PROJECTPART1

HASAN KHAN

2022-11-08

#library(dplyr)

#install.packages("quantmod")

#install.packages("PerformanceAnalytics")

###PART1A##  
### Import Stock Prices of 3 companies, over a span of about 2 years. Estimate the 𝜇, 𝜎 of those stocks.  
### Need to check:  
### i) The log-returns fit into a Noise process (Normal; with Zero serial  
###correlation).  
### ii) Calculate 𝜇, 𝜎 (Follow Part C) ###

library(quantmod)

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library(PerformanceAnalytics)

##   
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':  
##   
## legend

tickers<- c("META","BABA","BAC")   
weights<- c(1/3,1/3,1/3) #equally weighted portfolio

#NOTES:  
## ln53: "[,4]" gets us close column

#Getting data for this portfolio  
PfPrices<- NULL  
# We will populate it with all the columns of data that we want from quantmod  
for(ticker in tickers){  
 PfPrices<-cbind(PfPrices,  
 getSymbols.yahoo(ticker,from="2020-12-31",to="2022-10-31",periodicity='daily',auto.assign=FALSE)[,4])  
}  
PfReturns<- na.omit(ROC(PfPrices))#ROC just calculates daily change in each individual column  
head(PfReturns)

## META.Close BABA.Close BAC.Close  
## 2021-01-04 -0.015569405 -0.021191422 -0.009280743  
## 2021-01-05 0.007519804 0.053616634 0.007629793  
## 2021-01-06 -0.028676080 -0.054670535 0.060585805  
## 2021-01-07 0.020412295 -0.003124277 0.021843630  
## 2021-01-08 -0.004363092 0.040127209 -0.010093433  
## 2021-01-11 -0.040927933 -0.038057950 0.016463817

#check to see if we have any missing values(especially important for machine learning models)  
colSums(is.na(PfPrices))

## META.Close BABA.Close BAC.Close   
## 0 0 0

#no missing data

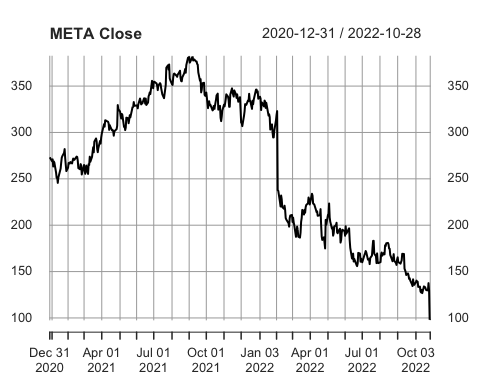
#creating benchmark using SP500  
bench<-getSymbols.yahoo("^GSPC",from="2020-12-31",to="2022-10-31",periodicity='daily',auto.assign=FALSE)[,4]  
benchReturns<- dailyReturn(bench)

#colSums(is.na(benchmarkPrices))

#Calculate Pf values at the end of the day  
#Aggregate the returns by using the ways we provided to output the return  
  
#PfReturn <- Return.portfolio(PfReturns)  
#head(PfReturn)

META.return<- dailyReturn(PfPrices$META.Close)  
BABA.return<- dailyReturn(PfPrices$BABA.Close)  
BAC.return<- dailyReturn(PfPrices$BAC.Close)

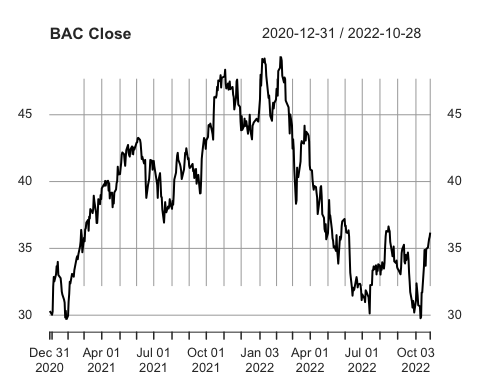
plot(PfPrices$META.Close,type="l",main="META Close")



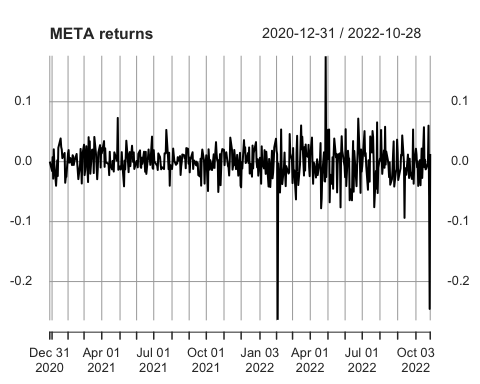
plot(PfPrices$BABA.Close,type="l",main="BABA Close")



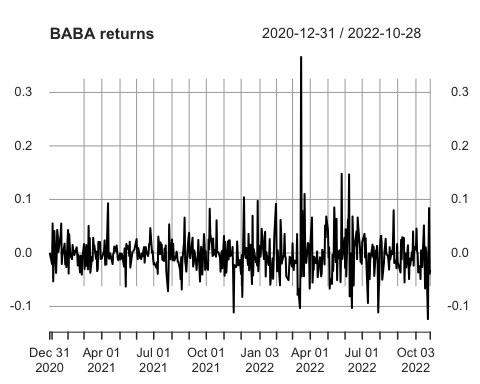
plot(PfPrices$BAC.Close,type="l",main="BAC Close")



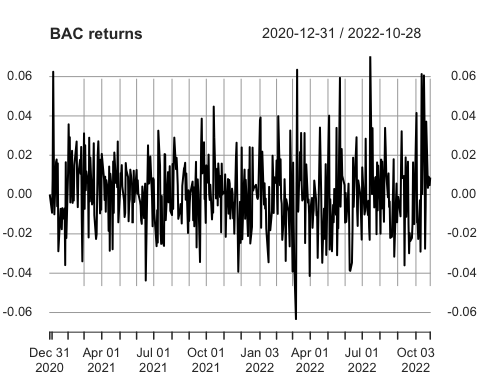
plot(META.return,type ="l",main="META returns")



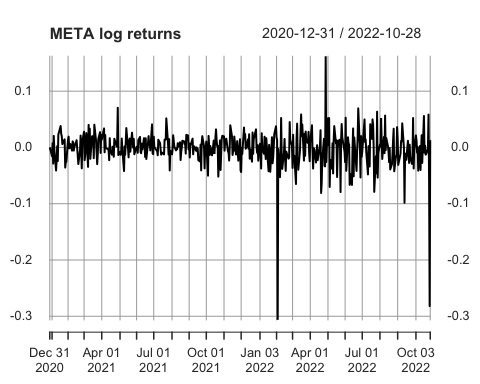
plot(BABA.return,type ="l",main="BABA returns")



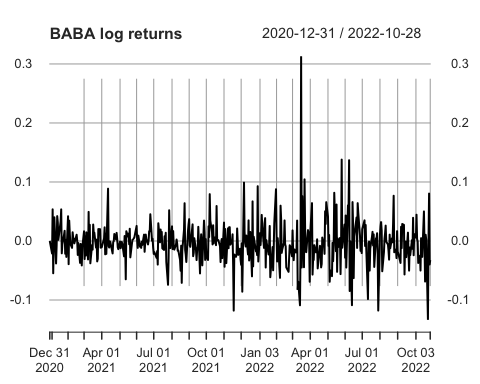
plot(BAC.return,type ="l",main="BAC returns")



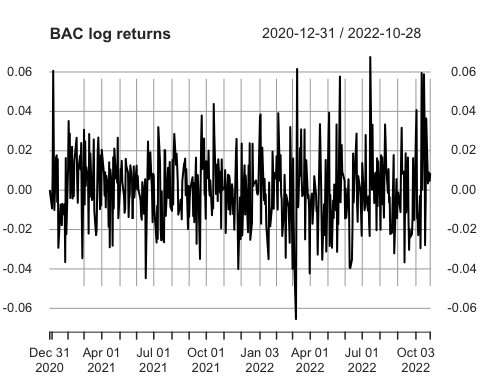
plot(log(1+META.return),type ="l",main="META log returns")



plot(log(1+BABA.return),type ="l",main="BABA log returns")



plot(log(1+BAC.return),type ="l",main="BAC log returns")



logreturn.META<-log(1+META.return)  
logreturn.BABA<-log(1+BABA.return)  
logreturn.BAC<-log(1+BAC.return)

#Estimation of mu and sigma from log returns  
#Calculate rbar  
  
rbarMETA= mean(logreturn.META$daily.returns)  
rbarBABA=mean(logreturn.BABA$daily.returns)  
rbarBAC = mean(logreturn.BAC$daily.returns)  
  
#Calculate s  
sMETA= sd(logreturn.META)  
sBABA= sd(logreturn.BABA)  
sBAC= sd(logreturn.BAC)  
  
#annualized sigma  
sigmaMETA <- sMETA \* sqrt(504)  
sigmaBABA <- sBABA \* sqrt(504)  
sigmaBAC<- sBAC \* sqrt(504)  
  
#annualized mu  
muMETA <- (rbarMETA\*504)+((sigmaMETA)^2)/2   
muBABA <- (rbarBABA\*504)+((sigmaBABA)^2)/2   
muBAC <- (rbarBAC\*504)+((sigmaBAC)^2)/2   
  
sigmaMETA

## [1] 0.7310755

sigmaBABA

## [1] 0.8341251

sigmaBAC

## [1] 0.4166854

muMETA

## [1] -0.8401646

muBABA

## [1] -1.067982

muBAC

## [1] 0.2803546

######################################  
#### A function to calculate mu, sigma   
######################################  
  
  
mu.sigma<- function(sample, lag=1){  
   
 N<-length(sample)  
 if (N < 1+lag){  
   
 stop("sample must be greater than 2 +lag")  
 }  
   
 ct <- sample[(1+lag):N]  
 pt<- sample[1: (N-lag)]  
 t=1  
 dt=t/N  
 returns <- (ct-pt)/pt  
   
 logreturns <- log(1+returns)  
 logreturns.bar <- mean(logreturns)  
   
 s <- sd(logreturns)  
   
   
 drift <- logreturns.bar\*N + s^2\*N/2  
   
 volatility <- sqrt(s^2\*N)  
   
 #cat("mu =", round(drift, 4) ,"sigma=",round(volatility,4) , "\n")  
   
 c(drift, volatility)  
   
}  
Meta.close <- data.frame(PfPrices$META)  
head(Meta.close)

## META.Close  
## 2020-12-31 273.16  
## 2021-01-04 268.94  
## 2021-01-05 270.97  
## 2021-01-06 263.31  
## 2021-01-07 268.74  
## 2021-01-08 267.57

dim(Meta.close)

## [1] 461 1

mu.sigma(Meta.close$META.Close,lag=1)

## [1] -0.7701558 0.6999513

Box.test(logreturn.META,lag = 365)

##   
## Box-Pierce test  
##   
## data: logreturn.META  
## X-squared = 217.92, df = 365, p-value = 1

Box.test(logreturn.BABA,lag = 365)

##   
## Box-Pierce test  
##   
## data: logreturn.BABA  
## X-squared = 217.23, df = 365, p-value = 1

Box.test(logreturn.BAC,lag = 365)

##   
## Box-Pierce test  
##   
## data: logreturn.BAC  
## X-squared = 213.16, df = 365, p-value = 1

#Calculating Beta  
beta.META<- cov(META.return,benchReturns)/var(benchReturns)  
beta.META # >1 therefore performed atleast 1.6 times better than the market

## daily.returns  
## daily.returns 1.642219

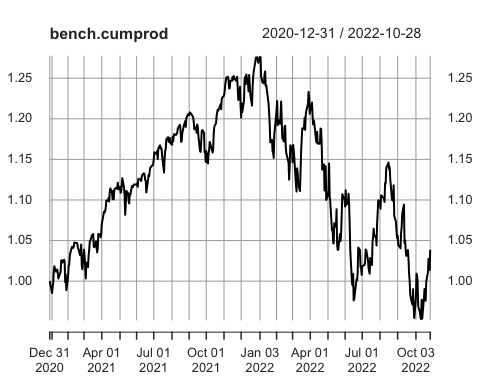
beta.BABA<- cov(BABA.return,benchReturns)/var(benchReturns)  
beta.BABA # >1 therefore performed atleast 1.1 times better the market

## daily.returns  
## daily.returns 1.147015

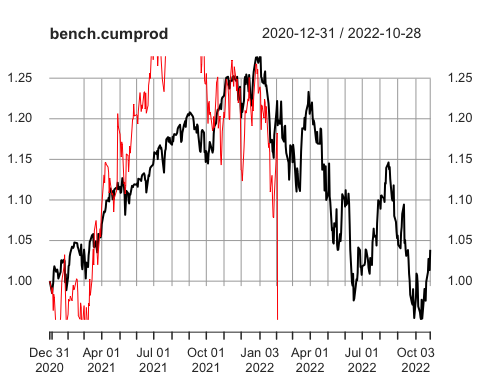
beta.BAC<- cov(BAC.return,benchReturns)/var(benchReturns)  
beta.BAC # <1 therefore did not perform better than the market although its almost 1

## daily.returns  
## daily.returns 0.9772168

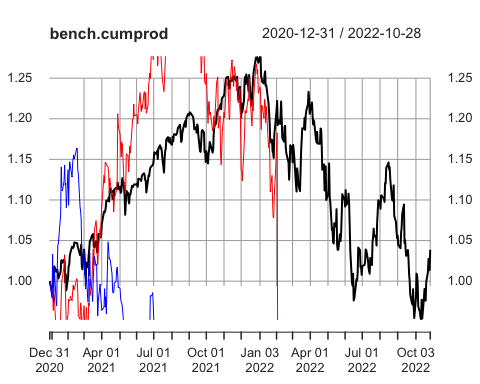
#Plot the cumulative product of (1+ returns) to compare stocks  
  
META.cumprod <- cumprod(1+ META.return)  
BABA.cumprod <- cumprod(1+ BABA.return)  
BAC.cumprod <- cumprod(1+BAC.return)  
  
bench.cumprod <- cumprod(1+ benchReturns)  
plot(bench.cumprod, type="l")



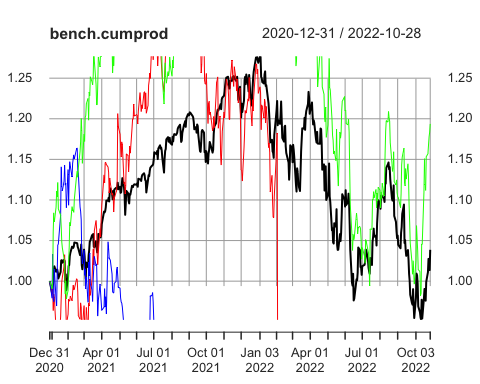
lines(META.cumprod, col="red")



lines(BABA.cumprod, col="blue")



lines(BAC.cumprod, col="green")



#EXPECTED RETURN  
returns<- cbind(META.return,BABA.return,BAC.return)  
mean.returns<- apply(returns,2,mean)  
mean.returns

## daily.returns daily.returns.1 daily.returns.2   
## -0.0016823886 -0.0021050641 0.0005563398

ff<-read.csv("FamaFrench.csv", header=T, skip=4)  
#dim(ff)

colnames(ff) <- c("date", "MkRF","SMB", "HML", "RF")  
head(ff)

## date MkRF SMB HML RF  
## 1 19260708 0.21 -0.38 0.19 0.009  
## 2 19260709 -0.71 0.43 0.57 0.009  
## 3 19260710 0.62 -0.53 -0.10 0.009  
## 4 19260712 0.04 -0.03 0.64 0.009  
## 5 19260713 0.48 -0.28 -0.20 0.009  
## 6 19260714 0.04 0.07 -0.43 0.009

ff$date <- as.Date(ff$date,"%Y%m%d")  
head(ff)

## date MkRF SMB HML RF  
## 1 1926-07-08 0.21 -0.38 0.19 0.009  
## 2 1926-07-09 -0.71 0.43 0.57 0.009  
## 3 1926-07-10 0.62 -0.53 -0.10 0.009  
## 4 1926-07-12 0.04 -0.03 0.64 0.009  
## 5 1926-07-13 0.48 -0.28 -0.20 0.009  
## 6 1926-07-14 0.04 0.07 -0.43 0.009

ff.new<- ff[ff$date >="2020-12-31"& ff$date<="2022-09-30",]  
head(ff.new)

## date MkRF SMB HML RF  
## 24892 2020-12-31 0.39 -0.89 0.41 0  
## 24893 2021-01-04 -1.41 0.22 0.58 0  
## 24894 2021-01-05 0.86 1.23 0.48 0  
## 24895 2021-01-06 0.79 2.14 3.93 0  
## 24896 2021-01-07 1.76 0.33 -0.83 0  
## 24897 2021-01-08 0.51 -0.75 -1.38 0

#Convert the return into dfs  
META.return.df<- data.frame(META.return)  
BABA.return.df<- data.frame(BABA.return)  
BAC.return.df<- data.frame(BAC.return)  
  
#add date column to later merge on  
  
META.return.df$date <- index(META.return)  
BABA.return.df$date <- index(BABA.return)  
BAC.return.df$date <- index(BAC.return)  
  
#merge  
META.1 <- merge(ff.new, META.return.df, by="date", all.x=TRUE)  
BABA.1 <- merge(ff.new, BABA.return.df, by="date", all.x=TRUE)  
BAC.1 <- merge(ff.new, BAC.return.df, by="date", all.x=TRUE)  
  
head(META.1)

## date MkRF SMB HML RF daily.returns  
## 1 2020-12-31 0.39 -0.89 0.41 0 0.000000000  
## 2 2021-01-04 -1.41 0.22 0.58 0 -0.015448828  
## 3 2021-01-05 0.86 1.23 0.48 0 0.007548148  
## 4 2021-01-06 0.79 2.14 3.93 0 -0.028268823  
## 5 2021-01-07 1.76 0.33 -0.83 0 0.020622050  
## 6 2021-01-08 0.51 -0.75 -1.38 0 -0.004353587

attach(META.1)  
META.mod<-lm(daily.returns - RF~ MkRF + SMB+HML, META.1)  
summary(META.mod)

##   
## Call:  
## lm(formula = daily.returns - RF ~ MkRF + SMB + HML, data = META.1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.215081 -0.008646 0.000268 0.010700 0.138549   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.0018500 0.0009885 -1.872 0.0619 .   
## MkRF 0.0146452 0.0009092 16.108 < 2e-16 \*\*\*  
## SMB -0.0018200 0.0013281 -1.370 0.1713   
## HML -0.0048597 0.0008979 -5.413 1.03e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.0207 on 437 degrees of freedom  
## (2 observations deleted due to missingness)  
## Multiple R-squared: 0.5043, Adjusted R-squared: 0.5009   
## F-statistic: 148.2 on 3 and 437 DF, p-value: < 2.2e-16

#some general analysis notes  
 #t value high magnitude, the coefficient is going to be statistically significant.  
# R-squared gives us a measurement of what percent of the variance in the response variable can be explained by the regression  
#y-variable response variable  
# mult R sq typically increases each time you add a predictor variable(x)  
# mult R sq amount of variation in the response variable explained by the predictor variable   
#adj R sq controls for each additional predictor added(to prevent from overfitting) so it may not increase as you add more variables  
#F-statistic indicates if the model as a whole is statistically significant(number further from 1 is better)  
# p-value <.05 indicates this model is statistically sign