we try to vary the number of blocks over a sequence of alpha. We have observed that the MSE of the MoM is not always smaller than the the MSE of mean. But as we observe a seuqence of alpha, it generates a sequence of the value of k. We observe that for some values if k, the MSE of MoM is smaller than the MSE of the mean. In this case, we

find the minimum MSE. Thus we can claim that partitioning the matrix into several blocks is indeed a better strategy and MoM is more robust statistic. but we have to be careful about the implimentations.

Considering heavy tailed: Multivariate t-distribution

```
knitr::opts_chunk$set(echo = TRUE)

sample.t=rmvt(10000,diag(3),df=2)
plot(sample.t,col = "red",lwd=7,cex=0.2,main = "t distribution in 2-D plot")
```




```
plot3d(sample.t[,1],sample.t[,2],sample.t[,3])
#rglwidget()
# Trivially, the mean vector is a null vector here.
mu.t=c(0,0,0)

# Estimate the population mean vector
mean.ml_t=vector()
for (i in 1:3){
  mean.ml_t[i]<-mean(sample.t[,i])
}
mean.ml_t
```

```
## [1] -0.04698944 -0.01202753 -0.01753112
```

```
# Calculate bias
bias.t<-mean.ml_t-mu.t
bias.t
```

```
## [1] -0.04698944 -0.01202753 -0.01753112
```

```
# Calculate variance.
var.t=vector()
for (i in 1:3){
  var.t[i]<-var(sample.t[,i])/10000
}
var.t
```

```
## [1] 0.0008308042 0.0008227267 0.0008790151
```

```
mse.ml_t<-sum(var.t+bias.t^2)
mse.ml_t
```

```
## [1] 0.005192554
```

```
## Function of split the matrix according to the block number

mat_split <- function(M, r, c){
  nr <- ceiling(nrow(M)/r)
  nc <- ceiling(ncol(M)/c)
  newM <- matrix(NA, nr*r, nc*c)
  newM[1:nrow(M), 1:ncol(M)] <- M

  div_k <- kronecker(matrix(seq_len(nr*nc), nr, byrow = TRUE), matrix(1, r, c))
  matlist <- split(newM, div_k)
  N <- length(matlist)
  mats <- unlist(matlist)
  dim(mats)<-c(r, c, N)
  return(mats)
}

#Here we compare the MSE of the mean with the MSE of the geometric median. Our concern is check whether
# In order to compare the the above concern, we write a function which splits the matrix, then calculat
# Once we get the geometric median, it is easy to make the comparison.
# Note that, there is a problem, the mat_split function allows to split the matrix iff the value of k i



obj.ml=function(k,df){
  n=nrow(df)
  m=ncol(df)

  # We run a if loop where if k is a factor of row number, it directly yields the matrices, else, take
  if(nrow(df)%%k==0){
    testmat=mat_split(df,n/k,m)
    mean.mat=t(colMeans(testmat))
  }else{
    a=n-(n%%k)
    testmat=mat_split(df[1:a,],(a/k),m)
    a=testmat[,,k]
    testmat=testmat[,,-1]
    sep=tail(df,n%%k)
    p=rbind(a,sep)
    c=t(colMeans(p))
    d=t(colMeans(testmat))
```

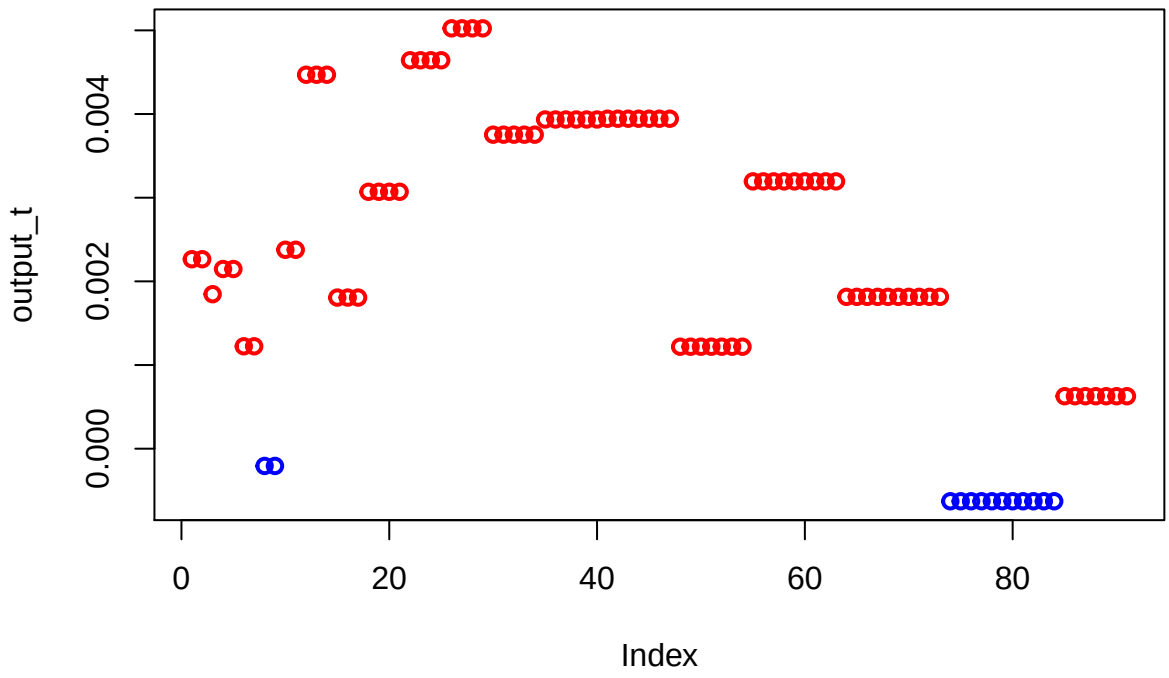
```
    mean.mat=rbind(c,d)
  }
  # Calculating the geometric median
  gmed=Gmedian(mean.mat)
  return(mse.ml_t-mse(gmed,mu.t))
}

## considering the values of alpha
## We recall the function thres
thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}
alpha=seq(0.01,0.1,0.001)


k.ml_t=thres(alpha)
k.ml_t
```

```
##  [1] 37 37 36 35 35 34 34 33 33 32 32 31 31 31 30 30 30 29 29 29 29 28 28
## [24] 28 28 27 27 27 27 26 26 26 26 26 25 25 25 25 25 25 24 24 24 24 24 24
## [47] 24 23 23 23 23 23 23 23 22 22 22 22 22 22 22 22 22 21 21 21 21 21 21
## [70] 21 21 21 21 20 20 20 20 20 20 20 20 20 20 20 19 19 19 19 19 19 19
```

```
output_t=vector()
for (j in 1:length(alpha)){
  output_t[j]=obj.ml(thres(alpha[j]),sample.t)
}
output_t
```

```
##  [1]  0.0022646377  0.0022646377  0.0018466569  0.0021477317  0.0021477317
##  [6]  0.0012240456  0.0012240456 -0.0002062970 -0.0002062970  0.0023775203
## [11]  0.0023775203  0.0044695921  0.0044695921  0.0044695921  0.0018063255
## [16]  0.0018063255  0.0018063255  0.0030705632  0.0030705632  0.0030705632
## [21]  0.0030705632  0.0046426115  0.0046426115  0.0046426115  0.0046426115
## [26]  0.0050240370  0.0050240370  0.0050240370  0.0050240370  0.0037532413
## [31]  0.0037532413  0.0037532413  0.0037532413  0.0037532413  0.0039346720
## [36]  0.0039346720  0.0039346720  0.0039346720  0.0039346720  0.0039346720
## [41]  0.0039440938  0.0039440938  0.0039440938  0.0039440938  0.0039440938
## [46]  0.0039440938  0.0039440938  0.0012189856  0.0012189856  0.0012189856
## [51]  0.0012189856  0.0012189856  0.0012189856  0.0012189856  0.0031951062
## [56]  0.0031951062  0.0031951062  0.0031951062  0.0031951062  0.0031951062
## [61]  0.0031951062  0.0031951062  0.0031951062  0.0018149829  0.0018149829
## [66]  0.0018149829  0.0018149829  0.0018149829  0.0018149829  0.0018149829
## [71]  0.0018149829  0.0018149829  0.0018149829 -0.0006281047 -0.0006281047
## [76] -0.0006281047 -0.0006281047 -0.0006281047 -0.0006281047 -0.0006281047
## [81] -0.0006281047 -0.0006281047 -0.0006281047 -0.0006281047  0.0006272100
## [86]  0.0006272100  0.0006272100  0.0006272100  0.0006272100  0.0006272100
## [91]  0.0006272100
```

```
plot(output_t, col= ifelse(output_t >= 0, "red", ifelse(output_t < 0,"blue", "black")),lwd=2)
```




```
alpha_sig_t=alpha[which(output_t>0)]
alpha_sig_t
```

```
##  [1] 0.010 0.011 0.012 0.013 0.014 0.015 0.016 0.019 0.020 0.021 0.022
## [12] 0.023 0.024 0.025 0.026 0.027 0.028 0.029 0.030 0.031 0.032 0.033
## [23] 0.034 0.035 0.036 0.037 0.038 0.039 0.040 0.041 0.042 0.043 0.044
## [34] 0.045 0.046 0.047 0.048 0.049 0.050 0.051 0.052 0.053 0.054 0.055
## [45] 0.056 0.057 0.058 0.059 0.060 0.061 0.062 0.063 0.064 0.065 0.066
## [56] 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074 0.075 0.076 0.077
## [67] 0.078 0.079 0.080 0.081 0.082 0.094 0.095 0.096 0.097 0.098 0.099
## [78] 0.100
```

```
#now we will see for the which k-value yields the lower MoM MSE than MSE of mean. For that we will use

k.ml_sig_t=thres(alpha_sig_t)
k.ml_sig_t
```

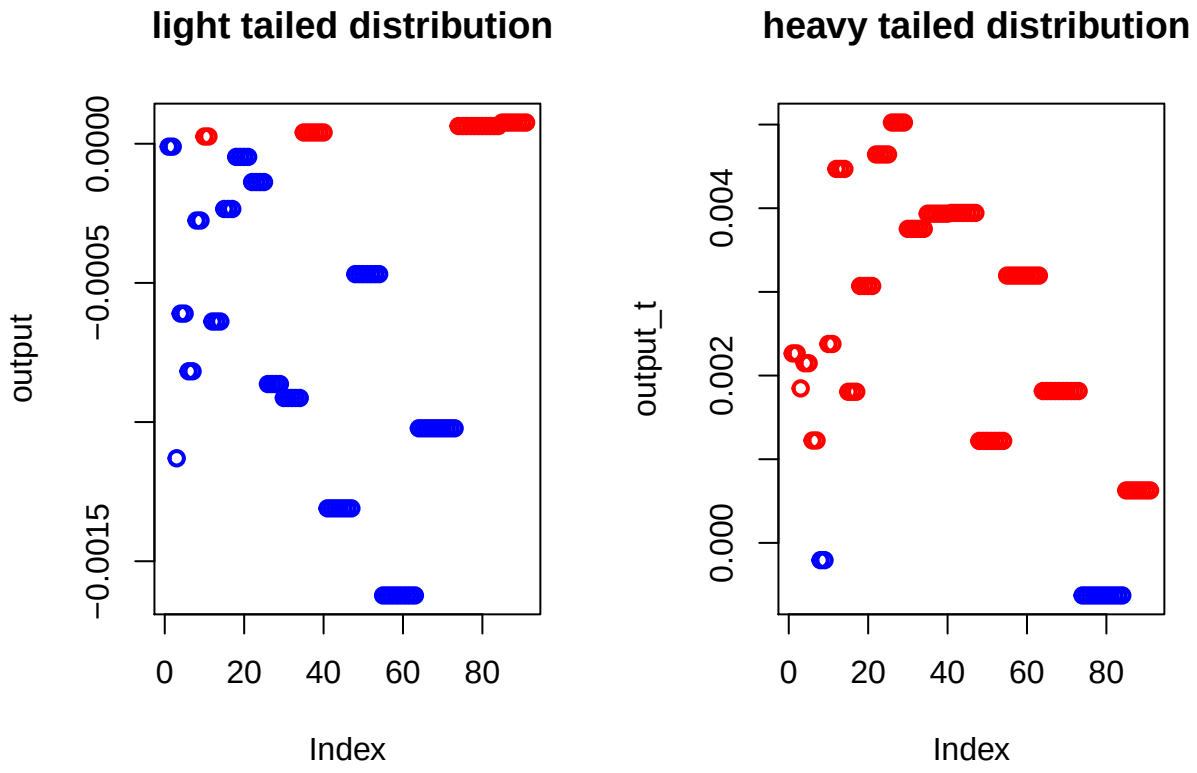
```
##  [1] 37 37 36 35 35 34 34 32 32 31 31 31 30 30 30 29 29 29 29 28 28 28 28
## [24] 27 27 27 27 26 26 26 26 26 25 25 25 25 25 25 24 24 24 24 24 24 24 23
## [47] 23 23 23 23 23 23 22 22 22 22 22 22 22 22 22 21 21 21 21 21 21 21 21
## [70] 21 21 19 19 19 19 19 19 19
```

```
# Finally we want to see for which value of k we get the lowest MSE
k.ml_min_t=thres(alpha[min(which(output_t>0))])
k.ml_min_t
```

```
## [1] 37
```

Comparison between a heavy tailed and light tailed distribution

```
knitr::opts_chunk$set(echo = TRUE)
par(mfrow=c(1,2))
p1=plot(output, col= ifelse(output >= 0, "red", ifelse(output < 0,"blue", "black")),lwd=2,main = " ligh
p2=plot(output_t, col= ifelse(output_t >= 0, "red", ifelse(output_t < 0,"blue", "black")),lwd=2,main=" h
```



Conclusion

The difference in the application of the MoM to heavy tailed and light tailed is pretty clear now. From the above image, the difference between the MSE of sample mean and MoM is classified in two colors viz. red and blue. Where red signifies the MoM MSE is lower than the MSE of sample. Now, in the first picture we see that the presence of red dots is much lower than the blue dots. That means, for a light tailed distribution, the sample means performs better than the MoM. But the scenario is entirely different in the case of a heavy tailed distribution. We took multivariate t distribution which is indeed a heavy tailed distribution. We see that, in most of the cases, the MSE of MoM is lower than the MSE of the sample mean. That is, the performance of MoM is better than the sample mean in the case of a heavy tailed distribution.

Question 3

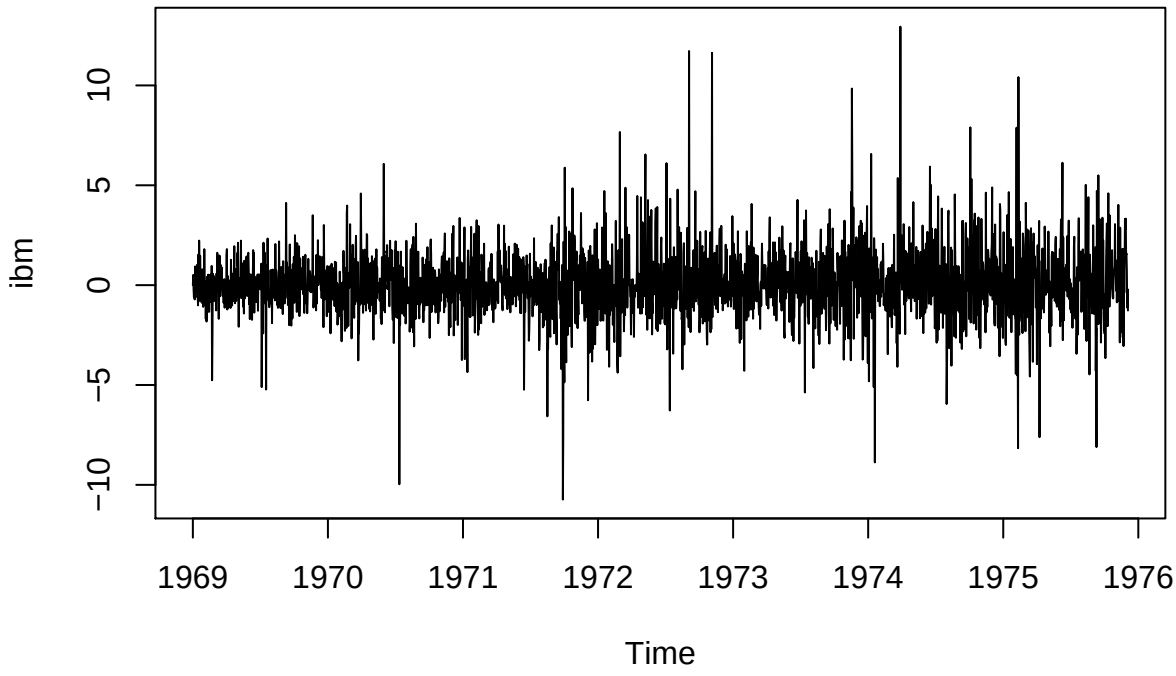
Exploring the dataset numerically

```
knitr::opts_chunk$set(echo = TRUE)

# Importing the dataset
load("~/Sapienza Learning Materials/Brutti_sem1/HW3/CRSPday.RData")
```

```
dimnames(CRSPday)[[2]]
```

```
## [1] "year"  "month" "day"   "ge"    "ibm"   "mobil" "crsp"
```

```
ibm=100*CRSPday[,5]
plot(ibm)
```




```
mode(ibm)
```

```
## [1] "numeric"
```

```
class(ibm)
```

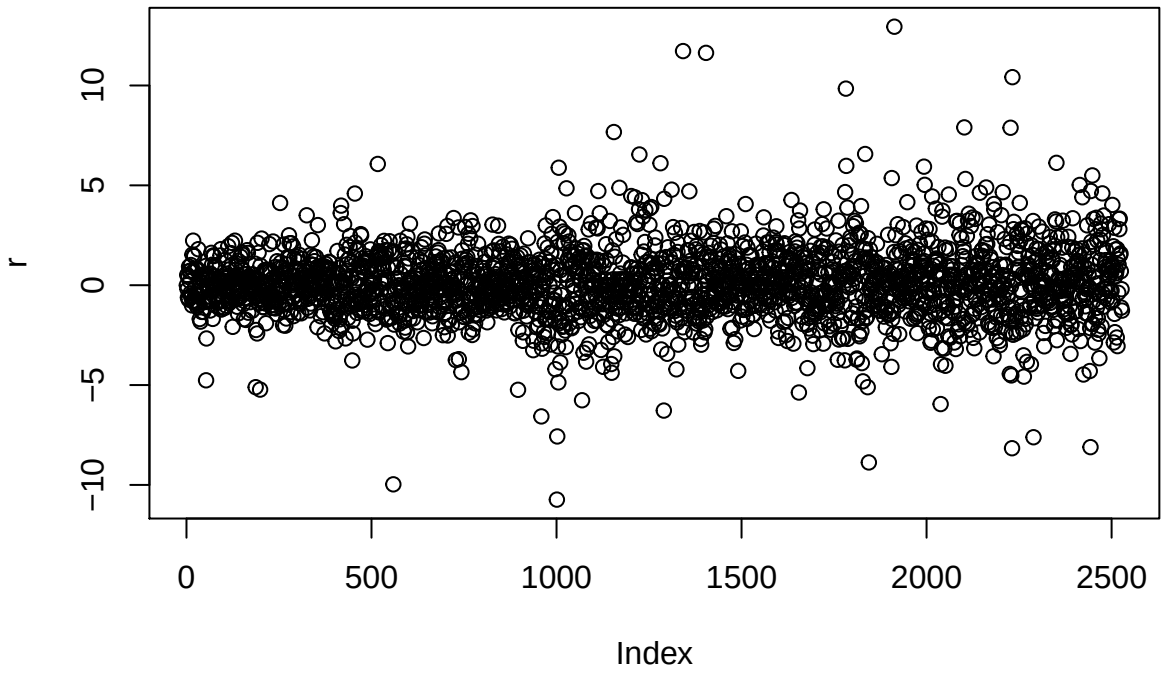
```
## [1] "ts"
```

The class of the variable IBM is “ts” which means time series. The time series data is plotted differently. So, the data of IBM is converted to numeric before plotting

```
knitr::opts_chunk$set(echo = TRUE)
r=as.numeric(ibm)
class(r)
```

```
## [1] "numeric"
```

```
plot(r)
```

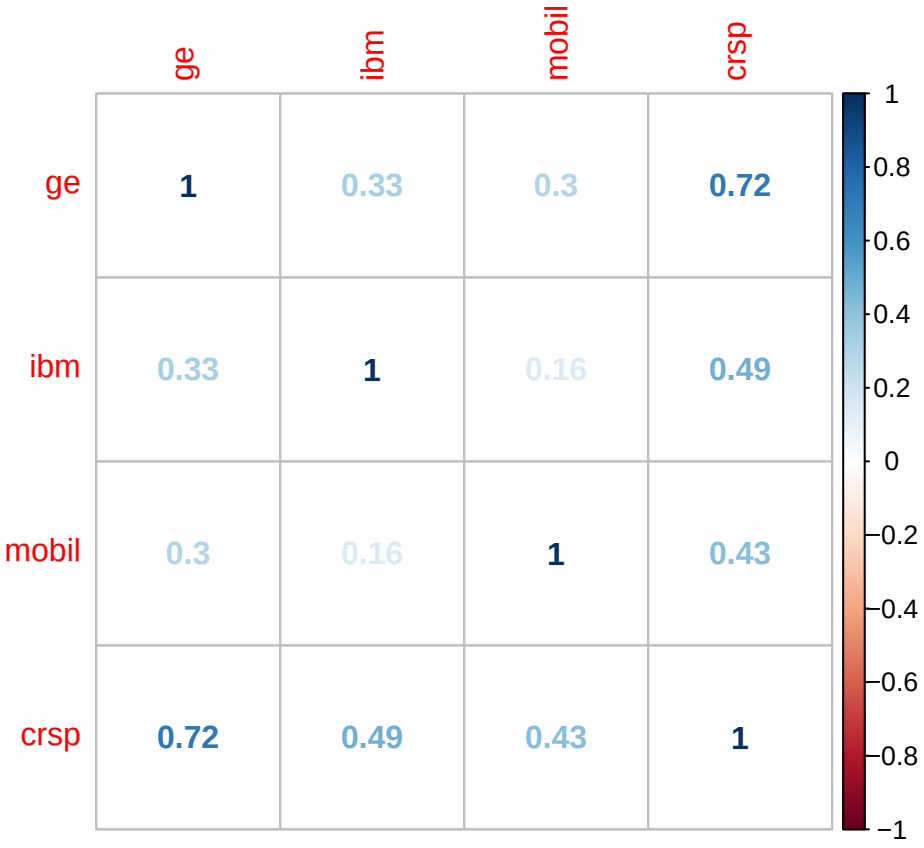


Exploring the variables it is evident that these are time series data. One varaible (IBM) is considered to illustrate it properly.

```
knitr::opts_chunk$set(echo = TRUE)
#Covariance matrix of GE, IBM, MOBIL and CRSP
covariance=(cov(CRSPday[,4:7]))
```

It is difficult to get much information just by inspecting the covariance matrix. The covariance between two random variables depends on their variances as well as the strength of the linear relationship between them. Covariance matrices are extremely important as input to, for example, a portfolio analysis but to understand the relationship between variables, it is much better to examine their sample correlation matrix.

```
knitr::opts_chunk$set(echo = TRUE)
#Correlation matrix of GE, IBM, MOBIL and CRSP
correlation=round(cor(CRSPday[,4:7]),3)
corrplot(correlation, method = "number")
```

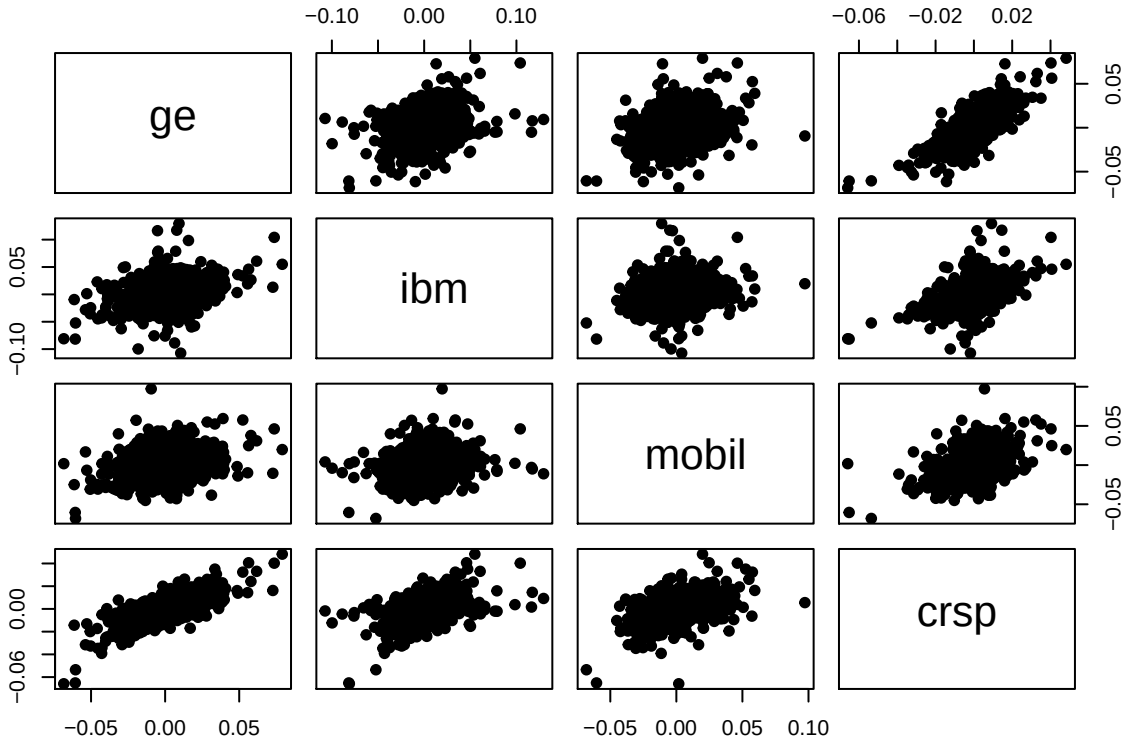


Interpretation

From the correlation matrix we ccan conclude that the sample correlations are positive and the largest correlatons are between CRSP and the individual stocks. GE is the stock most highy correlated with crsp. Thus, the correlation between individual stock and a market index such as crsp is a key component of finance theory.

A correlation coefficient is only a summary of the linear relationship between variables. Interesting features, such as nonlinearity or the joint behavior of extreme values, remain hidden when only correlations are examined. A solution to this problem is the scatterplot matrix, which is a matrix of scatterplots, one for each pair of variables.

```
knitr::opts_chunk$set(echo = TRUE)
pairs(CRSPday[,4:7], pch=19)
```



Interpretation

This is the nonlinearr relationships between variables. The strong linear association between GE and crsp which has been shown before by their high correlation coeficient can also be seen in the scatter plot. In the scatterplot for IBM and Mobil, extreme returns for one stock do not tend to occur on the same days as extreme returns on the other stock; this can be seen by noticing that the outliers tend to fall along the x- and y-axes. The extreme-value behavior is different with GE and crsp, where extreme values are more likely to occur together; note that the outliers have a tendency to occur together, that is, in the upper-right and lower-left corners, rather than being concentrated along the axes. The IBM and Mobil scatterplot is said to show tail independence. In contrast, the GE and crsp scatterplot is said to show tail dependence.

```
knitr::opts_chunk$set(echo = TRUE)

#Naming the variables
ge = CRSPday[,4]
ibm = CRSPday[,5]
mobil = CRSPday[,6]
crsp = CRSPday[,7]
#Hypothesis test
#H_0: rho(1,j)=0
#H_1: rho(i,j)!=0
#Level of test
alpha=0.05
#crsp vs ge
cg<-cor.test(crsp, ge, conconf.level = 1-alpha)
cg
```

```
##
##  Pearson's product-moment correlation
##
## data:  crsp and ge
## t = 51.374, df = 2526, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6952109 0.7333701
## sample estimates:
##       cor
## 0.7148222
```

```
cg$p.value
```

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

```
#We can reject null hypothesis
cg$conf.int
```

```
## [1] 0.6952109 0.7333701
## attr(,"conf.level")
## [1] 0.95
```

```
#It does not include 0, we can reject H_0
#crsp vs ibm
ci<-cor.test(crsp, ibm, conconf.level = 1-alpha)
ci
```

```
##
##  Pearson's product-moment correlation
##
## data:  crsp and ibm
## t = 27.976, df = 2526, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4560256 0.5155732
## sample estimates:
##       cor
## 0.4863639
```

```
ci$p.value
```

```
## [1] 2.918482e-150
```

```
#We can reject null hypothesis
ci$conf.int
```

```
## [1] 0.4560256 0.5155732
## attr(,"conf.level")
## [1] 0.95
```

```
# It does not include 0, we can reject H_0
# crsp vs mobil
cm<-cor.test(crsp, mobil, conconf.level = 1-alpha)
cm
```

```
##
##  Pearson's product-moment correlation
##
```

```
## data:  crsp and mobil
## t = 23.896, df = 2526, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3970563 0.4606680
## sample estimates:
##       cor
## 0.4293946
```

```
cm$p.value
```

```
## [1] 5.97643e-114
```

```
# We can reject null hypothesis
cm$conf.int
```

```
## [1] 0.3970563 0.4606680
## attr(,"conf.level")
## [1] 0.95
```

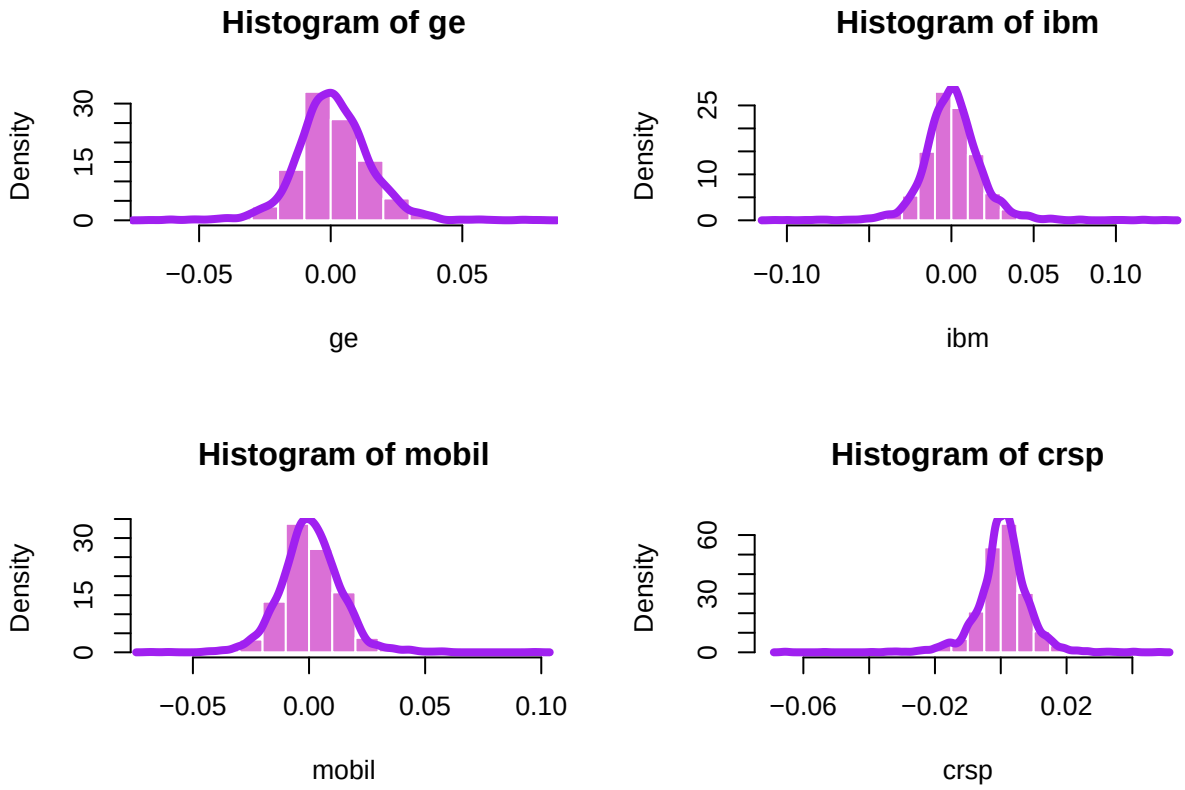
```
# It does not include 0, we can reject H_0
```

Interpretation

For all the test the null hypothesis has been rejected. Thus, it is evident that their exists relationship between market index such as CRSP and the individual stocks.

Exploration of the dataset graphically

```
knitr::opts_chunk$set(echo = TRUE)
par(mfrow=c(2,2))
ge=CRSPday[,4]
hist(ge, probability = T, border = "white", col = "orchid", breaks = 20)
lines( density(ge, na.rm = T), col = "purple", lwd = 4)
ibm=CRSPday[,5]
hist(ibm, probability = T, border = "white", col = "orchid", breaks = 20)
lines( density(ibm, na.rm = T), col = "purple", lwd = 4)
mobil=CRSPday[,6]
hist(mobil, probability = T, border = "white", col = "orchid", breaks = 20)
lines( density(mobil, na.rm = T), col = "purple", lwd = 4)
crsp=CRSPday[,7]
hist(crsp, probability = T, border = "white", col = "orchid", breaks = 20)
lines( density(crsp, na.rm = T), col = "purple", lwd = 4)
```



Interpretation

From the plots this is evident that the variables are normally distribution.

```
knitr::opts_chunk$set(echo = TRUE)


# We will consider only the part of the data to be considered.
data=CRSPday[,4:7]


bstrap <- function(df) {
  df[sample(1:nrow(df),10000, replace = TRUE),]
}

boot.data=bstrap(data)
dim(boot.data)
```

```
## [1] 10000     4
```

```
mu.data=colMeans(boot.data)
mu.data
```

```
##           ge          ibm        mobil         crsp
## 0.0010806513 0.0005082722 0.0006554504 0.0006383968
```

```
# Now in order to find the geometric median, we have to split the matrix wich can be done with the mat_
```

```r
obj.data=function(k,df){
  n=nrow(df)
  m=ncol(df)

  # We run a if loop where if k is a factor of row number, it directly yields the matrices, else, take
  if(nrow(df)%%k==0){
    testmat=mat_split(df,n/k,m)
    mean.mat=t(colMeans(testmat))
  }else{
    a=n-(n%%k)
    testmat=mat_split(df[1:a,],(a/k),m)
    a=testmat[,,k]
    testmat=testmat[,,-1]
    sep=tail(df,n%%k)
    p=rbind(a,sep)
    c=t(colMeans(p))
    d=t(colMeans(testmat))
    mean.mat=rbind(c,d)
  }
  # Calculating the geometric median
  gmed=Gmedian(mean.mat)
  return(mse(gmed,mu.data))
}

## considering the values of alpha
## We recall the function thres
thres=function(alpha){
  k=ceiling(8*log(1/alpha))
  return(k)
}
alpha=c(0.01,0.05,0.1, 0.2)
mom_mse.data=vector()
for (i in 1:length(alpha)){
  mom_mse.data[i]=obj.data(thres(alpha[i]),boot.data)
}
mom_mse.data
```

```
## [1] 0.0004942595 0.0003418467 0.0004931143 0.0002426546
```

```r
min(mom_mse.data)
```

```
## [1] 0.0002426546
```

```r
alpha[which(mom_mse.data==min(mom_mse.data))]
```

```
## [1] 0.2
```

```r
k.val=thres(0.012)
k.val
```

```
## [1] 36
```

Conclusion

Non-parametric bootstrap is used to estimate the MSE of MOM. The k.val represents the block size which has the smallest MSE.