Final Exam

Akhil Gupta

December 17, 2017

library(quantmod)

## Warning: package 'quantmod' was built under R version 3.4.1

## Loading required package: xts

## Warning: package 'xts' was built under R version 3.4.1

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.4.1

##   
## Attaching package: 'zoo'

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

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 3.4.1

## Version 0.4-0 included new data defaults. See ?getSymbols.

library(PerformanceAnalytics)

## Warning: package 'PerformanceAnalytics' was built under R version 3.4.2

##   
## Attaching package: 'PerformanceAnalytics'

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

library(quadprog)

## Warning: package 'quadprog' was built under R version 3.4.1

library(xts)  
library(zoo)

# Step1: Create Portfolio of 5 companies

#Microsoft  
data.MSFT <- getSymbols("MSFT", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)

## 'getSymbols' currently uses auto.assign=TRUE by default, but will  
## use auto.assign=FALSE in 0.5-0. You will still be able to use  
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")  
## and getOption("getSymbols.auto.assign") will still be checked for  
## alternate defaults.  
##   
## This message is shown once per session and may be disabled by setting   
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

##   
## WARNING: There have been significant changes to Yahoo Finance data.  
## Please see the Warning section of '?getSymbols.yahoo' for details.  
##   
## This message is shown once per session and may be disabled by setting  
## options("getSymbols.yahoo.warning"=FALSE).

data.MSFT <- to.monthly(data.MSFT)  
ret.MSFT <- Return.calculate(data.MSFT$data.MSFT.Adjusted)  
  
#Tesla  
data.TSLA <- getSymbols("TSLA", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)  
data.TSLA <- to.monthly(data.TSLA)  
ret.TSLA <- Return.calculate(data.TSLA$data.TSLA.Adjusted)  
  
#Google  
data.GOOGL <- getSymbols("GOOGL", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)  
data.GOOGL <- to.monthly(data.GOOGL)  
ret.GOOGL <- Return.calculate(data.GOOGL$data.GOOGL.Adjusted)  
  
#Texas Instruments  
data.TXN <- getSymbols("TXN", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)  
data.TXN <- to.monthly(data.TXN)  
ret.TXN <- Return.calculate(data.TXN$data.TXN.Adjusted)  
  
# Intutive Surgical  
data.ISRG <- getSymbols("ISRG", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)  
data.ISRG <- to.monthly(data.ISRG)  
ret.ISRG <- Return.calculate(data.ISRG$data.ISRG.Adjusted)  
  
  
## Calculate Monthly Return  
  
Ret.monthly <- cbind(ret.MSFT, ret.TSLA, ret.GOOGL, ret.TXN, ret.ISRG)  
Ret.monthly <- Ret.monthly[-1,]  
head(Ret.monthly)

## data.MSFT.Adjusted data.TSLA.Adjusted data.GOOGL.Adjusted  
## Jan 2013 0.02770493 0.10746971 0.068294326  
## Feb 2013 0.02113535 -0.07144751 0.060223087  
## Mar 2013 0.02913666 0.08785521 -0.008749389  
## Apr 2013 0.15693827 0.42491432 0.038252798  
## May 2013 0.06177410 0.81070566 0.056574966  
## Jun 2013 -0.01031525 0.09819966 0.010502537  
## data.TXN.Adjusted data.ISRG.Adjusted  
## Jan 2013 0.077746977 0.171319671  
## Feb 2013 0.040810022 -0.112277620  
## Mar 2013 0.030496380 -0.036674599  
## Apr 2013 0.028550793 0.002239467  
## May 2013 -0.008285243 0.010644177  
## Jun 2013 -0.029518119 0.017285403

mat.ret<-matrix(Ret.monthly,nrow(Ret.monthly))  
colnames(mat.ret)<-c("MSFT.Ret", "TSLA.Ret", "GOOGL.Ret", "TXN.Ret", "ISRG.Ret")  
head(mat.ret)

## MSFT.Ret TSLA.Ret GOOGL.Ret TXN.Ret ISRG.Ret  
## [1,] 0.02770493 0.10746971 0.068294326 0.077746977 0.171319671  
## [2,] 0.02113535 -0.07144751 0.060223087 0.040810022 -0.112277620  
## [3,] 0.02913666 0.08785521 -0.008749389 0.030496380 -0.036674599  
## [4,] 0.15693827 0.42491432 0.038252798 0.028550793 0.002239467  
## [5,] 0.06177410 0.81070566 0.056574966 -0.008285243 0.010644177  
## [6,] -0.01031525 0.09819966 0.010502537 -0.029518119 0.017285403

# Step 2: Calculate Variance-Covariance(VCOV) Matrix of Returns  
VCOV<-cov(mat.ret)  
VCOV

## MSFT.Ret TSLA.Ret GOOGL.Ret TXN.Ret  
## MSFT.Ret 0.004607195 0.0015684907 0.0012743542 0.0008201260  
## TSLA.Ret 0.001568491 0.0382006758 -0.0003301925 -0.0006336288  
## GOOGL.Ret 0.001274354 -0.0003301925 0.0036859695 0.0007479949  
## TXN.Ret 0.000820126 -0.0006336288 0.0007479949 0.0026505640  
## ISRG.Ret 0.001705845 0.0004380486 0.0014292895 -0.0001240979  
## ISRG.Ret  
## MSFT.Ret 0.0017058446  
## TSLA.Ret 0.0004380486  
## GOOGL.Ret 0.0014292895  
## TXN.Ret -0.0001240979  
## ISRG.Ret 0.0061470396

# Step 3: Construct the Target Portfolio Return Vector  
  
avg.ret<-matrix(apply(mat.ret,2,mean))  
colnames(avg.ret)<-paste("Avg.Ret")  
rownames(avg.ret)<-paste(c("MSFT","TSLA","GOOGL","TXN","ISRG"))  
avg.ret

## Avg.Ret  
## MSFT 0.025466509  
## TSLA 0.070950615  
## GOOGL 0.024225543  
## TXN 0.020285797  
## ISRG 0.006455963

min.ret<-min(avg.ret)  
min.ret

## [1] 0.006455963

max.ret<-max(avg.ret)  
max.ret

## [1] 0.07095061

increments=100  
tgt.ret<-seq(min.ret,max.ret,length=increments)  
head(tgt.ret)

## [1] 0.006455963 0.007107424 0.007758886 0.008410347 0.009061808 0.009713269

tail(tgt.ret)

## [1] 0.06769331 0.06834477 0.06899623 0.06964769 0.07029915 0.07095061

# Step 4: Construct Dummy Portfolio Standard DeviationVector  
tgt.sd<-rep(0,length=increments)  
tgt.sd

## [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

#Step 5: Construct Dummy Portfolio Weights Vector  
wgt<-matrix(0,nrow=increments,ncol=length(avg.ret))  
head(wgt)

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0 0 0 0 0  
## [2,] 0 0 0 0 0  
## [3,] 0 0 0 0 0  
## [4,] 0 0 0 0 0  
## [5,] 0 0 0 0 0  
## [6,] 0 0 0 0 0

# Step 6: Run the quadprog Optimizer  
  
for (i in 1:increments){  
 Dmat<-2\*VCOV  
 dvec<-c(rep(0,length(avg.ret)))  
 Amat<-cbind(rep(1,length(avg.ret)),avg.ret,  
 diag(1,nrow=ncol(Ret.monthly)))  
 bvec<-c(1,tgt.ret[i],rep(0,ncol(Ret.monthly)))  
 soln<-solve.QP(Dmat,dvec,Amat,bvec=bvec,meq=2)  
 tgt.sd[i]<-sqrt(soln$value)  
 wgt[i,]<-soln$solution  
}  
  
colnames(wgt)<-paste(c("wgt.MSFT", "wgt.TSLA", "wgt.GOOGL", "wgt.TXN", "wgt.ISRG"))  
wgt[1,2]<-0  
wgt[nrow(wgt),1]<-0  
head(wgt)

## wgt.MSFT wgt.TSLA wgt.GOOGL wgt.TXN wgt.ISRG  
## [1,] -1.207517e-17 0.000000e+00 0 -4.440892e-16 1.0000000  
## [2,] -1.104773e-17 1.357134e-17 0 4.710549e-02 0.9528945  
## [3,] -1.002028e-17 -2.773866e-19 0 9.421098e-02 0.9057890  
## [4,] 1.876274e-17 -2.483217e-19 0 1.413165e-01 0.8586835  
## [5,] 1.979019e-17 -2.192567e-19 0 1.884220e-01 0.8115780  
## [6,] 2.081764e-17 -1.901918e-19 0 2.355275e-01 0.7644725

# Step 7: Combine Portfolio Returns, Portfolio Standard Deviations, and Portfolio Weights  
tgt.port<-data.frame(cbind(tgt.ret,tgt.sd,wgt))  
head(tgt.port)

## tgt.ret tgt.sd wgt.MSFT wgt.TSLA wgt.GOOGL  
## 1 0.006455963 0.07840306 -1.207517e-17 0.000000e+00 0  
## 2 0.007107424 0.07467464 -1.104773e-17 1.357134e-17 0  
## 3 0.007758886 0.07103314 -1.002028e-17 -2.773866e-19 0  
## 4 0.008410347 0.06749265 1.876274e-17 -2.483217e-19 0  
## 5 0.009061808 0.06406989 1.979019e-17 -2.192567e-19 0  
## 6 0.009713269 0.06078477 2.081764e-17 -1.901918e-19 0  
## wgt.TXN wgt.ISRG  
## 1 -4.440892e-16 1.0000000  
## 2 4.710549e-02 0.9528945  
## 3 9.421098e-02 0.9057890  
## 4 1.413165e-01 0.8586835  
## 5 1.884220e-01 0.8115780  
## 6 2.355275e-01 0.7644725

# Step 8: Identify the Minimum Variance Portfolio  
minvar.port<-subset(tgt.port,tgt.port$tgt.sd==min(tgt.port$tgt.sd))  
minvar.port

## tgt.ret tgt.sd wgt.MSFT wgt.TSLA wgt.GOOGL wgt.TXN  
## 24 0.02143957 0.0383308 0.09933573 0.04291118 0.2046812 0.4837758  
## wgt.ISRG  
## 24 0.1692961

# Step 9: Identify the Tangency Portfolio  
riskfree = 0.0000583  
tgt.port$Sharpe<-(tgt.port$tgt.ret-riskfree)/tgt.port$tgt.sd  
head(tgt.port)

## tgt.ret tgt.sd wgt.MSFT wgt.TSLA wgt.GOOGL  
## 1 0.006455963 0.07840306 -1.207517e-17 0.000000e+00 0  
## 2 0.007107424 0.07467464 -1.104773e-17 1.357134e-17 0  
## 3 0.007758886 0.07103314 -1.002028e-17 -2.773866e-19 0  
## 4 0.008410347 0.06749265 1.876274e-17 -2.483217e-19 0  
## 5 0.009061808 0.06406989 1.979019e-17 -2.192567e-19 0  
## 6 0.009713269 0.06078477 2.081764e-17 -1.901918e-19 0  
## wgt.TXN wgt.ISRG Sharpe  
## 1 -4.440892e-16 1.0000000 0.08159966  
## 2 4.710549e-02 0.9528945 0.09439784  
## 3 9.421098e-02 0.9057890 0.10840835  
## 4 1.413165e-01 0.8586835 0.12374751  
## 5 1.884220e-01 0.8115780 0.14052635  
## 6 2.355275e-01 0.7644725 0.15883861

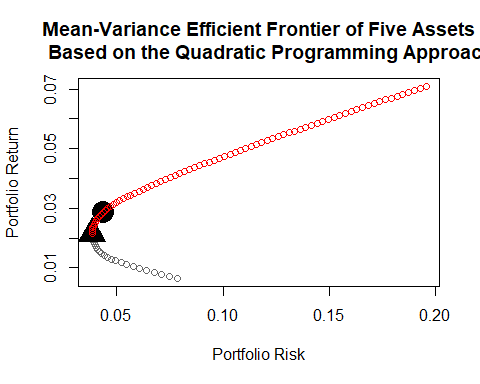
tangency.port<-subset(tgt.port,tgt.port$Sharpe==max(tgt.port$Sharpe))  
tangency.port

## tgt.ret tgt.sd wgt.MSFT wgt.TSLA wgt.GOOGL wgt.TXN  
## 35 0.02860564 0.04346084 0.1676788 0.1231472 0.3076133 0.4015607  
## wgt.ISRG Sharpe  
## 35 3.469447e-18 0.656852

# Step 10: Identify Efficient Portfolios  
eff.frontier<-subset(tgt.port,tgt.port$tgt.ret>=minvar.port$tgt.ret)  
head(eff.frontier)

## tgt.ret tgt.sd wgt.MSFT wgt.TSLA wgt.GOOGL wgt.TXN  
## 24 0.02143957 0.03833080 0.09933573 0.04291118 0.2046812 0.4837758  
## 25 0.02209103 0.03838341 0.10646381 0.04948616 0.2148642 0.4773371  
## 26 0.02274249 0.03852318 0.11359189 0.05606115 0.2250472 0.4708984  
## 27 0.02339395 0.03874915 0.12071996 0.06263613 0.2352302 0.4644597  
## 28 0.02404541 0.03905984 0.12784804 0.06921112 0.2454132 0.4580210  
## 29 0.02469687 0.03945324 0.13497611 0.07578610 0.2555961 0.4515823  
## wgt.ISRG Sharpe  
## 24 0.16929610 0.5578092  
## 25 0.15184876 0.5740170  
## 26 0.13440142 0.5888453  
## 27 0.11695407 0.6022236  
## 28 0.09950673 0.6141119  
## 29 0.08205938 0.6245006

# Step 11: Plot theMVEfficient Frontier  
plot(x=tgt.sd, xlab="Portfolio Risk", y=tgt.ret, ylab="Portfolio Return", col="gray40",  
 main="Mean-Variance Efficient Frontier of Five Assets  
 Based on the Quadratic Programming Approach")  
abline(h=0,lty=1)  
points(x=minvar.port$tgt.sd,y=minvar.port$tgt.ret,pch=17,cex=3)  
points(x=tangency.port$tgt.sd,y=tangency.port$tgt.ret,pch=19,cex=3)  
points(x=eff.frontier$tgt.sd,y=eff.frontier$tgt.ret, col ="red" )



# Portfolio Return

portfolio <- Ret.monthly  
head(portfolio)

## data.MSFT.Adjusted data.TSLA.Adjusted data.GOOGL.Adjusted  
## Jan 2013 0.02770493 0.10746971 0.068294326  
## Feb 2013 0.02113535 -0.07144751 0.060223087  
## Mar 2013 0.02913666 0.08785521 -0.008749389  
## Apr 2013 0.15693827 0.42491432 0.038252798  
## May 2013 0.06177410 0.81070566 0.056574966  
## Jun 2013 -0.01031525 0.09819966 0.010502537  
## data.TXN.Adjusted data.ISRG.Adjusted  
## Jan 2013 0.077746977 0.171319671  
## Feb 2013 0.040810022 -0.112277620  
## Mar 2013 0.030496380 -0.036674599  
## Apr 2013 0.028550793 0.002239467  
## May 2013 -0.008285243 0.010644177  
## Jun 2013 -0.029518119 0.017285403

port.ret <- Return.portfolio(portfolio, weights = c(0.09933573, 0.04291118,0.2046812,0.4837758,0.1692961), rebalance\_on = "quarters")  
head(port.ret)

## portfolio.returns  
## Jan 2013 0.0879581738  
## Feb 2013 0.0100590784  
## Mar 2013 0.0136310206  
## Apr 2013 0.0559940975  
## May 2013 0.0626252201  
## Jun 2013 -0.0003342324

# Cumulative Return  
Return.cumulative(port.ret)

## portfolio.returns  
## Cumulative Return 1.131002

# Annualized standard deviation

returns <- cbind(ret.MSFT, ret.TSLA, ret.GOOGL, ret.TXN, ret.ISRG)  
returns <- returns[-1,]  
names(returns) <- c("MSFT.ret", "TSLA.ret", "GOOGL.ret", "TXN.ret", "ISRG.ret")  
head(returns)

## MSFT.ret TSLA.ret GOOGL.ret TXN.ret ISRG.ret  
## Jan 2013 0.02770493 0.10746971 0.068294326 0.077746977 0.171319671  
## Feb 2013 0.02113535 -0.07144751 0.060223087 0.040810022 -0.112277620  
## Mar 2013 0.02913666 0.08785521 -0.008749389 0.030496380 -0.036674599  
## Apr 2013 0.15693827 0.42491432 0.038252798 0.028550793 0.002239467  
## May 2013 0.06177410 0.81070566 0.056574966 -0.008285243 0.010644177  
## Jun 2013 -0.01031525 0.09819966 0.010502537 -0.029518119 0.017285403

#Weight and Transpose weight matrix  
WGT.asset<-c(0.09933573, 0.04291118,0.2046812,0.4837758,0.1692961)  
WGT.asset<-matrix(WGT.asset,1)  
WGT.asset

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.09933573 0.04291118 0.2046812 0.4837758 0.1692961

tWGT.asset<-t(WGT.asset)  
tWGT.asset

## [,1]  
## [1,] 0.09933573  
## [2,] 0.04291118  
## [3,] 0.20468120  
## [4,] 0.48377580  
## [5,] 0.16929610

#Constructing Variance-Covariance Matrix  
mat.Ret<-as.matrix(returns)  
VCOV.asset<-cov(mat.Ret)\*252  
VCOV.asset

## MSFT.ret TSLA.ret GOOGL.ret TXN.ret ISRG.ret  
## MSFT.ret 1.1610131 0.39525966 0.32113725 0.20667175 0.42987284  
## TSLA.ret 0.3952597 9.62657031 -0.08320851 -0.15967445 0.11038825  
## GOOGL.ret 0.3211372 -0.08320851 0.92886431 0.18849472 0.36018095  
## TXN.ret 0.2066718 -0.15967445 0.18849472 0.66794212 -0.03127268  
## ISRG.ret 0.4298728 0.11038825 0.36018095 -0.03127268 1.54905397

#Portfolio Risk  
mat.varasset<-WGT.asset %\*% VCOV.asset %\*% tWGT.asset  
mat.sdasset<-sqrt(mat.varasset)  
mat.sdasset #multi asset annualized std dev

## [,1]  
## [1,] 0.6084825

# Historical VaR

head(returns)

## MSFT.ret TSLA.ret GOOGL.ret TXN.ret ISRG.ret  
## Jan 2013 0.02770493 0.10746971 0.068294326 0.077746977 0.171319671  
## Feb 2013 0.02113535 -0.07144751 0.060223087 0.040810022 -0.112277620  
## Mar 2013 0.02913666 0.08785521 -0.008749389 0.030496380 -0.036674599  
## Apr 2013 0.15693827 0.42491432 0.038252798 0.028550793 0.002239467  
## May 2013 0.06177410 0.81070566 0.056574966 -0.008285243 0.010644177  
## Jun 2013 -0.01031525 0.09819966 0.010502537 -0.029518119 0.017285403

#Calculating the current asset value  
ret.cum.MSFT <- Return.cumulative(returns$MSFT.ret)  
ret.cum.MSFT

## MSFT.ret  
## Cumulative Return 1.292215

ret.cum.TSLA <- Return.cumulative(returns$TSLA.ret)  
ret.cum.TSLA

## TSLA.ret  
## Cumulative Return 6.029525

ret.cum.GOOGL <- Return.cumulative(returns$GOOGL.ret)  
ret.cum.GOOGL

## GOOGL.ret  
## Cumulative Return 1.232208

ret.cum.TXN <- Return.cumulative(returns$TXN.ret)  
ret.cum.TXN

## TXN.ret  
## Cumulative Return 0.9720234

ret.cum.ISRG <- Return.cumulative(returns$ISRG.ret)  
ret.cum.ISRG

## ISRG.ret  
## Cumulative Return 0.1267206

val\_investent <- WGT.asset\*1000000 #If we had 1 million, this is the distribution of investment according to effective frontier  
val\_investent

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 99335.73 42911.18 204681.2 483775.8 169296.1

MSFT.val <- val\_investent[,1] \* ( 1 + ret.cum.MSFT)  
MSFT.val

## MSFT.ret  
## Cumulative Return 227698.8

TSLA.val <- val\_investent[,2] \* ( 1 + ret.cum.TSLA)  
TSLA.val

## TSLA.ret  
## Cumulative Return 301645.2

GOOGL.val <- val\_investent[,3] \* ( 1 + ret.cum.GOOGL)  
GOOGL.val

## GOOGL.ret  
## Cumulative Return 456891

TXN.val <- val\_investent[,4] \* ( 1 + ret.cum.TXN)  
TXN.val

## TXN.ret  
## Cumulative Return 954017.2

ISRG.val <- val\_investent[,5] \* ( 1 + ret.cum.ISRG)  
ISRG.val

## ISRG.ret  
## Cumulative Return 190749.4

last.idx <- c(MSFT.val, TSLA.val, GOOGL.val, TXN.val, ISRG.val)  
sum(last.idx)

## [1] 2131002

#Calculated simulated return  
sim.portPnL <- last.idx[1] \* returns$MSFT.ret + last.idx[2] \* returns$TSLA.ret + last.idx[3] \* returns$GOOGL.ret + last.idx[4] \* returns$TXN.ret + last.idx[5] \* returns$ISRG.ret  
names(sim.portPnL) <- "Port.PnL"  
head(sim.portPnL)

## Port.PnL  
## Jan 2013 176780.246  
## Feb 2013 28292.658  
## Mar 2013 51236.379  
## Apr 2013 209050.508  
## May 2013 278586.063  
## Jun 2013 7207.589

# Historical VaR at 1% and 5%

VaR01.Historical=quantile(-sim.portPnL$Port.PnL,0.99)  
VaR01.Historical

## 99%   
## 87493.66

VaR05.Historical=quantile(-sim.portPnL$Port.PnL,0.95)  
VaR05.Historical

## 95%   
## 75999.14

# Historical Excess Shortfall

#Identify Simulated Portfolio Losses in Excess of VaR  
ES.PnL <-sim.portPnL$Port.PnL  
ES.PnL$dummy01<-ifelse(ES.PnL$Port.PnL< (- VaR01.Historical) ,1,0)  
ES.PnL$dummy05<-ifelse(ES.PnL$Port.PnL< (-VaR05.Historical) ,1,0)  
head(ES.PnL)

## Port.PnL dummy01 dummy05  
## Jan 2013 176780.246 0 0  
## Feb 2013 28292.658 0 0  
## Mar 2013 51236.379 0 0  
## Apr 2013 209050.508 0 0  
## May 2013 278586.063 0 0  
## Jun 2013 7207.589 0 0

#Extract Portfolio Losses in Excess of VaR and Compute Average of Losses in Excess of VaR  
shortfall01<-subset(ES.PnL,ES.PnL$dummy01==1)  
shortfall05<-subset(ES.PnL,ES.PnL$dummy05==1)  
ES01.Historical<- -mean(shortfall01$Port.PnL)  
ES01.Historical

## [1] 88999.18

ES05.Historical<- -mean(shortfall05$Port.PnL)  
ES05.Historical

## [1] 86848.44

# CAPM

head(port.ret)

## portfolio.returns  
## Jan 2013 0.0879581738  
## Feb 2013 0.0100590784  
## Mar 2013 0.0136310206  
## Apr 2013 0.0559940975  
## May 2013 0.0626252201  
## Jun 2013 -0.0003342324

port.csv <- cbind(index(port.ret), data.frame(port.ret))  
head(port.csv)

## index(port.ret) portfolio.returns  
## Jan 2013 Jan 2013 0.0879581738  
## Feb 2013 Feb 2013 0.0100590784  
## Mar 2013 Mar 2013 0.0136310206  
## Apr 2013 Apr 2013 0.0559940975  
## May 2013 May 2013 0.0626252201  
## Jun 2013 Jun 2013 -0.0003342324

port<-port.csv  
names(port)<-c("date", "port.ret")  
head(port)

## date port.ret  
## Jan 2013 Jan 2013 0.0879581738  
## Feb 2013 Feb 2013 0.0100590784  
## Mar 2013 Mar 2013 0.0136310206  
## Apr 2013 Apr 2013 0.0559940975  
## May 2013 May 2013 0.0626252201  
## Jun 2013 Jun 2013 -0.0003342324

nrow(port)

## [1] 36

#Step 2:LOAD MARKET DATA   
  
data.GSPC <- getSymbols("^GSPC", from = "2012-12-31", to = "2015-12-31", auto.assign = FALSE)  
data.GSPC <- to.monthly(data.GSPC)  
mkt.ret <- Return.calculate(data.GSPC$data.GSPC.Adjusted)  
mkt.ret <- mkt.ret[-1,]  
head(mkt.ret)

## data.GSPC.Adjusted  
## Jan 2013 0.05042810  
## Feb 2013 0.01106065  
## Mar 2013 0.03598772  
## Apr 2013 0.01808577  
## May 2013 0.02076281  
## Jun 2013 -0.01499930

nrow(mkt.ret)

## [1] 36

#Step 3: Load Risk Free return  
getSymbols("DGS3MO",src = 'FRED', return.class = "xts")

## [1] "DGS3MO"

rf <- DGS3MO  
rf[1:5,]

## DGS3MO  
## 1982-01-04 11.87  
## 1982-01-05 12.20  
## 1982-01-06 12.16  
## 1982-01-07 12.17  
## 1982-01-08 11.98

tail(rf)

## DGS3MO  
## 2017-12-07 1.29  
## 2017-12-08 1.28  
## 2017-12-11 1.33  
## 2017-12-12 1.34  
## 2017-12-13 1.30  
## 2017-12-14 1.32

#Apply to.monthly Command to Identify FirstYield for Each Month  
rf.monthly<-to.monthly(rf)

## Warning in to.period(x, "months", indexAt = indexAt, name = name, ...):  
## missing values removed from data

rf.monthly[1:3,]

## rf.Open rf.High rf.Low rf.Close  
## Jan 1982 11.87 14.06 11.87 13.08  
## Feb 1982 14.77 15.49 12.79 13.00  
## Mar 1982 12.81 14.16 12.68 13.99

#Convert Opening Annualized Yield for Each Month Into a Monthly Yield  
options(scipen="100")  
rf.monthly<-(1+rf.monthly[,1]/100)^(1/12)-1  
rf.monthly[c(1:3,nrow(rf.monthly)),]

## rf.Open  
## Jan 1982 0.009391097  
## Feb 1982 0.011546143  
## Mar 1982 0.010095183  
## Dec 2017 0.001052222

#Subset Data to January 2011 Through December 2016  
rf.sub<-subset(rf.monthly, index(rf.monthly) >= as.yearmon("Jan 2013") & index(rf.monthly) <= as.yearmon("Dec 2015"))  
rf.sub[c(1:3,nrow(rf.sub)),]

## rf.Open  
## Jan 2013 0.00006664223  
## Feb 2013 0.00004998626  
## Mar 2013 0.00009162048  
## Dec 2015 0.00017483179

nrow(rf.sub)

## [1] 36

# Step 4: Combine all the returns   
combo <- cbind(data.frame(mkt.ret),data.frame(rf.sub), port$port.ret)  
names(combo)<-paste(c("mkt.ret","rf","port.ret"))  
head(combo)

## mkt.ret rf port.ret  
## Jan 2013 0.05042810 0.00006664223 0.0879581738  
## Feb 2013 0.01106065 0.00004998626 0.0100590784  
## Mar 2013 0.03598772 0.00009162048 0.0136310206  
## Apr 2013 0.01808577 0.00006664223 0.0559940975  
## May 2013 0.02076281 0.00004998626 0.0626252201  
## Jun 2013 -0.01499930 0.00004165712 -0.0003342324

# Step 5: Calculate excess portfolio and market return   
combo$exret<-combo$port.ret - combo$rf  
combo$exmkt<-combo$mkt.ret - combo$rf  
head(combo)

## mkt.ret rf port.ret exret exmkt  
## Jan 2013 0.05042810 0.00006664223 0.0879581738 0.0878915316 0.05036145  
## Feb 2013 0.01106065 0.00004998626 0.0100590784 0.0100090922 0.01101066  
## Mar 2013 0.03598772 0.00009162048 0.0136310206 0.0135394001 0.03589610  
## Apr 2013 0.01808577 0.00006664223 0.0559940975 0.0559274552 0.01801913  
## May 2013 0.02076281 0.00004998626 0.0626252201 0.0625752338 0.02071283  
## Jun 2013 -0.01499930 0.00004165712 -0.0003342324 -0.0003758895 -0.01504096

# Step 6: Run Regression of Excess Firm Return on Excess Market Return  
CAPM<-lm(exret~exmkt, data = combo)  
summary(CAPM)

##   
## Call:  
## lm(formula = exret ~ exmkt, data = combo)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.057039 -0.013597 0.002671 0.015490 0.053562   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.011682 0.004547 2.569 0.0148 \*   
## exmkt 0.953663 0.142819 6.677 0.000000115 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02569 on 34 degrees of freedom  
## Multiple R-squared: 0.5674, Adjusted R-squared: 0.5546   
## F-statistic: 44.59 on 1 and 34 DF, p-value: 0.0000001152

beta <- summary(CAPM)$coefficients[2]  
beta

## [1] 0.9536632

adj.beta<-(2/3)\*beta+(1/3)\*1  
adj.beta

## [1] 0.9691088

## CAPM - alpha

* CAPM Alpha model is used to find the contribution of the fund manager to the performance of the fund.
* Thus we need to find whether the manager's contribution is worth more than just passively investing in the index.
* For this we see alpha of the manager as a measure this outperformance.
* Alpha of a portfolio can be calculated by using the excess returnform of the CAPM
* Regression controls for the sensitivity of the portfolio's return to its benchmark
* Thus any return which is not accounted in the market is attributed to the manager's contribution
* If alpha is positive and statistically significant, the manager is taken to have provided positive value
* Alpha of the portfolio is the intercept term from the regression output
* The intercept is equal to 0.012004, which translates to a monthly return of 1.2%
* This alpha is statistically significant at the 5% level, which is also the conventional level of significance used in practice

TO sum up everything, if our hypothetical portfolio gives us returns, the manager provided an incremental return of 1.2% per month beyond that of what is expected from its sensitivity to the benchmark.

## CAPM - Beta

* The beta of a portfolio calculates the sensitivity of the portfolio's return to the movement of the overall market.
* Beta capture the systematic risk. Systematic risk is the portion of a security's risk that cannot be diversified away and, as such, it is commonly thought of as the level of risk that investors are compensated from taking on.
* The results of the CAPM regression show that the CAPM beta is 0.953663. This beta of 0.953663 can then be used in the CAPM to calculate, say, the cost of equity for the company.
* This means that if the market goes up by 1 %, we expect our portfolio to go up by only 0.95 %. However, if the market goes down by 1 %, we expect our portfolio to only go down by 0.95 %. A beta less than one is consistent with betas of defensive stocks as these stocks are less affected by adverse market movements.

## Calculate Adjusted Beta

* Generally speaking, betas that are above the market beta of one tend to go down in the long-term, while betas that are below the market beta of one tend to go up in the long-term
* Since Betas are used to calculate the cost of equity, they should be adjusted to reflect the market Beta.
* After adjustement the Beta is coming out to be 0.9691088.

# French Fama 3 factor model

#Step 1: Import Portfolio Returns Data  
#Already done  
  
#Step 2: Import Fama-French Data Retrieved FromKen French's Website  
FF.raw<-read.csv(file="F-F\_Research\_Data\_Factors.csv")  
head(FF.raw)

## date Mkt.RF SMB HML RF  
## 1 192607 2.96 -2.30 -2.87 0.22  
## 2 192608 2.64 -1.40 4.19 0.25  
## 3 192609 0.36 -1.32 0.01 0.23  
## 4 192610 -3.24 0.04 0.51 0.32  
## 5 192611 2.53 -0.20 -0.35 0.31  
## 6 192612 2.62 -0.04 -0.02 0.28

tail(FF.raw)

## date Mkt.RF SMB HML RF  
## 1089 201703 0.17 1.20 -3.17 0.03  
## 1090 201704 1.09 0.73 -1.91 0.05  
## 1091 201705 1.06 -2.54 -3.75 0.06  
## 1092 201706 0.78 2.15 1.32 0.06  
## 1093 201707 1.87 -1.41 -0.28 0.07  
## 1094 201708 0.17 -1.70 -2.25 0.07

FF.raw$date <- seq(as.Date("1926-07-01"), as.Date("2017-08-31"),by="months")  
FF.data<-subset(FF.raw, FF.raw$date>="2013-01-01" & FF.raw$date<="2015-12-31")  
names(FF.data) <- c("date", "exmkt", "SMB", "HML", "rf")  
FF.data$date <- as.yearmon(FF.data$date,"%Y-%m-%d")  
head(FF.data)

## date exmkt SMB HML rf  
## 1039 Jan 2013 5.57 0.39 0.92 0  
## 1040 Feb 2013 1.29 -0.45 0.00 0  
## 1041 Mar 2013 4.03 0.79 -0.26 0  
## 1042 Apr 2013 1.55 -2.44 0.59 0  
## 1043 May 2013 2.80 1.67 2.55 0  
## 1044 Jun 2013 -1.20 1.22 -0.19 0

tail(FF.data)

## date exmkt SMB HML rf  
## 1069 Jul 2015 1.54 -4.15 -4.14 0.00  
## 1070 Aug 2015 -6.04 0.49 2.69 0.00  
## 1071 Sep 2015 -3.08 -2.64 0.53 0.00  
## 1072 Oct 2015 7.75 -1.98 -0.09 0.00  
## 1073 Nov 2015 0.56 3.64 -0.51 0.00  
## 1074 Dec 2015 -2.17 -2.81 -2.57 0.01

#Step 3: Combine FF.data with portfolio  
FF.data<-cbind(FF.data,data.frame(port))  
FF.data$exmkt <- FF.data$exmkt/100  
FF.data$SMB <- FF.data$SMB/100  
FF.data$HML <- FF.data$HML/100  
FF.data$rf <- FF.data$rf/100  
head(FF.data)

## date exmkt SMB HML rf date port.ret  
## 1039 Jan 2013 0.0557 0.0039 0.0092 0 Jan 2013 0.0879581738  
## 1040 Feb 2013 0.0129 -0.0045 0.0000 0 Feb 2013 0.0100590784  
## 1041 Mar 2013 0.0403 0.0079 -0.0026 0 Mar 2013 0.0136310206  
## 1042 Apr 2013 0.0155 -0.0244 0.0059 0 Apr 2013 0.0559940975  
## 1043 May 2013 0.0280 0.0167 0.0255 0 May 2013 0.0626252201  
## 1044 Jun 2013 -0.0120 0.0122 -0.0019 0 Jun 2013 -0.0003342324

#Step 4: create excess portfolio return  
FF.data$exret <- FF.data$port.ret-FF.data$rf  
head(FF.data)

## date exmkt SMB HML rf date port.ret  
## 1039 Jan 2013 0.0557 0.0039 0.0092 0 Jan 2013 0.0879581738  
## 1040 Feb 2013 0.0129 -0.0045 0.0000 0 Feb 2013 0.0100590784  
## 1041 Mar 2013 0.0403 0.0079 -0.0026 0 Mar 2013 0.0136310206  
## 1042 Apr 2013 0.0155 -0.0244 0.0059 0 Apr 2013 0.0559940975  
## 1043 May 2013 0.0280 0.0167 0.0255 0 May 2013 0.0626252201  
## 1044 Jun 2013 -0.0120 0.0122 -0.0019 0 Jun 2013 -0.0003342324  
## exret  
## 1039 0.0879581738  
## 1040 0.0100590784  
## 1041 0.0136310206  
## 1042 0.0559940975  
## 1043 0.0626252201  
## 1044 -0.0003342324

#Step 5: Run Regression Using Fama-French Factors  
FF.reg<-lm(exret~exmkt+SMB+HML,data=FF.data)  
summary(FF.reg)

##   
## Call:  
## lm(formula = exret ~ exmkt + SMB + HML, data = FF.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.043882 -0.015044 0.000466 0.013363 0.045563   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.009201 0.004462 2.062 0.0474 \*   
## exmkt 0.956125 0.133322 7.172 0.0000000384 \*\*\*  
## SMB -0.042413 0.175219 -0.242 0.8103   
## HML -0.416403 0.225067 -1.850 0.0735 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02449 on 32 degrees of freedom  
## Multiple R-squared: 0.6301, Adjusted R-squared: 0.5954   
## F-statistic: 18.17 on 3 and 32 DF, p-value: 0.0000004588

#Step 6: compare with CAPM model  
CAPM<-lm(exret~exmkt, data = FF.data)  
summary(CAPM)

##   
## Call:  
## lm(formula = exret ~ exmkt, data = FF.data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.048318 -0.014073 0.002734 0.014098 0.051844   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.01066 0.00447 2.385 0.0228 \*   
## exmkt 0.93952 0.13436 6.993 0.0000000457 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.02501 on 34 degrees of freedom  
## Multiple R-squared: 0.5898, Adjusted R-squared: 0.5778   
## F-statistic: 48.9 on 1 and 34 DF, p-value: 0.00000004567

# Comparing beta, p-value of the beta, and adjusted R-squared  
betas<-rbind( cbind(summary(FF.reg)$coefficient[2], summary(FF.reg)$coefficient[14], summary(FF.reg)$adj.r.squared), cbind(summary(CAPM)$coefficient[2], summary(CAPM)$coefficient[8], summary(CAPM)$adj.r.squared))  
  
colnames(betas)<-paste(c("Beta","p-Value","Adj. R-Squared"))  
rownames(betas)<-paste(c("Fama-French","CAPM"))  
betas

## Beta p-Value Adj. R-Squared  
## Fama-French 0.9561254 0.00000003842001 0.5953773  
## CAPM 0.9395159 0.00000004567320 0.5777847

# F-F Model

* suggests other factors may need to be added to help explain the remaining variation in asset returns unexplained by the market.
* The betas and p-values suggest that the returns of our portfolio is sensitive to the changes in the market.
* The CAPM beta was high at 0.94 but F-F beta is a bit higher at 0.95.
* Since FF is a three-factor model, the calculation of the cost of equity has to be with all three factors.
* The output shows that FF regression is a slightly better model than the CAPM in explaining the variation in our portfolio's returns based on having a higher Adjusted R-Squared.

# Write a function for bond evaluation on non-coupon payment dates.

# Bond Evaluation on non-coupon payment dates  
  
settle.date<-as.Date("2014-01-08")  
next.coupon<-as.Date("2014-12-08")  
mat.date<-as.Date("2017-12-08")  
cpn.pmts<-4  
coupon.freq = 1  
yield=0.018  
par = 1000  
coupon = 0.0  
  
bondval <- function(coupon, par, yield, coupon.freq, cpn.pmts, settle.date, next.coupon, mat.date)  
{  
days.next.cpn<-as.numeric((next.coupon-settle.date))  
days.next.cpn  
  
  
days.cpn.per<-360/coupon.freq  
days.cpn.per  
  
days.last.cpn<-days.cpn.per-days.next.cpn  
days.last.cpn  
  
yield.period=yield  
yield.period  
pv.principal<-par/(1+(yield.period))^(cpn.pmts-1+(days.next.cpn/days.cpn.per))  
pv.principal  
coupon.period=coupon/coupon.freq  
bond.cf<-rep(coupon.period \* par, times=cpn.pmts, length.out=NA, each=1)  
  
bond.cf<-data.frame(bond.cf)  
bond.cf  
bond.cf$period<-c(1:cpn.pmts)  
bond.cf  
bond.cf$disc<-(1+yield.period)^(bond.cf$period-1+(days.next.cpn/days.cpn.per))  
bond.cf  
bond.cf$value<-bond.cf$bond.cf/bond.cf$disc  
bond.cf  
pv.coupons<-sum(bond.cf$value)  
pv.coupons  
interest<- -(par \* (coupon.period) \* (days.last.cpn/days.cpn.per))  
interest  
bond.value<-pv.principal+pv.coupons+interest  
bond.value  
}  
  
bondval(coupon, par, yield, coupon.freq, cpn.pmts, settle.date, next.coupon, mat.date)

## [1] 932.3274

####################Reinforcing this in the standard BondPrice function  
  
bondprc<-function(coupon,maturity,yield,par,coupon.freq){  
 periods=maturity\*coupon.freq  
 coupon.period=coupon/coupon.freq  
 yield.period=yield/coupon.freq  
 bond.coupon<-rep(coupon.period,times=periods,length.out=NA,each=1)  
 bond.df<-as.data.frame(bond.coupon)  
 for (i in 1:periods) {  
 bond.df$cf[i]=par\*coupon.period  
 bond.df$period[i]=i  
 bond.df$yield[i]=yield.period  
 }  
 bond.df$cf[periods]=bond.df$cf[periods]+par  
 bond.df$PV=bond.df$cf/((1+bond.df$yield)^bond.df$period)  
 value=sum(bond.df$PV)  
 value  
}  
  
  
coupon = 0  
maturity1 = 5  
maturity2 = 25  
yield = c(.06, .07, .08, .0850, .0890, .0899, .09, .0901, .0910, .0950, .10, .11, .12)  
par = 1000  
coupon.freq = 2  
  
pv1 <- c()  
pv2 <- c()  
  
for (i in 1:length(yield)){  
 pv1 <- c(pv1, bondprc(coupon,maturity1,yield[i],par,coupon.freq))  
 pv2 <- c(pv2, bondprc(coupon,maturity2,yield[i],par,coupon.freq))  
}  
  
price <- data.frame(pv1, pv2)  
names(price) = c(paste0(coupon\*100, "%/",maturity1), paste0(coupon\*100, "%/",maturity2))  
price <- cbind(price, pv1, pv2)  
names(price)[1:2] = c(paste0(coupon\*100, "%/",maturity1), paste0(coupon\*100, "%/",maturity2))  
price

## 0%/5 0%/25 pv1 pv2  
## 1 744.0939 228.10708 744.0939 228.10708  
## 2 708.9188 179.05337 708.9188 179.05337  
## 3 675.5642 140.71262 675.5642 140.71262  
## 4 659.5373 124.79489 659.5373 124.79489  
## 5 647.0168 113.39079 647.0168 113.39079  
## 6 644.2359 110.97483 644.2359 110.97483  
## 7 643.9277 110.70965 643.9277 110.70965  
## 8 643.6197 110.44512 643.6197 110.44512  
## 9 640.8548 108.09314 640.8548 108.09314  
## 10 628.7235 98.24228 628.7235 98.24228  
## 11 613.9133 87.20373 613.9133 87.20373  
## 12 585.4306 68.76652 585.4306 68.76652  
## 13 558.3948 54.28836 558.3948 54.28836

# What is binomial OPM. Write a function to for binomial OPM ?

* Binomial model is a lattice-based or tree-based model that allows us to model the path of the underlying asset's price in discrete time steps.
* The binomial option pricing model uses an iterative procedure, allowing for the specification of nodes, or points in time, during the time span between the valuation date and the option's expiration date. The model reduces possibilities of price changes, and removes the possibility for arbitrage.
* The Binomial Model is said to converge to the Black-Scholes-Merton OPM when the time increments approaches infinity.
* Using the Binomial Model Function, we can see the effect of increasing the time increments

## function for Binomian OPM. we give x= 1 if its a call, else x = -1 if its a put  
EuroCRR<- function(S,K,T,r,sigma,n,type){  
 x=NA  
 if (type=="call") x=1  
 if (type=="put") x=-1  
 if (is.na(x)) stop("Option Type can only be call or put")  
 dt=T/ n  
 u=exp(sigma\*sqrt(dt))  
 d=1/ u  
 p=((1+r\*dt)-d)/ (u-d)  
 disc<- (1+r\*dt)  
 OptVal<- x\*(S\*u^(0:n)\*d^(n:0)-K)  
 OptVal=ifelse(OptVal<0,0,OptVal)  
 for (j in seq(from=n-1,to=0,by=-1))  
 for (i in 0:j)  
 OptVal[i+1]=(p\*OptVal[i+2]+(1-p)\*OptVal[i+1])/disc  
 value=OptVal[1]  
 results<- rbind(u,d,p,value)  
 results  
}  
  
EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,2,"call")

## [,1]  
## u 1.1147023  
## d 0.8971005  
## p 0.4732367  
## value 24.3977599

EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,100,"call")

## [,1]  
## u 1.0154751  
## d 0.9847607  
## p 0.4962115  
## value 26.3065046

EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,1000,"call")

## [,1]  
## u 1.0048680  
## d 0.9951556  
## p 0.4988020  
## value 26.2583744

#We can do a similar analysis with put options  
EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,2,"put")

## [,1]  
## u 1.1147023  
## d 0.8971005  
## p 0.4732367  
## value 20.5464068

EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,100,"put")

## [,1]  
## u 1.0154751  
## d 0.9847607  
## p 0.4962115  
## value 22.4551491

EuroCRR(398.79,395,0.2219178,0.0007,0.3259855,1000,"put")

## [,1]  
## u 1.0048680  
## d 0.9951556  
## p 0.4988020  
## value 22.4070189

##confirming the answer with longer method  
  
S=398.79  
K=395  
TTM=0.2219178  
r=0.0007  
sigma=0.3259855  
n=2  
dt=TTM/n  
dt

## [1] 0.1109589

disc=(1+r \* dt)  
disc

## [1] 1.000078

u=exp(sigma \* sqrt(dt))  
u

## [1] 1.114702

d=1/u  
d

## [1] 0.8971005

p=((1+r \* dt)-d)/(u-d)  
p

## [1] 0.4732367

UP<- u^(0:n)  
UP

## [1] 1.000000 1.114702 1.242561

DOWN<- d^(n:0)  
DOWN

## [1] 0.8047893 0.8971005 1.0000000

terminal<- S \* UP \* DOWN  
terminal

## [1] 320.9419 398.7900 495.5210

terminal.optval<-ifelse(terminal-K<0,0,terminal-K)  
terminal.optval

## [1] 0.000 3.790 100.521

for (j in seq(from=n-1,to=0,by=-1)) for (i in 0:j) terminal.optval[i+1]= (p \* terminal.optval[i+2]+(1-p) \* terminal.optval[i+1])/disc  
terminal.optval

## [1] 24.39776 49.56281 100.52099

call.optval<-terminal.optval[1]  
call.optval

## [1] 24.39776