Appendix (code)

```
# Data loading
data <- read.csv("/Users/macbook/Downloads/full_close.csv")
n <- dim(data)[1]
data_5 <- data[,c(2,3,5,6)]
return <- data_5[2:n,]/data_5[1:(n-1),]-1
log_return <- log(data_5[2:n,]/data_5[1:(n-1),])

# Data Description
## QQ Plot for Net_Return
layout(matrix(c(1,1,1,1,1,0,2,2,2,2,2,
                3,3,3,3,3,0,4,4,4,4,4,
                0,0,0,5,5,5,5,5,0,0,0), 3, 11, byrow = TRUE))


qqnorm(return[,1], main = 'QQ plot for PFE')
qqline(return[,1], col ='blue')

qqnorm(return[,2], main = 'QQ plot for JNJ')
qqline(return[,2], col ='blue')

qqnorm(return[,3], main = 'QQ plot for MRK')
qqline(return[,3], col ='blue')

qqnorm(return[,4], main = 'QQ plot for NVS')
qqline(return[,4], col ='blue')

layout.show(4)

## Histogram for Net_Return
layout(matrix(c(1,1,1,1,1,0,2,2,2,2,2,
                3,3,3,3,3,0,4,4,4,4,4,
                0,0,0,5,5,5,5,5,0,0,0), 3, 11, byrow = TRUE))


hist(return[,1], breaks = 30, main = 'Histogram for PFE', xlab = "PFE Net Return" )


hist(return[,2], breaks = 30, main = 'Histogram for JNJ', xlab = "JNJ Net Return" )


hist(return[,3], breaks = 30, main = 'Histogram for MRK',xlab = "MRK Net Return" )


hist(return[,4], breaks = 30, main = 'Histogram for NVS', xlab = "NVS Net Return" )

layout.show(4)


# Time-Series Analysis

library(forecast)
```

```r
library(fGarch)
library(rugarch)
library(MASS)
returns<-read.csv('C:/users/hongy/Desktop/return.csv',header=T)
PFEr<-ts(returns[,2],frequency=365)[c(1:1218)]
ts.plot(PFEr)

arima_110_111 <- arima(PFEr, order=c(1,1,0), seasonal = list(order = c(1,1,1), period = 12), method="ML

pred<- predict(arima_110_111, n.ahead = 100)
U.tr= pred$pred + 1.96*pred$se
L.tr= pred$pred - 1.96*pred$se

ts.plot(PFEr,xlim=c(1100,length(PFEr))+100,ylim=c(-0.2,0.2), main='Time Series Forcast On PFE Stock Ret
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
## Predicted values of transformed data
points((length(PFEr)+1):(length(PFEr)+100), pred$pred, col="red")
abline(h=0,col='red')

JNJr<-ts(returns[,3],frequency=365)[c(1:1218)]

ts.plot(JNJr)

ts.plot(JNJr,xlim=c(1100,length(JNJr))+100,ylim=c(-0.2,0.2), main='Time Series Forcast on JNJ Stock Ret
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
## Predicted values of transformed data
points((length(JNJr)+1):(length(JNJr)+100), pred$pred, col="red")
abline(h=0,col='red')

MRKr<-ts(returns[,5],frequency=365)[c(1:1218)]

ts.plot(MRKr)

ts.plot(MRKr,xlim=c(1100,length(MRKr))+100,ylim=c(-0.2,0.2), main='Time Series Forcast On MRK Stock Ret
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
## Predicted values of transformed data
points((length(MRKr)+1):(length(MRKr)+100), pred$pred, col="red")
abline(h=0,col='red')

NVSr<-ts(returns[,6],frequency=365)[c(1:1218)]

ts.plot(NVSr)

ts.plot(NVSr,xlim=c(1100,length(NVSr))+100,ylim=c(-0.2,0.2), main='Time Series Forcast On NVS Stock Ret
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
## Predicted values of transformed data
points((length(NVSr)+1):(length(NVSr)+100), pred$pred, col="red")
abline(h=0,col='red')
```

```r
# Portfolio Theory
## Load data
library(readr)
data=read_csv("full_close.csv")
prices=data
n=dim(prices)[1]
returns =(prices[2:n,]/prices[1:(n-1),]-1) # returns by day
library(tidyr)
library(dplyr)
## mean, covariance, standard deviation of PRICES
mean.p = colMeans(prices)
cov.p = cov(prices)
sd.p = sqrt(diag(cov.p))
## mean, covariance, standard deviation of RETURN
mean.r = colMeans(returns)
cov.r = cov(returns)
sd.r = sqrt(diag(cov.r))
## Skewness Coefficients, Kurtosis Coefficients and beta of PRICES
list=as.list(prices)
sk.p=unlist(lapply(list,function(data){
  (sum((data-mean(data))^3/(sd(data))^3))/length(data)
}))
kt.p=unlist(lapply(list,function(data){
  (sum((data-mean(data))^4/(sd(data))^4))/length(data)
}))
## Info Table
info=cbind(mean.r,sd.r,sk.p,kt.p)
colnames(info)=c("mean","sd","skewness","kurtosis")
sd.r[order(sd.r,decreasing = F)]

##Sharpe's Ratio (using sample mean &sd in month)
rf.m=1.73/100/365 #daily risk free
sharpes.ratio=(mean.r-rf.m)/sd.r
sort(sharpes.ratio,decreasing = T)
cat("\n The biggest sharpe ratio is MRK:",max(sharpes.ratio))

library(quadprog)
library(Ecdat)
efficient.portfolio <-
function(er, cov.mat, target.return, shorts=TRUE)
{
  # compute minimum variance portfolio subject to target return
  #
  # inputs:
  # er                      N x 1 vector of expected returns
  # cov.mat               N x N covariance matrix of returns
  # target.return     scalar, target expected return
  # shorts           logical, allow shorts is TRUE
  #
  # output is portfolio object with the following elements
  # call                    original function call
  # er                      portfolio expected return
  # sd                      portfolio standard deviation
```

```r
  # weights              N x 1 vector of portfolio weights
  #
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  er <- as.vector(er)                    # assign names if none exist
  N <- length(er)
  cov.mat <- as.matrix(cov.mat)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semidefinite
  #
  # compute efficient portfolio
  #
  if(shorts==TRUE){
    ones <- rep(1, N)
    top <- cbind(2*cov.mat, er, ones)
    bot <- cbind(rbind(er, ones), matrix(0,2,2))
    A <- rbind(top, bot)
    b.target <- as.matrix(c(rep(0, N), target.return, 1))
    x <- solve(A, b.target)
    w <- x[1:N]
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    Amat <- cbind(rep(1,N), er, diag(1,N))
    bvec <- c(1, target.return, rep(0,N))
    result <- solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=2)
    w <- round(result$solution, 6)
  } else {
    stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }
  #
  # compute portfolio expected returns and variance
  #
  names(w) <- asset.names
  er.port <- crossprod(er,w)
  sd.port <- sqrt(w %*% cov.mat %*% w)
  ans <- list("call" = call,
          "er" = as.vector(er.port),
          "sd" = as.vector(sd.port),
          "weights" = w)
  class(ans) <- "portfolio"
  ans
}

globalMin.portfolio <-function(er, cov.mat, shorts=TRUE){
  # Compute global minimum variance portfolio
  #
```

```r
  # inputs:
  # er              N x 1 vector of expected returns
  # cov.mat     N x N return covariance matrix
  # shorts          logical, allow shorts is TRUE
  #
  # output is portfolio object with the following elements
  # call            original function call
  # er              portfolio expected return
  # sd              portfolio standard deviation
  # weights     N x 1 vector of portfolio weights
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  er <- as.vector(er)                   # assign names if none exist
  cov.mat <- as.matrix(cov.mat)
  N <- length(er)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semi-definite
  #
  # compute global minimum portfolio
  #
  if(shorts==TRUE){
    cov.mat.inv <- solve(cov.mat)
    one.vec <- rep(1,N)
    w.gmin <- rowSums(cov.mat.inv) / sum(cov.mat.inv)
    w.gmin <- as.vector(w.gmin)
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    Amat <- cbind(rep(1,N), diag(1,N))
    bvec <- c(1, rep(0,N))
    result <- solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=1)
    w.gmin <- round(result$solution, 6)
  } else {
    stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }

  names(w.gmin) <- asset.names
  er.gmin <- crossprod(w.gmin,er)
  sd.gmin <- sqrt(t(w.gmin) %*% cov.mat %*% w.gmin)
  gmin.port <- list("call" = call,
          "er" = as.vector(er.gmin),
          "sd" = as.vector(sd.gmin),
          "weights" = w.gmin)
  class(gmin.port) <- "portfolio"
  gmin.port
}
```

```
efficient.frontier <- function(er, cov.mat, nport=20, alpha.min=-0.5, alpha.max=1.5, shorts=TRUE)
{
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  er <- as.vector(er)
  N <- length(er)
  cov.mat <- as.matrix(cov.mat)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  #
  # create portfolio names
  #
  port.names <- rep("port",nport)
  ns <- seq(1,nport)
  port.names <- paste(port.names,ns)
  #
  # compute global minimum variance portfolio
  #
  cov.mat.inv <- solve(cov.mat)
  one.vec <- rep(1, N)
  port.gmin <- globalMin.portfolio(er, cov.mat, shorts)
  w.gmin <- port.gmin$weights
  if(shorts==TRUE){
    # compute efficient frontier as convex combinations of two efficient portfolios
    # 1st efficient port: global min var portfolio
    # 2nd efficient port: min var port with ER = max of ER for all assets
    er.max <- max(er)
    port.max <- efficient.portfolio(er,cov.mat,er.max)
    w.max <- port.max$weights
    a <- seq(from=alpha.min,to=alpha.max,length=nport) # convex combinations
    we.mat <- a %o% w.gmin + (1-a) %o% w.max             # rows are efficient portfolios
    er.e <- we.mat %*% er                                           # expected returns of efficient po
    er.e <- as.vector(er.e)
  } else if(shorts==FALSE){
    we.mat <- matrix(0, nrow=nport, ncol=N)
    we.mat[1,] <- w.gmin
    we.mat[nport, which.max(er)] <- 1
    er.e <- as.vector(seq(from=port.gmin$er, to=max(er), length=nport))
    for(i in 2:(nport-1))
      we.mat[i,] <- efficient.portfolio(er, cov.mat, er.e[i], shorts)$weights
  } else {
    stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }

  names(er.e) <- port.names
  cov.e <- we.mat %*% cov.mat %*% t(we.mat) # cov mat of efficient portfolios
  sd.e <- sqrt(diag(cov.e))                          # std of efficient portfolios
  sd.e <- as.vector(sd.e)
```

```r
  names(sd.e) <- port.names
  dimnames(we.mat) <- list(port.names,asset.names)
  #
  # summarize results
  #
  ans <- list("call" = call,
              "er" = er.e,
              "sd" = sd.e,
              "weights" = we.mat)
  class(ans) <- "Markowitz"
  ans
}

tangency.portfolio <- function(er,cov.mat,risk.free, shorts=TRUE)
{
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  if(risk.free < 0)
    stop("Risk-free rate must be positive")
  er <- as.vector(er)
  cov.mat <- as.matrix(cov.mat)
  N <- length(er)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semi-definite
  #
  # compute global minimum variance portfolio
  #
  gmin.port <- globalMin.portfolio(er, cov.mat, shorts=shorts)
  if(gmin.port$er < risk.free)
    stop("Risk-free rate greater than avg return on global minimum variance portfolio")
  #
  # compute tangency portfolio
  #
  if(shorts==TRUE){
    cov.mat.inv <- solve(cov.mat)
    w.t <- cov.mat.inv %*% (er - risk.free) # tangency portfolio
    w.t <- as.vector(w.t/sum(w.t))          # normalize weights
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    er.excess <- er - risk.free
    Amat <- cbind(er.excess, diag(1,N))
    bvec <- c(1, rep(0,N))
    result <- quadprog::solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=1)
    w.t <- round(result$solution/sum(result$solution), 6)
  } else {
    stop("Shorts needs to be logical. For no-shorts, shorts=FALSE.")
```

```
  }

  names(w.t) <- asset.names
  er.t <- crossprod(w.t,er)
  sd.t <- sqrt(t(w.t) %*% cov.mat %*% w.t)
  tan.port <- list("call" = call,
                   "er" = as.vector(er.t),
                   "sd" = as.vector(sd.t),
                   "weights" = w.t)
  class(tan.port) <- "portfolio"
  return(tan.port)
}
plot.portfolio <- function(object, ...)
{
  asset.names <- names(object$weights)
  barplot(object$weights, names=asset.names,
      xlab="Assets", ylab="Weight", main="Portfolio Weights", ...)
  invisible()
}

##SHORT SELL IS NOT ALLOWED
##mvp with no short sell
mvp.noshort=globalMin.portfolio(mean.r,cov.r,shorts=FALSE)
mvp.ns.sd=mvp.noshort$sd
mvp.ns.er=mvp.noshort$er
mvp.ns.w=mvp.noshort$weights
##Tangency Portfolio
tangency.noshort=tangency.portfolio(mean.r,cov.r,risk.free = rf.m,shorts = FALSE)
tg.ns.er=tangency.noshort$er #tangency return
tg.ns.sd=tangency.noshort$sd
tg.ns.w=tangency.noshort$weights
##Efficient Portfolio Frontier
eff.front.noshort=efficient.frontier(mean.r,cov.r,nport=50, shorts = FALSE)
ef.ns.er=eff.front.noshort$er
ef.ns.sd=eff.front.noshort$sd
ef.ns.w=eff.front.noshort$weights
##Data Table
library(formattable)
noshort.table=data.frame("risk.free.day"=rbind(round(rf.m,5),0) ,
                       "mvp day"=rbind(mvp.ns.er,mvp.ns.sd),
                       "tangency.day"=rbind(tg.ns.er,tg.ns.sd),
                       "mvp year"=rbind(mvp.ns.er*365,mvp.ns.sd*sqrt(365)),
                       "Tangency year"=rbind(tg.ns.er*365,tg.ns.sd*sqrt(365)),
                       "risk.free.year"=rbind(rf.m*365,0),
                       row.names = c("return","risk"))
formattable(noshort.table)
##Value at Risk
s=100000
## t= one month
##Loss~N(-mean.mvp,sd.mvp), assuming mvp is normal
VaR.ns.mvp=(-mvp.ns.er+mvp.ns.sd*qnorm(0.95))*s
VaR.r=(-mean.r+sd.r*qnorm(0.95))*s
##Sharpe Ratio
```

```r
sharpe.ns = ( ef.ns.er- rf.m) / ef.ns.sd
## compute Sharpe's ratios
(tg.ns.sharpe=max(sharpe.ns))
assets.ns.sharpe=(mean.r -rf.m)/sd.r

sort(mvp.ns.w,decreasing = T)

sort(tg.ns.w,decreasing = T)

sort(sd.r,decreasing = F)

##SHORT SELL IS NOT ALLOWED
##Portfolio Plot

plot(ef.ns.sd,ef.ns.er,type="l",main="Efficient Frontier (No Short)",
     xlab="daily Risk", ylab="daily Return",
     ylim = c(0,0.0008),xlim=c(0,0.025),lty=3)

text(sqrt(cov.r[1,1]),mean.r[1],'PFE',cex=1.1)
text(sqrt(cov.r[2,2]),mean.r[2],'JNJ',cex=1.1)
text(sqrt(cov.r[3,3]),mean.r[3],'MRK',cex=1.1)
text(sqrt(cov.r[4,4]),mean.r[4],'NVS',cex=1.1)

points(0, rf.m, cex = 4, pch = "*") # show risk-free asset
sharpe = ( ef.ns.er- rf.m) / ef.ns.sd # compute Sharpe's ratios
ind = (sharpe == max(sharpe)) # Find maximum Sharpe's ratio
lines(c(0, 2), rf.m + c(0, 2) * (ef.ns.er[ind] - rf.m) / ef.ns.sd[ind], lwd = 4, lty = 1, col = "blue")
points(ef.ns.sd[ind], ef.ns.er[ind], cex = 4, pch = "*") # tangency portfolio
ind2 = (ef.ns.sd == min(ef.ns.sd)) # find minimum variance portfolio
points(ef.ns.sd[ind2], ef.ns.er[ind2], cex = 2, pch = "+") # min var portfolio
ind3 = (ef.ns.er > ef.ns.er[ind2])
lines(ef.ns.sd[ind3], ef.ns.er[ind3], type = "l", lwd = 3, col = "red") # plot efficient frontier

##SHORT SELL IS ALLOWED
##ALL VARIABLES are in MONTH
##mvp
mvp.short=globalMin.portfolio(mean.r ,cov.r,shorts=TRUE)
mvp.s.sd=mvp.short$sd
mvp.s.er=mvp.short$er
mvp.s.w=mvp.short$weights
##Tangency Portfolio
tangency.short=tangency.portfolio(mean.r ,cov.r,risk.free = rf.m,shorts = TRUE)
tg.s.er=tangency.short$er #tangency return
tg.s.sd=tangency.short$sd
tg.s.w=tangency.short$weights
##SHORT SELL IS ALLOWED
##Efficient Portfolio Frontier
eff.front.short=efficient.frontier(mean.r ,cov.r,nport=50,
                                   shorts = TRUE)
ef.s.er=eff.front.short$er
ef.s.sd=eff.front.short$sd
ef.s.w=eff.front.short$weights
##Data Table
```

```
library(formattable)
short.table=data.frame("risk.free.day"=rbind(rf.m,0) ,
                       "mvp day"=rbind(mvp.s.sd,mvp.s.sd),
                       "tangency.day"=rbind(tg.s.er,tg.s.sd),
                       "mvp year"=rbind(mvp.s.er*365,mvp.s.sd*sqrt(365)),
                       "Tangency year"=rbind(tg.s.er*365,tg.s.sd*sqrt(365)),
                       "risk.free.year"=rbind(rf.m*365,0),
                       row.names = c("return","risk")
                       )
formattable(short.table)
##Value at Risk
s=100000
## t= one month
##Loss~N(-mean.mvp,sd.mvp), assuming mvp is normal
VaR.s.mvp=(-mvp.s.er+mvp.s.sd*qnorm(0.95))*s
VaR.s.mvp
VaR.s.tg=(-tg.s.er+tg.s.sd*qnorm(0.95))*s
VaR.s.tg
##Sharpe Ratio
sharpe.s = ( ef.s.er- rf.m) / ef.s.sd # compute Sharpe's ratios
(tg.s.sharpe=max(sharpe.s))
assets.s.sharpe=(mean.r -rf.m)/sd.r

sort(mvp.s.w,decreasing = T)

sort(tg.s.w,decreasing = T)

efficient.portfolio(scale(mean.r),cov.r,0.005,shorts=FALSE)

##Portfolio Plot
plot(ef.s.sd,ef.s.er,type="l",main="Efficient Frontier (Short)",
     xlab="daily Risk", ylab="daily Return",
     ylim = c(0,0.001),xlim=c(0,0.02),lty=3)

text(sqrt(cov.r[1,1]),mean.r[1],'PFE',cex=1.1)
text(sqrt(cov.r[2,2]),mean.r[2],'JNJ',cex=1.1)
text(sqrt(cov.r[3,3]),mean.r[3],'MRK',cex=1.1)
text(sqrt(cov.r[4,4]),mean.r[4],'NVS',cex=1.1)

points(0, rf.m, cex = 4, pch = "*") # show risk-free asset
sharpe = ( ef.s.er- rf.m) / ef.s.sd # compute Sharpe's ratios
ind = (sharpe == max(sharpe)) # Find maximum Sharpe's ratio
lines(c(0, 2), rf.m + c(0, 2) * (ef.s.er[ind] - rf.m) / ef.s.sd[ind], lwd = 4, lty = 1, col = "blue") #
points(ef.s.sd[ind], ef.s.er[ind], cex = 4, pch = "*") # tangency portfolio
ind2 = (ef.s.sd == min(ef.s.sd)) # find minimum variance portfolio
points(ef.s.sd[ind2], ef.s.er[ind2], cex = 2, pch = "+") # min var portfolio
ind3 = (ef.s.er > ef.s.er[ind2])
lines(ef.s.sd[ind3], ef.s.er[ind3], type = "l", xlim = c(0, 1),
      ylim = c(0, 0.3), lwd = 3, col = "red") # plot efficient frontier

# Copula
library(copula)
library(VineCopula)
```

```r
library(tidyverse)
## Gaussian, t, archimedean, clayton, gumbel
data = read.csv("full_close.csv")
cop.norm = normalCopula(dim=4)
fit.copnorm = fitCopula(cop.norm,pobs(data),method="ml")
fit.copnorm
AIC(fit.copnorm)
BIC(fit.copnorm)
logLik(fit.copnorm)
cop.t = tCopula(dim=4)
fit.copt = fitCopula(cop.t,pobs(data),method = "ml")
fit.copt
AIC(fit.copt)
BIC(fit.copt)
logLik(fit.copt)
cop.gumbel = gumbelCopula(dim=4)
fit.copgumbel = fitCopula(cop.gumbel,pobs(data),method = "ml")
fit.copgumbel
AIC(fit.copgumbel)
BIC(fit.copgumbel)
logLik(fit.copgumbel)
cop.archm = archmCopula(family = "frank",dim=4)
fit.archm = fitCopula(cop.archm, pobs(data),method = "ml")
fit.archm
AIC(fit.archm)
BIC(fit.archm)
logLik(fit.archm)
cop.clayton = claytonCopula(dim=4)
fit.copclayton = fitCopula(cop.clayton,pobs(data),method = "ml")
fit.copclayton
AIC(fit.copclayton)
BIC(fit.copclayton)
logLik(fit.copclayton)
```

```r
# Risk Management
## Normal distribution method
library(PerformanceAnalytics)
s = 100000
VaR.gaussian=sapply(return,function(data_5){-s*VaR(data_5, method="gaussian")})
VaR.gaussian
ES.gaussian=sapply(return,function(data_5){-s*ES(data_5, method="gaussian")})
ES.gaussian

## t distribution method
library(MASS)
info <- data.frame(matrix(ncol = 5, nrow = 2))

colnames(info) <- c('PFE_Close', 'JNJ_Close', 'ABBV_Close',
                  'MRK_Close', 'NVS_Close')

for (i in 1:5){
  alpha = 0.05
  fitt = fitdistr(return[,i], "t")
  param = as.numeric(fitt$estimate)
  mean = param[1]
  df = param[3]
  sd = param[2] * sqrt((df) / (df - 2))
  lambda = param[2]
  qalpha = qt(alpha, df = df)
  VaR_par = -100000 * (mean + lambda * qalpha)
  es1 = dt(qalpha, df = df) / (alpha)
  es2=(df+qalpha^2)/(df-1)
  es3=-mean+lambda*es1*es2
  ES_par = 100000*es3
  info[1,i] = VaR_par
  info[2,i] = ES_par
}

info

## Nonparametrix Method
VaR.nonparam=sapply(return,function(data_5){-s*VaR(data_5, method="historical")})
VaR.nonparam
ES.nonparam=sapply(return,function(data_5){-s*ES(data_5, method="historical")})
ES.nonparam
```