FIS FINAL

Asmita Kumar

12/17/2017

```
##Create a portfolio of 5 companies of your choice
##Time period - Jan-2013 to Dec-2015
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

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

library(xts)
library(PerformanceAnalytics)

##
## Attaching package: 'PerformanceAnalytics'

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

library(zoo)
library(quadprog)

##Step 1: Import Returns Data and Convert it Into a Matrix
data.IBM <- getSymbols("IBM", from = "2012-12-31", to = "2015-12-31", auto.as
sign = 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).

## Warning in strptime(xx, f <- "%Y-%m-%d", tz = "GMT"): unknown timezone
## 'zone/tz/2017c.1.0/zoneinfo/America/New_York'

data.IBM <- to.monthly(data.IBM)
IBM.ret <- Return.calculate(data.IBM$data.IBM.Adjusted)
IBM.ret <- IBM.ret[-1,]
head(IBM.ret)

##          data.IBM.Adjusted
## Jan 2013       0.060141176
## Feb 2013      -0.006867989
## Mar 2013       0.062092173
## Apr 2013      -0.050445401
## May 2013       0.031870398
## Jun 2013      -0.081290078

data.GOOGL <- getSymbols("GOOGL", from = "2012-12-31", to = "2015-12-31", aut
o.assign = FALSE)
data.GOOGL <- to.monthly(data.GOOGL)
GOOGL.ret <- Return.calculate(data.GOOGL$data.GOOGL.Adjusted)
GOOGL.ret <- GOOGL.ret[-1,]
head(GOOGL.ret)

##          data.GOOGL.Adjusted
## Jan 2013         0.068294326
## Feb 2013         0.060223087
## Mar 2013        -0.008749389
## Apr 2013         0.038252798
## May 2013         0.056574966
## Jun 2013         0.010502537

data.AMZN <- getSymbols("AMZN", from = "2012-12-31", to = "2015-12-31", auto.
assign = FALSE)
data.AMZN <- to.monthly(data.AMZN)
AMZN.ret <- Return.calculate(data.AMZN$data.AMZN.Adjusted)
AMZN.ret <- AMZN.ret[-1,]
head(AMZN.ret)

##          data.AMZN.Adjusted
## Jan 2013        0.058317078
## Feb 2013       -0.004632810
## Mar 2013        0.008400504
## Apr 2013       -0.047581495
```

```
## May 2013        0.060635964
## Jun 2013        0.031537851

data.MSFT <- getSymbols("MSFT", from = "2012-12-31", to = "2015-12-31", auto.
assign = FALSE)
data.MSFT <- to.monthly(data.MSFT)
MSFT.ret <- Return.calculate(data.MSFT$data.MSFT.Adjusted)
MSFT.ret <- MSFT.ret[-1,]
head(MSFT.ret)

##          data.MSFT.Adjusted
## Jan 2013         0.02770493
## Feb 2013         0.02113535
## Mar 2013         0.02913666
## Apr 2013         0.15693827
## May 2013         0.06177410
## Jun 2013        -0.01031525

data.FB <- getSymbols("FB", from = "2012-12-31", to = "2015-12-31", auto.assi
gn = FALSE)
data.FB <- to.monthly(data.FB)
FB.ret <- Return.calculate(data.FB$data.FB.Adjusted)
FB.ret <- FB.ret[-1,]
head(FB.ret)

##          data.FB.Adjusted
## Jan 2013       0.16378658
## Feb 2013      -0.12040026
## Mar 2013      -0.06128440
## Apr 2013       0.08561376
## May 2013      -0.12315448
## Jun 2013       0.02176587

Ret.monthly <- cbind(IBM.ret , GOOGL.ret, AMZN.ret, MSFT.ret, FB.ret)
mat.ret<-matrix(Ret.monthly,nrow(Ret.monthly))
colnames(mat.ret)<-c("IBM.Ret", "GOOGL.Ret","AMZN.ret", "MSFT.ret", "FB.ret")
head(mat.ret)

##           IBM.Ret    GOOGL.Ret     AMZN.ret    MSFT.ret      FB.ret
## [1,]  0.060141176  0.068294326  0.058317078  0.02770493  0.16378658
## [2,] -0.006867989  0.060223087 -0.004632810  0.02113535 -0.12040026
## [3,]  0.062092173 -0.008749389  0.008400504  0.02913666 -0.06128440
## [4,] -0.050445401  0.038252798 -0.047581495  0.15693827  0.08561376
## [5,]  0.031870398  0.056574966  0.060635964  0.06177410 -0.12315448
## [6,] -0.081290078  0.010502537  0.031537851 -0.01031525  0.02176587

###Step 2: Calculate Variance-Covariance(VCOV) Matrix of Returns

VCOV<-cov(mat.ret)
VCOV
```

```
##                IBM.Ret    GOOGL.Ret     AMZN.ret      MSFT.ret
## IBM.Ret   2.214550e-03 2.407471e-06 0.0006824569  0.0006953608
## GOOGL.Ret 2.407471e-06 3.685969e-03 0.0034314501  0.0012743542
## AMZN.ret  6.824569e-04 3.431450e-03 0.0070710921  0.0015912500
## MSFT.ret  6.953608e-04 1.274354e-03 0.0015912500  0.0046071950
## FB.ret    3.584108e-04 1.336897e-03 0.0015648578 -0.0012740161
##                  FB.ret
## IBM.Ret    0.0003584108
## GOOGL.Ret  0.0013368966
## AMZN.ret   0.0015648578
## MSFT.ret  -0.0012740161
## FB.ret     0.0122330261

##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("IBM","GOOGL","AMZN","MSFT","FB"))
avg.ret

##            Avg.Ret
## IBM   -0.005624103
## GOOGL  0.024225543
## AMZN   0.031723510
## MSFT   0.025466509
## FB     0.044432257

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

## [1] -0.005624103

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

## [1] 0.04443226

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

## [1] -0.005624103 -0.005118483 -0.004612863 -0.004107244 -0.003601624
## [6] -0.003096004

tail(tgt.ret)

## [1] 0.04190416 0.04240978 0.04291540 0.04342102 0.04392664 0.04443226

##Step 4: Construct Dummy Portfolio Standard Deviation Vector
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
##  [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
##  [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
```

```
##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.IBM","wgt.GOOGL","wgt.AMZN","wt.MSFT","wt.FB"))
wgt[1,2:5]<-0
wgt[nrow(wgt),1]<-0
head(wgt)
```

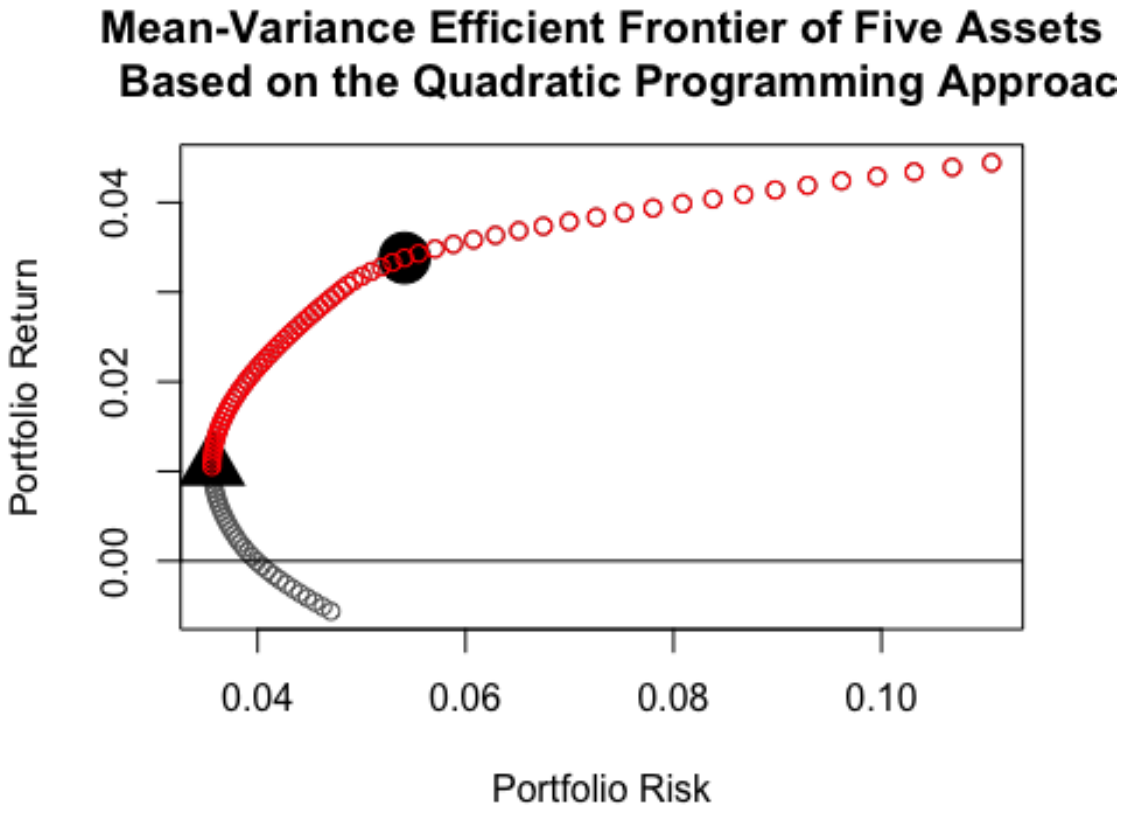
```
##        wgt.IBM  wgt.GOOGL      wgt.AMZN wt.MSFT wt.FB
## [1,] 1.0000000 0.00000000  0.000000e+00       0     0
## [2,] 0.9830611 0.01693889 -3.587565e-17       0     0
## [3,] 0.9661222 0.03387777 -3.338412e-17       0     0
## [4,] 0.9491833 0.05081666 -3.089259e-17       0     0
## [5,] 0.9322445 0.06775555 -2.840106e-17       0     0
## [6,] 0.9153056 0.08469444 -2.590953e-17       0     0
```

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

```
##        tgt.ret     tgt.sd   wgt.IBM  wgt.GOOGL      wgt.AMZN wt.MSFT wt.FB
## 1 -0.005624103 0.04705901 1.0000000 0.00000000  0.000000e+00       0     0
## 2 -0.005118483 0.04627418 0.9830611 0.01693889 -3.587565e-17       0     0
## 3 -0.004612863 0.04551299 0.9661222 0.03387777 -3.338412e-17       0     0
```

```
## 4 -0.004107244 0.04477664 0.9491833 0.05081666 -3.089259e-17       0     0
## 5 -0.003601624 0.04406639 0.9322445 0.06775555 -2.840106e-17       0     0
## 6 -0.003096004 0.04338351 0.9153056 0.08469444 -2.590953e-17       0     0

##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.IBM wgt.GOOGL    wgt.AMZN   wt.MSFT
## 33 0.01055573 0.03560099 0.5144024 0.2666804 4.07718e-20 0.1444013
##         wt.FB
## 33 0.07451589

##Step 9: Identify the Tangency Portfolio
## RISK-FREE RATE ON 12/15 = 0.0124
riskfree = 0.0124
tgt.port$Sharpe<-(tgt.port$tgt.ret-riskfree)/tgt.port$tgt.sd
head(tgt.port)

##        tgt.ret     tgt.sd   wgt.IBM  wgt.GOOGL      wgt.AMZN wt.MSFT wt.FB
## 1 -0.005624103 0.04705901 1.0000000 0.00000000  0.000000e+00       0     0
## 2 -0.005118483 0.04627418 0.9830611 0.01693889 -3.587565e-17       0     0
## 3 -0.004612863 0.04551299 0.9661222 0.03387777 -3.338412e-17       0     0
## 4 -0.004107244 0.04477664 0.9491833 0.05081666 -3.089259e-17       0     0
## 5 -0.003601624 0.04406639 0.9322445 0.06775555 -2.840106e-17       0     0
## 6 -0.003096004 0.04338351 0.9153056 0.08469444 -2.590953e-17       0     0
##       Sharpe
## 1 -0.3830107
## 2 -0.3785801
## 3 -0.3738024
## 4 -0.3686575
## 5 -0.3631254
## 6 -0.3571865

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

##       tgt.ret     tgt.sd       wgt.IBM wgt.GOOGL  wgt.AMZN   wt.MSFT
## 79 0.03381424 0.05412948 -5.796567e-18         0 0.1948782 0.4292664
##        wt.FB    Sharpe
## 79 0.3758554 0.3956114

##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.IBM wgt.GOOGL     wgt.AMZN   wt.MSFT
## 33 0.01055573 0.03560099 0.5144024 0.2666804 4.077180e-20 0.1444013
## 34 0.01106135 0.03561123 0.5011214 0.2673804 1.863044e-18 0.1520484
## 35 0.01156697 0.03564106 0.4878404 0.2680805 3.685316e-18 0.1596955
## 36 0.01207259 0.03569043 0.4745594 0.2687806 5.507589e-18 0.1673426
## 37 0.01257821 0.03575927 0.4612784 0.2694806 7.329861e-18 0.1749897
```

```
## 38 0.01308383 0.03584744 0.4479974 0.2701807 9.152134e-18 0.1826368
##         wt.FB       Sharpe
## 33 0.07451589 -0.051803889
## 34 0.07944973 -0.037590665
## 35 0.08438356 -0.023372761
## 36 0.08931740 -0.009173613
## 37 0.09425124  0.004983592
## 38 0.09918507  0.019076098

##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" )
```




```
### HENCE MINIMUM VARIANCE EFFICIENT PORTFOLIO IS
## 33 0.01055573 0.03560099 0.5144024 0.2666804 0.0000000000000000000407718 0
.1444013 0.07451589 -0.051803889
## wgt.IBM=0.5144024 WGT.GOOGL=0.2666804 WGT.AMZN=0.0000000000000000000407718
```

```
WGT.MSFT=0.1444013 WGT.FB = 0.07451589



## CONSTRUCT NEW PORTFOLIO
data.IBM <- getSymbols("IBM", from = "2012-12-31", to = "2015-12-31", auto.as
sign = FALSE)
data.GOOGL <- getSymbols("GOOGL", from="2012-12-31", to="2015-12-31", auto.as
sign=FALSE)
data.AMZN <- getSymbols("AMZN", from="2012-12-31", to="2015-12-31", auto.assi
gn=FALSE)
data.MSFT <- getSymbols("MSFT", from="2012-12-31", to="2015-12-31", auto.assi
gn=FALSE)
data.FB <- getSymbols("FB", from="2012-12-31", to="2015-12-31", auto.assign=F
ALSE)

ret.IBM <- Return.calculate(data.IBM$IBM.Adjusted)
ret.GOOGL <- Return.calculate(data.GOOGL$GOOGL.Adjusted)
ret.AMZN <- Return.calculate(data.AMZN$AMZN.Adjusted)
ret.MSFT <- Return.calculate(data.MSFT$MSFT.Adjusted)
ret.FB <- Return.calculate(data.FB$FB.Adjusted)



returns <- cbind(ret.IBM, ret.GOOGL, ret.AMZN, ret.MSFT, ret.FB)
returns <- returns[-1,]
names(returns) <- c("IBM.ret", "GOOGL.ret", "AMZN.ret", "MSFT.ret","FB.ret")

####Cumulative Portfolio return
##wgt.IBM=0.5144024 WGT.GOOGL=0.2666804 WGT.AMZN=0.000000000000000000407718
WGT.MSFT=0.1444013 WGT.FB = 0.07451589
port.ret <- Return.portfolio(returns, weights = c(0.5144024 ,0.2666804,0.0000
000000000000000407718,0.1444013,0.07451589), rebalance_on = "quarters")
port.mean<-mean(port.ret)
cum.port.ret <- Return.cumulative(port.ret)
cum.port.ret

##                   portfolio.returns
## Cumulative Return         0.4464865

#annualized standard deviation and covariance
sd.IBM <- sd(returns$IBM.ret)
annualized.sd.IBM <- sd.IBM*sqrt(252)
annualized.sd.IBM

## [1] 0.1905667

sd.GOOGL <- sd(returns$GOOGL.ret)
annualized.sd.GOOGL <- sd.GOOGL*sqrt(252)
annualized.sd.GOOGL

## [1] 0.2427054
```

```r
sd.AMZN <- sd(returns$AMZN.ret)
annualized.sd.AMZN <- sd.AMZN*sqrt(252)
annualized.sd.AMZN
```

```
## [1] 0.3114065
```

```r
sd.MSFT <- sd(returns$MSFT.ret)
annualized.sd.MSFT <- sd.MSFT*sqrt(252)
annualized.sd.MSFT
```

```
## [1] 0.2432222
```

```r
sd.FB <- sd(returns$FB.ret)
annualized.sd.FB <- sd.FB*sqrt(252)
annualized.sd.FB
```

```
## [1] 0.3694343
```

```r
ret.cov12 <- cov(returns$IBM.ret,returns$GOOGL.ret)
annualized.ret.cov12 <- ret.cov12 * 252
annualized.ret.cov12
```

```
##          GOOGL.ret
## IBM.ret 0.01662691
```

```r
ret.cov13 <- cov(returns$IBM.ret,returns$AMZN.ret)
annualized.ret.cov13 <- ret.cov13 * 252
annualized.ret.cov13
```

```
##           AMZN.ret
## IBM.ret 0.01550124
```

```r
ret.cov14 <- cov(returns$IBM.ret,returns$MSFT.ret)
annualized.ret.cov14 <- ret.cov14 * 252
annualized.ret.cov14
```

```
##           MSFT.ret
## IBM.ret 0.01800898
```

```r
ret.cov15 <- cov(returns$IBM.ret,returns$FB.ret)
annualized.ret.cov15 <- ret.cov15 * 252
annualized.ret.cov15
```

```
##             FB.ret
## IBM.ret 0.01530914
```

```r
ret.cov23 <- cov(returns$GOOGL.ret,returns$AMZN.ret)
annualized.ret.cov23 <- ret.cov23 * 252
annualized.ret.cov23
```

```
##             AMZN.ret
## GOOGL.ret 0.03815633
```

```
ret.cov24 <- cov(returns$GOOGL.ret,returns$MSFT.ret)
annualized.ret.cov24 <- ret.cov24 * 252
annualized.ret.cov24

##             MSFT.ret
## GOOGL.ret 0.02369955

ret.cov25 <- cov(returns$GOOGL.ret,returns$FB.ret)
annualized.ret.cov25 <- ret.cov25 * 252
annualized.ret.cov25

##               FB.ret
## GOOGL.ret 0.03453379

ret.cov34 <- cov(returns$AMZN.ret,returns$MSFT.ret,)
annualized.ret.cov34 <- ret.cov34 * 252
annualized.ret.cov34

##            MSFT.ret
## AMZN.ret 0.02537972

ret.cov35 <- cov(returns$AMZN.ret,returns$FB.ret,)
annualized.ret.cov35 <- ret.cov35 * 252
annualized.ret.cov35

##              FB.ret
## AMZN.ret 0.04313669

ret.cov45 <- cov(returns$MSFT.ret,returns$FB.ret)
annualized.ret.cov45 <- ret.cov45 * 252
annualized.ret.cov45

##             FB.ret
## MSFT.ret 0.0211183

###Portfolio Risk
wgt.IBM=0.5144024
wgt.GOOGL=0.2666804
wgt.AMZN=0.000000000000000000407718
wgt.MSFT=0.1444013
wgt.FB = 0.07451589

port.var <- (wgt.IBM^2*annualized.sd.IBM^2+ wgt.GOOGL^2*annualized.sd.GOOGL^2
+wgt.AMZN^2*annualized.sd.AMZN^2 + wgt.MSFT^2*annualized.sd.MSFT^2 + wgt.FB^2
*annualized.sd.FB^2+ 2*annualized.ret.cov12*wgt.GOOGL*wgt.IBM+ 2*annualized.r
et.cov13*wgt.IBM*wgt.AMZN+2*annualized.ret.cov14*wgt.IBM*wgt.MSFT+2*annualize
d.ret.cov15*wgt.IBM*wgt.FB+2*annualized.ret.cov23*wgt.GOOGL*wgt.AMZN+2*annual
ized.ret.cov24*wgt.GOOGL*wgt.MSFT+2*annualized.ret.cov25*wgt.GOOGL*wgt.FB+2*a
nnualized.ret.cov34*wgt.AMZN*wgt.MSFT+2*annualized.ret.cov35*wgt.AMZN*wgt.FB+
2*annualized.ret.cov45*wgt.MSFT*wgt.FB)

port.var
```

```
##          GOOGL.ret
## IBM.ret 0.02785326

port.sd <- sqrt(port.var)
port.sd

##         GOOGL.ret
## IBM.ret  0.166893

######################## MULTIPLE ASSET ################
### Multiple Asset ###
### Load the data and calculate the return
data.IBM <- getSymbols("IBM", from = "2012-12-31", to = "2015-12-31", auto.as
sign = FALSE)
data.GOOGL <- getSymbols("GOOGL", from="2012-12-31", to="2015-12-31", auto.as
sign=FALSE)
data.AMZN <- getSymbols("AMZN", from="2012-12-31", to="2015-12-31", auto.assi
gn=FALSE)
data.MSFT <- getSymbols("MSFT", from="2012-12-31", to="2015-12-31", auto.assi
gn=FALSE)
data.FB <- getSymbols("FB", from="2012-12-31", to="2015-12-31", auto.assign=F
ALSE)

ret.IBM <- Return.calculate(data.IBM$IBM.Adjusted)
ret.GOOGL <- Return.calculate(data.GOOGL$GOOGL.Adjusted)
ret.AMZN <- Return.calculate(data.AMZN$AMZN.Adjusted)
ret.MSFT <- Return.calculate(data.MSFT$MSFT.Adjusted)
ret.FB <- Return.calculate(data.FB$FB.Adjusted)

returns <- cbind(ret.IBM,ret.GOOGL,ret.AMZN,ret.MSFT, ret.FB)
returns <- returns[-1,]
names(returns) <- c("IBM.ret","GOOGL.ret", "AMZN.ret","MSFT.ret", "FB.ret")
head(returns)

##                IBM.ret     GOOGL.ret      AMZN.ret     MSFT.ret
## 2013-01-02  0.025058852  0.0224349577  0.0256706786  0.034069574
## 2013-01-03 -0.005500590  0.0005806976  0.0045470950 -0.013396147
## 2013-01-04 -0.006554921  0.0197603264  0.0025920109 -0.018715530
## 2013-01-07 -0.004381556 -0.0043632979  0.0359251291 -0.001869757
## 2013-01-08 -0.001398046 -0.0019734122 -0.0077478435 -0.005245410
## 2013-01-09 -0.002851715  0.0065730302 -0.0001126173  0.005649585
##                 FB.ret
## 2013-01-02  0.051840682
## 2013-01-03 -0.008214286
## 2013-01-04  0.035649982
## 2013-01-07  0.022948540
## 2013-01-08 -0.012236608
## 2013-01-09  0.052649727

## Weight and Transpose weight matrix
```

```
WGT.asset<-c(0.5144024,0.2666804,0.00000000000000000407718,0.1444013,0.0745
1589)
WGT.asset<-matrix(WGT.asset,1)
WGT.asset
```

```
##           [,1]      [,2]        [,3]      [,4]       [,5]
## [1,] 0.5144024 0.2666804 4.07718e-20 0.1444013 0.07451589
```

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

```
##              [,1]
## [1,] 5.144024e-01
## [2,] 2.666804e-01
## [3,] 4.077180e-20
## [4,] 1.444013e-01
## [5,] 7.451589e-02
```

```
###Construct Variance-Covariance Matrix

mat.Ret<-as.matrix(returns)
VCOV.asset<-cov(mat.Ret)*252
VCOV.asset
```

```
##              IBM.ret  GOOGL.ret   AMZN.ret   MSFT.ret     FB.ret
## IBM.ret   0.03631567 0.01662691 0.01550124 0.01800898 0.01530914
## GOOGL.ret 0.01662691 0.05890589 0.03815633 0.02369955 0.03453379
## AMZN.ret  0.01550124 0.03815633 0.09697402 0.02537972 0.04313669
## MSFT.ret  0.01800898 0.02369955 0.02537972 0.05915704 0.02111830
## FB.ret    0.01530914 0.03453379 0.04313669 0.02111830 0.13648173
```

```
##Portfolio Risk

mat.varasset<-WGT.asset %*% VCOV.asset %*% tWGT.asset
mat.sdasset<-sqrt(mat.varasset)
mat.sdasset
```

```
##          [,1]
## [1,] 0.166893
```

```
############### HISTORICAL VAR ###########

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

```
##                       IBM.ret
## Cumulative Return -0.2156418
```

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

```
##                   GOOGL.ret
## Cumulative Return  1.232208

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

##                   AMZN.ret
## Cumulative Return 1.746721

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

##                   MSFT.ret
## Cumulative Return 1.292215

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

##                     FB.ret
## Cumulative Return 2.990233

IBM.val <- 514402.4 * ( 1 + ret.cum.IBM)
IBM.val

##                    IBM.ret
## Cumulative Return 403475.7

GOOGL.val <- 266680.4 * ( 1 + ret.cum.GOOGL)
GOOGL.val

##                   GOOGL.ret
## Cumulative Return  595286.2

AMZN.val <-0.0000000000000407718  * ( 1 + ret.cum.AMZN)
AMZN.val

##                      AMZN.ret
## Cumulative Return 1.119888e-13

MSFT.val <- 144401.3 * ( 1 + ret.cum.MSFT)
MSFT.val

##                   MSFT.ret
## Cumulative Return 330998.8

FB.val <- 74515.89 * ( 1 + ret.cum.FB)
FB.val

##                     FB.ret
## Cumulative Return 297335.7

last.idx <- cbind(IBM.val, GOOGL.val,  AMZN.val,MSFT.val, FB.val)
sum(last.idx)
```

```
## [1] 1627096
```

```
#Calculated simulated return

sim.portPnL <- last.idx[1]*returns$IBM.ret + last.idx[2]*returns$GOOGL.ret +
last.idx[3]*returns$AMZN.ret + last.idx[4]*returns$MSFT.ret + last.idx[5]*ret
urns$FB.ret
names(sim.portPnL) <- "Port.PnL"
head(sim.portPnL)
```

```
##             Port.PnL
## 2013-01-02 50156.934
## 2013-01-03 -8750.183
## 2013-01-04 13523.494
## 2013-01-07  1839.271
## 2013-01-08 -7113.428
## 2013-01-09 20286.888
```

```
#Historical VaR at 1% and 5%######
VaR01.Historical=quantile(-sim.portPnL$Port.PnL,0.99)
VaR01.Historical
```

```
##      99%
## 45978.01
```

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

```
##      95%
## 27603.93
```

```
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
## 2013-01-02 50156.934       0       0
## 2013-01-03 -8750.183       0       0
## 2013-01-04 13523.494       0       0
## 2013-01-07  1839.271       0       0
## 2013-01-08 -7113.428       0       0
## 2013-01-09 20286.888       0       0
```

```
#Extract Portfolio Losses in Excess of VaR and Compute Average of Losses in E
xcess 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] 57424.09
```

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

## [1] 39335.85

#######################CAPM ####################

######## Hypothetical Portfolio ###########
data.IBM <- getSymbols("IBM", from = "2012-12-31", to = "2015-12-31",
auto.assign = FALSE)
data.IBM <- to.monthly(data.IBM)
IBM.ret <- Return.calculate(data.IBM$data.IBM.Adjusted)

data.GOOGL <- getSymbols("GOOGL", from = "2012-12-31", to = "2015-12-31",
auto.assign = FALSE)
data.GOOGL <- to.monthly(data.GOOGL)
GOOGL.ret <- Return.calculate(data.GOOGL$data.GOOGL.Adjusted)

data.AMZN <- getSymbols("AMZN", from = "2012-12-31", to = "2015-12-31",
auto.assign = FALSE)
data.AMZN <- to.monthly(data.AMZN)
AMZN.ret <- Return.calculate(data.AMZN$data.AMZN.Adjusted)

data.MSFT <- getSymbols("MSFT", from = "2012-12-31", to = "2015-12-31",
auto.assign = FALSE)
data.MSFT <- to.monthly(data.MSFT)
MSFT.ret <- Return.calculate(data.MSFT$data.MSFT.Adjusted)

data.FB <- getSymbols("FB", from = "2012-12-31", to = "2015-12-31",
auto.assign = FALSE)
data.FB <- to.monthly(data.FB)
FB.ret <- Return.calculate(data.FB$data.FB.Adjusted)

portfolio <- cbind(IBM.ret , GOOGL.ret, AMZN.ret, MSFT.ret, FB.ret)
portfolio <- portfolio[-1,]
portfolio.ret <- Return.portfolio(portfolio, weights =
c(0.5144024,0.2666804,0.00000000000000000000407718,0.1444013,0.07451589),rebal
ance_on = "quarters" )

port.csv <- cbind(index(portfolio.ret), data.frame(portfolio.ret))
names(port.csv) <- c("date", "port.ret")
row.names(port.csv) <- seq(1, nrow(port.csv), by = 1)
write.csv(port.csv, "/Users/Asmita/Desktop/Hypothetical_Portfolio.csv",
row.names = F)
```

CAPITAL ASSET PRICING MODEL

CAPM is static (one period) model. The primary prediction is that a market portfolio of invested wealth is mean-variance efficient resulting in a linear cross sectional relationship between mean excess returns and exposures to the market factor (Fama and French, 1992). Capital asset pricing model describes the relationship between systematic risk or Beta and the expected return for the portfolio of assets that we have considered. The factor models help us assess the systematic risk.

The first market model that we have used in our explanation is the CAPM or Capital Asset Pricing Model. It is a single factor model.
CAPM takes the form of the following formula:
$r_f + B_a * (r_m – r_f)$
$r_f = A + B_a * (r_m – r_f)$
Where
is the expected return
r_f is the risk-free return
B_a is the beta of our portfolio = $Cov(\bar{r}, \bar{r}m)/Var(\bar{r}m)$
A is the alpha value of our portfolio
r_m is the expected market return

```
#Step 1: Create/Load your Porfolio Return
port <- read.csv("/Users/Asmita/Desktop/Hypothetical_Portfolio.csv")
port$date<-as.yearmon(as.character(port$date),"%b %Y")
head(port)
```

```
##       date     port.ret
## 1 Jan 2013  0.065354844
## 2 Feb 2013  0.005732464
## 3 Mar 2013  0.028677234
## 4 Apr 2013  0.011832193
## 5 May 2013  0.032788873
## 6 Jun 2013 -0.037210684
```

```
#Step 2: Load market return
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
```

```
#Step 3: Load Risk Free return
rf <- read.csv("/Users/Asmita/Desktop/DGS3MO.csv")
rf$date<-as.Date(rf$DATE,"%Y-%m-%d")
rf$DGS3MO<-as.numeric(as.character(rf$DGS3MO))
```

```
## Warning: NAs introduced by coercion
```

```
rf$DATE <- NULL
rf<-xts(rf$DGS3MO,order.by=rf$date)
```

```r
names(rf)<-paste("DGS3MO")
rf.monthly<-to.monthly(rf)

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

rf.monthly<-(1+rf.monthly[,1]/100)^(1/12)-1
rf.sub<-subset(rf.monthly,index(rf.monthly) >= as.yearmon("Jan 2013") & index
(rf.monthly) <= as.yearmon("Dec 2015"))

#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 6.664223e-05  0.065354844
## Feb 2013  0.01106065 4.998626e-05  0.005732464
## Mar 2013  0.03598772 9.162048e-05  0.028677234
## Apr 2013  0.01808577 6.664223e-05  0.011832193
## May 2013  0.02076281 4.998626e-05  0.032788873
## Jun 2013 -0.01499930 4.165712e-05 -0.037210684

#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 6.664223e-05  0.065354844  0.065288202  0.05036145
## Feb 2013  0.01106065 4.998626e-05  0.005732464  0.005682477  0.01101066
## Mar 2013  0.03598772 9.162048e-05  0.028677234  0.028585613  0.03589610
## Apr 2013  0.01808577 6.664223e-05  0.011832193  0.011765550  0.01801913
## May 2013  0.02076281 4.998626e-05  0.032788873  0.032738887  0.02071283
## Jun 2013 -0.01499930 4.165712e-05 -0.037210684 -0.037252341 -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
## -0.08915 -0.01686 -0.00361  0.01776  0.06950
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.005467   0.004907   1.114    0.273
```

```
## exmkt       0.817325   0.154114   5.303 6.93e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02772 on 34 degrees of freedom
## Multiple R-squared:  0.4527, Adjusted R-squared:  0.4366
## F-statistic: 28.13 on 1 and 34 DF,  p-value: 6.935e-06

#Calculate Adjusted Beta
beta <- summary(CAPM)$coefficients[2]
beta

## [1] 0.8173252

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

## [1] 0.8782168
```

CAPM BETA

Beta which is the systematic risk is also known as the sensitivity i.e. it determines how sensitive our portfolio's return is to changes in the market. The results above show that the beta value is 0.81. As the market increases by 1% our portfolio will go up by 0.81% but at the same a 1% decline in the market will lead to a 0.81% decline in our portfolio. A beta of less than 1 is less affected by adverse market movements.

CAPM ALPHA

Intercept here is the alpha value. Alpha determines the contribution of fund manager to the performance of the portfolio built. The alpha value of 0.005 means a monthly return of 0.5% beyond what's expected from its sensitivity to the benchmark.

ADJUSTED BETA

Adjusted beta is calculated as Adjusted Beta = 2/3 * beta + 1/3 * market beta of 1
Certain studies show that beta above the market beta of 1 tend to go down in the long run while betas below 1 tend to go up in the long term. To adjust betas to show this reversion to market beta we calculate adjusted which is done by applying a weight of 2/3 to the actual and weight of 1/3 to market beta of 1.
This calculation gives us a beta of 0.878.

```
#Market Model
#The CAPM requires that we use expected returns and the "true" market portfol
io. A more common way to calculate beta in practice is to use the market mode
l, because the market model uses a market proxy without the requirement that
this market proxy be the "true" market portfolio. In addition, the market mod
el does not require the use of a risk-free rate and, therefore, there is no n
eed to calculate the excess returns of the firm and the market.

reg<-lm(port.ret~mkt.ret, data = combo)
summary(reg)
```

```
##
## Call:
## lm(formula = port.ret ~ mkt.ret, data = combo)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.089154 -0.016867 -0.003605  0.017759  0.069495
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.005473   0.004909   1.115    0.273
## mkt.ret     0.817414   0.154145   5.303 6.94e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02772 on 34 degrees of freedom
## Multiple R-squared:  0.4527, Adjusted R-squared:  0.4366
## F-statistic: 28.12 on 1 and 34 DF,  p-value: 6.945e-06

#Notice the alpha and beta for this model and compare it with the CAPM model.
beta.mktmod<-summary(reg)$coefficients[2]
beta.mktmod

## [1] 0.8174141

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

## [1] 0.8782761
```

```
#####################  Fama-French  Three  Factor  Model  ####################
```

FAMA AND FRENCH THREE FACTOR MODEL

The Fama and French Three Factor Model is a multi factor asset pricing model that expands on the capital asset pricing model (CAPM) by adding size and value factors to the market risk factor in CAPM. This model considers the fact that value and small-cap stocks outperform markets on a regular basis. By including these two additional factors, the model adjusts for the outperformance tendency, which is thought to make it a better tool for evaluating manager performance.

In the FF Model, we have

$r_i = r_f + \beta_i (r_m - r_F) + hHML + sSMB$

where HML,SMB are defined as in the above table rm-rF = MRP from the above table. We downloaded data from Professor Kenneth French's Data Library up until August 2017.

```
#Step 1: Import Portfolio Returns Data
port <- read.csv("/Users/Asmita/Desktop/Hypothetical_Portfolio.csv")
port$date<-as.yearmon(as.character(port$date),"%b %Y")
head(port)
```

```
##       date     port.ret
## 1 Jan 2013  0.065354844
## 2 Feb 2013  0.005732464
## 3 Mar 2013  0.028677234
## 4 Apr 2013  0.011832193
## 5 May 2013  0.032788873
## 6 Jun 2013 -0.037210684
```

```
#Step 2: Import Fama-French Data Retrieved FromKen French's Website
FF.raw<-read.csv("/Users/Asmita/Desktop/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
```

```
#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
```

```r
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  0.065354844
## 1040 Feb 2013  0.0129 -0.0045  0.0000  0 Feb 2013  0.005732464
## 1041 Mar 2013  0.0403  0.0079 -0.0026  0 Mar 2013  0.028677234
## 1042 Apr 2013  0.0155 -0.0244  0.0059  0 Apr 2013  0.011832193
## 1043 May 2013  0.0280  0.0167  0.0255  0 May 2013  0.032788873
## 1044 Jun 2013 -0.0120  0.0122 -0.0019  0 Jun 2013 -0.037210684
```

```r
#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  0.065354844
## 1040 Feb 2013  0.0129 -0.0045  0.0000  0 Feb 2013  0.005732464
## 1041 Mar 2013  0.0403  0.0079 -0.0026  0 Mar 2013  0.028677234
## 1042 Apr 2013  0.0155 -0.0244  0.0059  0 Apr 2013  0.011832193
## 1043 May 2013  0.0280  0.0167  0.0255  0 May 2013  0.032788873
## 1044 Jun 2013 -0.0120  0.0122 -0.0019  0 Jun 2013 -0.037210684
##             exret
## 1039  0.065354844
## 1040  0.005732464
## 1041  0.028677234
## 1042  0.011832193
## 1043  0.032788873
## 1044 -0.037210684
```

```r
#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
## -0.065192 -0.017534  0.000993  0.020417  0.040292
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.002867   0.004565   0.628  0.53438
## exmkt        0.827034   0.136377   6.064 9.01e-07 ***
## SMB         -0.634775   0.179234  -3.542  0.00124 **
## HML         -0.214157   0.230223  -0.930  0.35923
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.02505 on 32 degrees of freedom
## Multiple R-squared:  0.5795, Adjusted R-squared:   0.54
## F-statistic:  14.7 on 3 and 32 DF,  p-value: 3.436e-06
```

Fama and French Alpha

Intercept here is the alpha value. Alpha is a risk-adjusted measure of active return on an investment. Similar to the CAPM model, Alpha determines the contribution of fund manager to the performance of the portfolio built. The alpha value we obtained was 0.0028 means a monthly return of 0.28% in excess of the reward for the assumed risk.

Fama and French Market Betas

The Fama and French Model essentially has 3 betas- Beta (mrkt), Beta(size) and Beta (value) which we obtained to be 0.827, -0.634 and -0.214 respectively of which the Beta(mrkt) was more statistically significant.

The market is described as having a beta of 1. The beta for a stock describes how much the stock's price moves in relation to the market. If a stock has a beta above 1, it's more volatile than the overall market. Hence for our result with a Beta(mrkt) of 0.827, it's theoretically 17.3% less volatile than the market.

A negative SMB coefficient would indicate a portfolio that favors large - cap stocks and the returns are attributable to exposure to large stocks.

The negative slope on beta(HML) of -0.214 means that the fund co-moves with the short side of HML, i.e. the portfolio has higher expected returns if low book-to-market (i.e., growth) stocks outperform high book-to-market (i.e., value) stocks, suggesting that the portfolio is predominantly growth stocks.

```
#Step 6: compare with CAPM model

CAPM<-lm(exret~exmkt, data = FF.data)
betas<-rbind( cbind(summary(FF.reg)$coefficient[2], summary(FF.reg)$coefficie
nt[14],summary(FF.reg)$adj.r.squared),cbind(summary(CAPM)$coefficient[2], sum
mary(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 0.8270336 9.011599e-07      0.5400243
## CAPM        0.7455613 3.150212e-05      0.3860164
```

Comparison of CAPM and Fama And French Models

The betas and p-values suggest that the returns of our portfolio is sensitive to the changes in the market. The CAPM beta was low at 0.745, but the FF beta is higher at 0.827. Since FF is a

three-factor model, the calculation of the cost of equity has to be with all three factors. We cannot simply use the market beta to calculate a cost of equity. In addition, 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 of 54% as opposed to the CAPM Adjusted R-Squared of 38.6%.

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

######## bondeval function ########

WHAT IS BINOMIAL OPM?

- The binomial option pricing model is an options valuation method that 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 a lattice-based or tree-based model that allows us to model the path of the underlying asset's price in discrete time steps.
- To describe how the binomial model works, consider a stock with value V today. In a binomial model, the value in 6 months can either go up by u or go down by d . That is, the value of the stock can either be Vu or Vd at the end of 6 months. At the end of the year, the value of the stock depends on whether you start at node Vu or node Vd . Starting in node Vu, we can either go up to node $V uu$ or go down to node $V u d$. Starting in node Vd, we can either go up to node Vdu or go down to node Vdd. Note that one feature of the binomial tree is that since u and d are fixed increments, the tree recombines.
- The binomial option pricing model assumes a perfectly efficient market. Under this assumption, it is able to provide a mathematical valuation of an option at each point in the timeframe specified.
- The binomial model takes a risk-neutral approach to valuation and assumes that underlying security prices can only either increase or decrease with time until the option expires worthless.
- Due to its simple and iterative structure, the binomial option pricing model presents certain unique advantages. For example, since it provides a stream of valuations for a derivative for each node in a span of time, it is useful for valuing derivatives such as American options. It is also much simpler than other pricing models such as the Black-Scholes model.

```
bondeval<-function(settle.date,next.coupon,mat.date,cpn.pmts,coupon.freq,yiel
d,par,coupon){
settle.date<-as.Date(settle.date)
next.coupon<-as.Date(next.coupon)
mat.date<-as.Date(mat.date)
days.next.cpn<-as.numeric((next.coupon-settle.date))
days.cpn.per<-360/coupon.freq
days.last.cpn<-days.cpn.per-days.next.cpn
yield.period=yield
pv.principal<-par/(1+(yield.period))^(cpn.pmts-1+(days.next.cpn/days.cpn.per)
)
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$period<-c(1:cpn.pmts)
bond.cf$disc<-(1+yield.period)^(bond.cf$period-1+(days.next.cpn/days.cpn.per)
)
bond.cf$value<-bond.cf$bond.cf/bond.cf$disc
pv.coupons<-sum(bond.cf$value)
interest<- -(par*(coupon.period)*(days.last.cpn/days.cpn.per))
bond.value<-pv.principal+pv.coupons+interest
bond.value
}
bondeval("2014-01-08","2014-12-08","2017-12-08",4,1,0.018,1000,0.0425)
```

```
## [1] 1092.085
```

```
## AS per the text for settle.date="2014-01-08",next.coupon="2014-12-08",mat.
date="2017-12-08",cpn.pmts=4,coupon.freq=1,yield=0.018,par=1000,coupon=0.0425
I got the bond valued at $1092.085
```

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

## Function ###
binomial.opm<-function(S,K,TTM,r,sigma,n){
 dt=TTM/ n
 disc=(1+r*dt)
 u=exp(sigma*sqrt(dt))
 d=1/ u
 p=((1+r*dt)-d)/ (u-d)
 UP<- u^(0:n)
 DOWN<- d^(n:0)
 terminal<- S*UP*DOWN
 terminal.optval<- ifelse(terminal-K<0,0,terminal-K)
 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])/dis
```

```
c
 call.optval<- terminal.optval[1]
 call.optval
}
binomial.opm(398.79,395,0.2219178,0.0007,0.3259855,2)

## [1] 24.39776

## AS per the text for S=398.79, K=395, TTM=0.2219178, r=0.0007,sigma=0.32598
55,n=2 I got the price today for the call option as $24.39776
```