충남대학교
Chungnam National University

# 투자론

- *R*과 Excel을 통한 금융데이터 분석 -

## 8주차
## R의 Collecting Data 및 Portfolio Analysis

충남대학교
장호규 교수

충남대학교
Chungnam National University

Unit 03

# Portfolio Analysis with R (2)

# Overview

- Warm up with `PortfolioAnalytics` package
- Beta Estimation
- Optimal Portfolio Analysis

## Warm up with pre-installed R data set

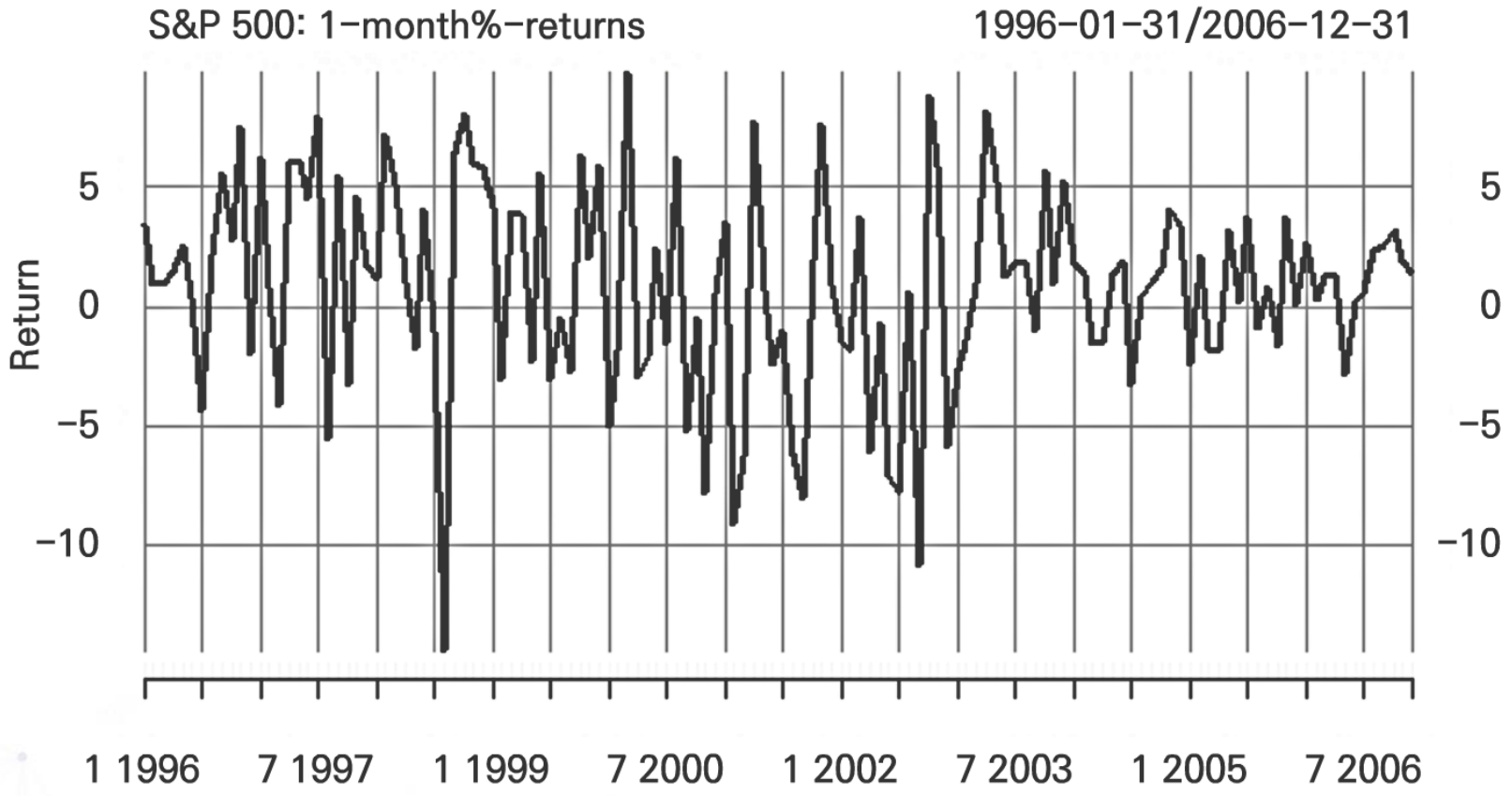
```
library(PortfolioAnalytics)
data(managers)
sp500.ret <- managers$SP500
plot(sp500.ret*100,
     main="S&P 500: 1-month %-returns",
     ylab="return",
     xlab="date")
```

# Warm up with pre-installed R data set

| | HAM1 | HAM2 | HAM3 | HAM4 | HAM5 | HAM6 | EDHEC LS EQ | SP500 TR | US 10Y TR | US 3m TR |
|---|---|---|---|---|---|---|---|---|---|---|
| **1996-01-31** | 0.0074 | NA | 0.0349 | 0.0222 | NA | NA | NA | 0.034000 | 0.00380 | 0.00456 |
| **1996-02-29** | 0.0193 | NA | 0.0351 | 0.0195 | NA | NA | NA | 0.009300 | -0.03532 | 0.00398 |
| **1996-03-31** | 0.0155 | NA | 0.0258 | -0.0098 | NA | NA | NA | 0.009600 | -0.01057 | 0.00371 |
| **1996-04-30** | -0.0091 | NA | 0.0449 | 0.0236 | NA | NA | NA | 0.014700 | -0.01739 | 0.00428 |
| **1996-05-31** | 0.0076 | NA | 0.0353 | 0.0028 | NA | NA | NA | 0.025800 | -0.00543 | 0.00443 |
| **1996-06-30** | -0.0039 | NA | -0.0303 | -0.0019 | NA | NA | NA | 0.003800 | 0.01507 | 0.00412 |
| **1996-07-31** | -0.0231 | NA | -0.0337 | -0.0446 | NA | NA | NA | -0.044200 | -0.00100 | 0.00454 |
| **1996-08-31** | 0.0395 | -0.0001 | 0.0461 | 0.0351 | NA | NA | NA | 0.021100 | -0.00448 | 0.00451 |
| **1996-09-30** | 0.0147 | 0.1002 | 0.0653 | 0.0757 | NA | NA | NA | 0.056300 | 0.02229 | 0.00470 |
| **1996-10-31** | 0.0288 | 0.0338 | 0.0395 | -0.0180 | NA | NA | NA | 0.027600 | 0.02869 | 0.00428 |
| **1996-11-30** | 0.0156 | 0.0737 | 0.0666 | 0.0458 | NA | NA | NA | 0.075600 | 0.02797 | 0.00427 |
| **1996-12-31** | 0.0176 | 0.0298 | 0.0214 | 0.0439 | NA | NA | NA | -0.019800 | -0.02094 | 0.00442 |
| **1997-01-31** | 0.0212 | 0.0794 | 0.0771 | 0.0437 | NA | NA | 0.0281 | 0.062500 | -0.00055 | 0.00457 |

출처 : 교수자 제공

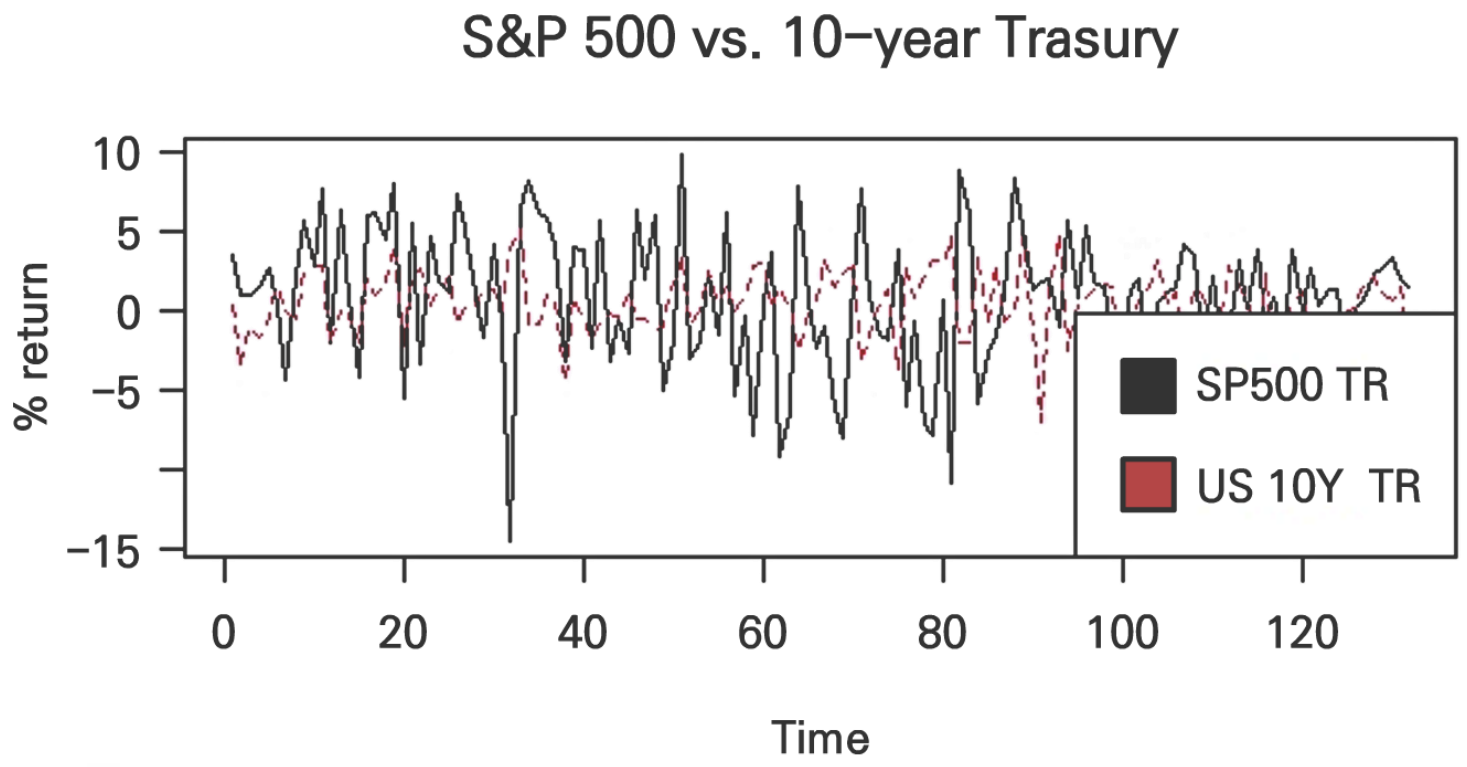
## Warm up with pre-installed R data set



## Multivariate Plot

```
sp500.10yrT.ret <- managers[,8:9]*100
matplot(sp500.10yrT.ret,
        main="S&P 500 vs. 10-year Treasury",
        ylab="% return",
        xlab="time",
        type='l')
legend("bottomright", colnames(sp500.10yrT.ret),
col=seq_len(ncol(sp500.10yrT.ret))
       ,cex=0.8,fill=seq_len(sp500.10yrT.ret))
```

## Multivariate Plot



# Beta Estimation

- **Single index model**

$$r_{i,t} - r_f = \alpha_i + \beta_i\left[r_{m,t} - r_f\right] + \epsilon_{i,t}$$

- **CAPM**

$$E(r_i) = r_f + \beta_i\left[E(r_m) - r_f\right]$$

- **CAPM empirical model: same as single index model**

```
ri <- managers$EDHEC
rm <- managers$SP500
```

- **## assume that the riskfree rate is zero**

```
reg = lm(ri ~ rm)
summary(reg)
```

## Result

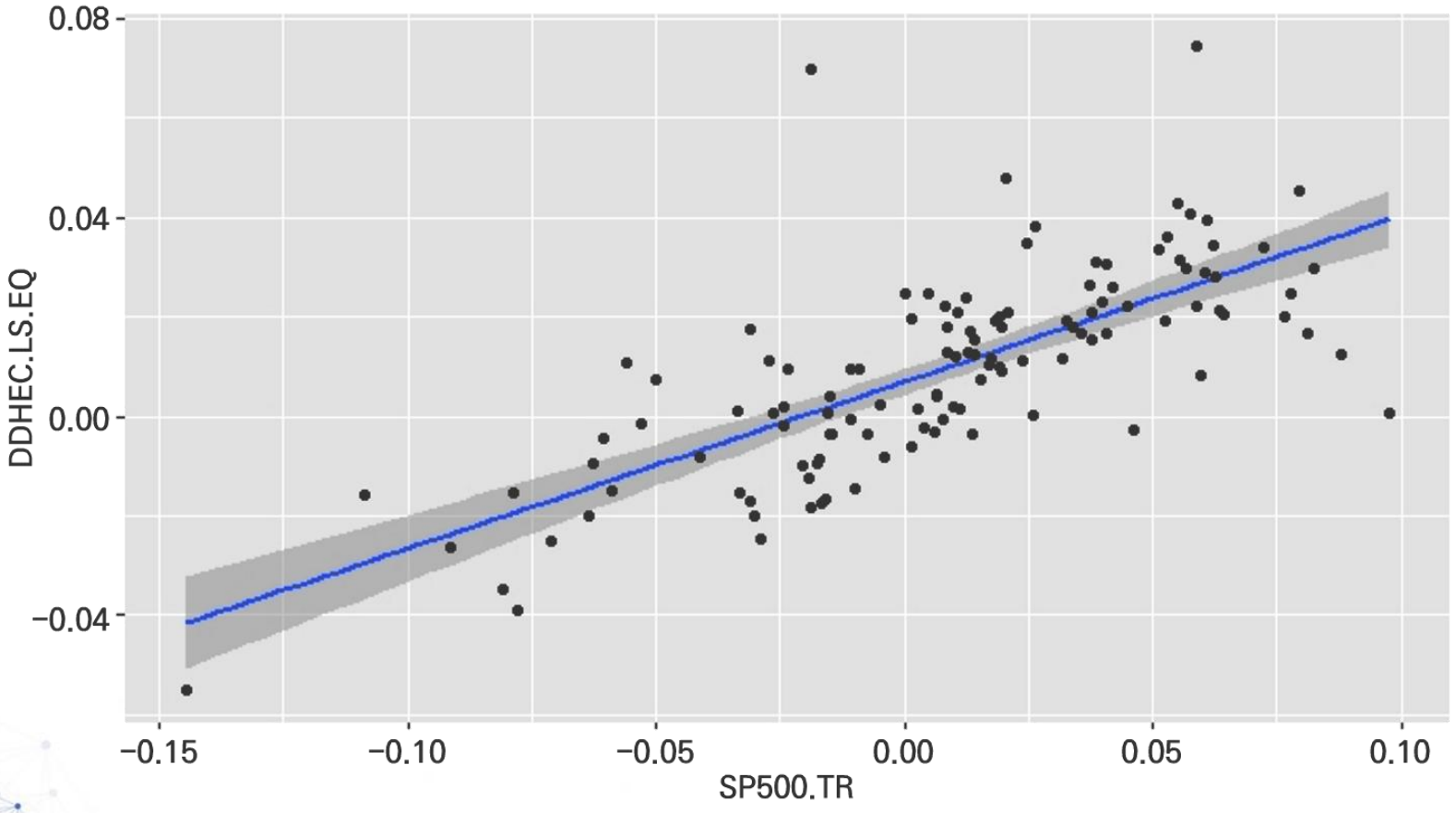
```
> reg = lm(ri ~ rm)
> summary(reg)

Call:
lm(formula = ri ~ rm)

Residuals:
      Min        1Q    Median        3Q       Max
-0.039160 -0.008253 -0.000801  0.006788  0.069297

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.006944   0.001307   5.314 5.14e-07 ***
rm          0.335542   0.029164  11.505  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0141 on 118 degrees of freedom
  (결측으로 인하여 12개의 관측치가 삭제되었습니다.)
Multiple R-squared:  0.5287,    Adjusted R-squared:  0.5247
F-statistic: 132.4 on 1 and 118 DF,  p-value: < 2.2e-16
```

출처 : 교수자 제공

## Result

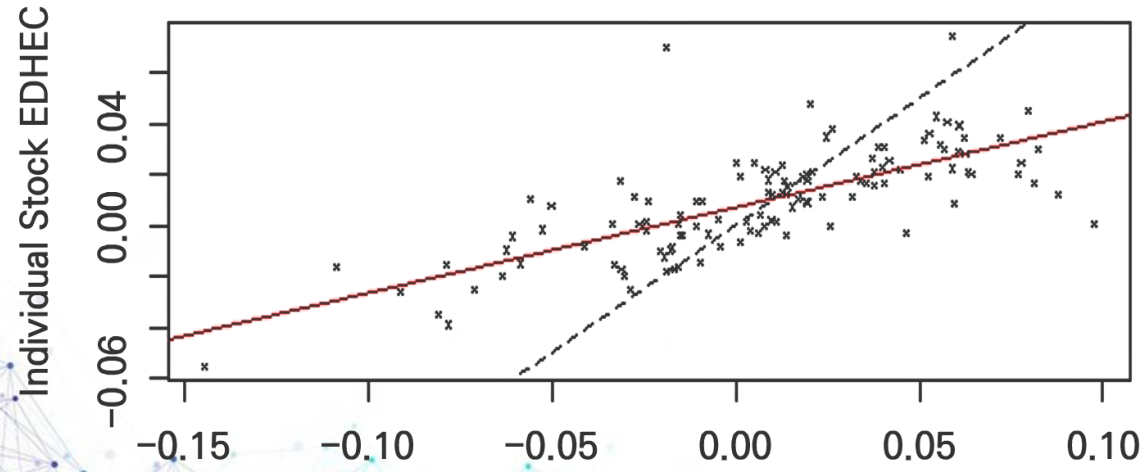
```
testdata <- cbind(rm, ri)
install.packages("ggpubr")
library(ggpubr)
g <- ggplot(data=testdata, aes(x = SP500.TR, y =
EDHEC.LS.EQ)) +
  geom_smooth(method="lm") +
  geom_point() +
  stat_regline_equation(label.x=-0.1, label.y=0.05)
+
  stat_cor(aes(label=..rr.label..), label.x=-0.1,
label.y=0.03)
```

## Result



## Another Plotting

```
plot(as.numeric(rm),
as.numeric(ri), pch=4, cex=0.3,
     xlab = "S&P 500",
ylab="Individual Stock EDHEC")
abline(a=0, b=1, lty=2)
abline(reg, col='red')
```



## Mean-Variance Analysis

- MV analysis is based on optimization mathematics: maximize expected return & minimize variance
- There are many different optimization technique in R, just like other programming languages. Or you can just write up an optimization function with a programming language that you like to use, i.e., Julia, Matlab, C++, etc.
- However, there are some packages or libraries that we can easily use without worries for most problems.
- In R, we can probably use `slsqp()` from nloptr package, `solve.QP()` from quadprog package for fundamental computation.
- Or one can use `optimalPortfolio()` from `RiskPortfolios` package.

## What package?

- We first use widely used RiskPortfolios package and then proceed to the package, IntroCompFinR, that is the main package that we use for the whole MV analysis

## Import Data

```
individual_ret <- read.csv("KSE_listed_returns.csv")
mkt_ret <- read.csv("kospi_ret_all.csv")
colnames(individual_ret) <- individual_ret[1,]
individual_ret <- individual_ret[-1, ]
individual_ret[,2] <- as.Date(individual_ret[,2])
riskfree <- read_excel("KTreasury_3yr.xlsx")
riskfree <- transpose(riskfree[,5:ncol(riskfree)])
# Note that riskfree rate for 2022 Sep. does not exist yet
```

# Data Management

## Do this

```
class(individual_ret),class(individual_ret[,10])
```

## Convert individual_ret to numerical values

```
colchange <- c(3:ncol(individual_ret))
for (i in colchange){
  individual_ret[,i] <- as.numeric(individual_ret[,i])
}
```

## MV Analysis with RiskPortfolios

```
library(PortfolioAnalytics)
library(RiskPortfolios)
```

- # we randomly select 5 individual stocks → 5 can vary

```
set.seed(80) ## For the reproducibility
picker <- floor(runif(6,3,ncol(individual_ret))
stocks_ret <- individual_ret[,picker]
stocks_ret1 <- stocks_ret[ ,
colSums(is.na(stocks_ret))==0]
```

## Construct a Risky Stock Universe

```
risky_stocks <-
cbind(individual_ret[,2],stocks_ret1,mkt_ret[,3])
colnames(risky_stocks) <- c("Dates",
colnames(stocks_ret1), "KOSPI Ret")
stockdates <- risky_stocks[,1]
risky_stocks <-
risky_stocks[,2:ncol(risky_stocks)]/100
Note: we convert % numbers into decimal
```

- Note: we convert % numbers into decimal

## Function: optimalPortfolio

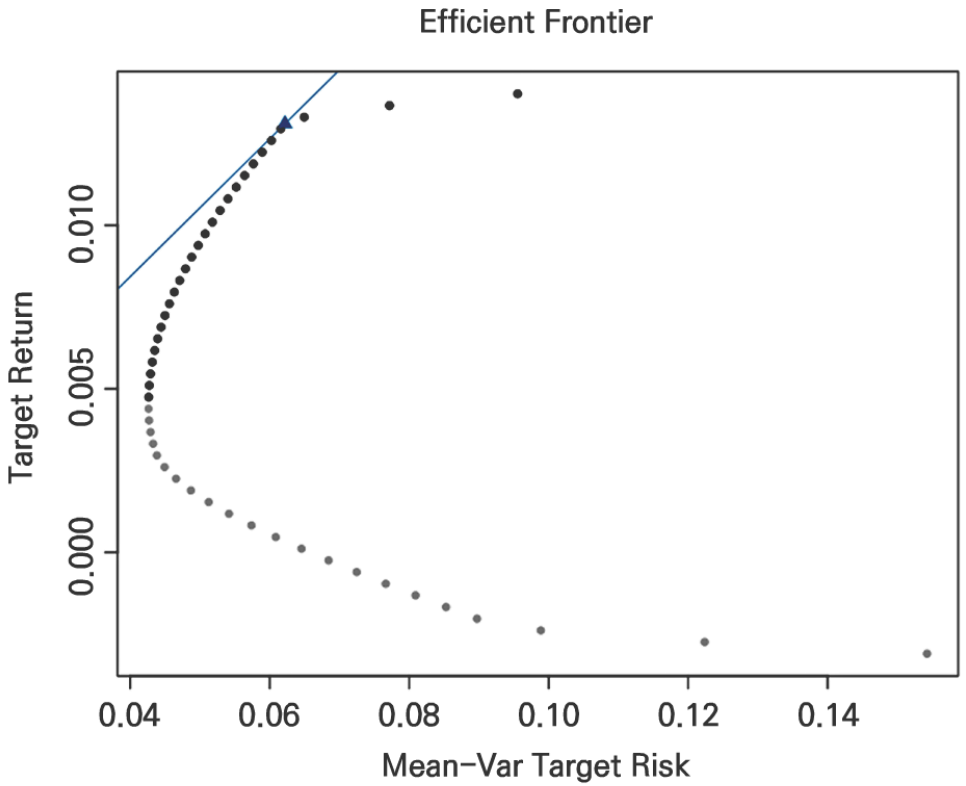
```
covmat <- cov(risky_stocks) # create cov-var
matrix
w_mvp <- optimalPortfolio(cov.mat,
                          control = list(type =
'minvol')) %>%
  round(., 4) %>%
  setNames(colnames(risky_stocks))
print(w_mvp)
```

```
> print(w_mvp)
    KSS해운          웅진 LG디스플레이       삼성SDI    KOSPI Ret
      0.1335      -0.0077      -0.0699       0.0220       0.9222
```

출처 : 교수자 제공

## Efficient Frontier

```
tp <-
tangencyPortfolio(as.timeSeries
(risky_stocks))
frontier <-
portfolioFrontier(as.timeSeries
(risky_stocks))
plot(frontier)
```




- **What is Caveat?**

## Package: IntroCompFinR

```
install.packages("IntroCompFinR", repos="http://R-Forge.R-project.org")
library(IntroCompFinR)
```

### Details of the Package

- # -- getPortfolio: create portfolio object
- # -- globalMin.portfolio: compute global minimum variance portfolio
- # -- efficient.portfolio: compute mvp subject to target return
- # -- tangency.portfolio: compute tangency portfolio
- # -- efficient.frontier: compute efficient frontier of risky assets

## Package: IntroCompFinR

```
er <- colMeans(risky_stocks) # average stock returns -- E(r)
rfree <- mean(riskfree[,1]/1200) # r_f -- set risk free asset: monthly and
decimal
ew <- rep(1,ncol(risky_stocks))/ncol(risky_stocks) # equal weights portfolio
equalWeights.portfolio = getPortfolio(er=er, cov.mat=covmat, weights=ew)
class(equalWeights.portfolio)
```

```
> equalWeights.portfolio
Call:
getPortfolio(er = er, cov.mat = covmat, weights = ew)

Portfolio expected return:     0.004885229
Portfolio standard deviation:  0.06268657
Portfolio weights:
     KSS해운          웅진 LG디스플레이       삼성SDI     KOSPI Ret
         0.2           0.2           0.2           0.2           0.2
```

출처 : 교수자 제공

## GMVP

```
gmin.port <- globalMin.portfolio(er, covmat)
gmin.port
```

```
> gmin.port
Call:
globalMin.portfolio(er = er, cov.mat = covmat)

Portfolio expected return:     0.004973064
Portfolio standard deviation:  0.04221948
Portfolio weights:
     KSS해운          웅진 LG디스플레이       삼성SDI     KOSPI Ret
      0.1335       -0.0077       -0.0699        0.0220        0.9222
```

출처 : 교수자 제공

## Efficient Portfolio with Target Return

```
target.ret <- er[1]
eff.port <- efficient.portfolio(er, covmat, target.ret)
eff.port
```

```
> eff.port
Call:
efficient.portfolio(er = er, cov.mat = covmat, target.return = target.ret)

Portfolio expected return:     0.0125
Portfolio standard deviation:  0.05581672
Portfolio weights:
     KSS해운          웅진 LG디스플레이       삼성SDI     KOSPI Ret
      0.4949       -0.0448       -0.2262        0.3085        0.4676
```

출처 : 교수자 제공

## Tangent Portfolio

```
tan.port <- tangency.portfolio(er, covmat, rfree)
tan.port
```

```
> tan.port
Call:
tangency.portfolio(er = er, cov.mat = covmat, risk.free = rfree)

Portfolio expected return:     0.02957863
Portfolio standard deviation:  0.1266004
Portfolio weights:
     KSS해운          웅진 LG디스플레이       삼성SDI    KOSPI Ret
      1.3150       -0.1289       -0.5807        0.9586       -0.5640
```

출처 : 교수자 제공

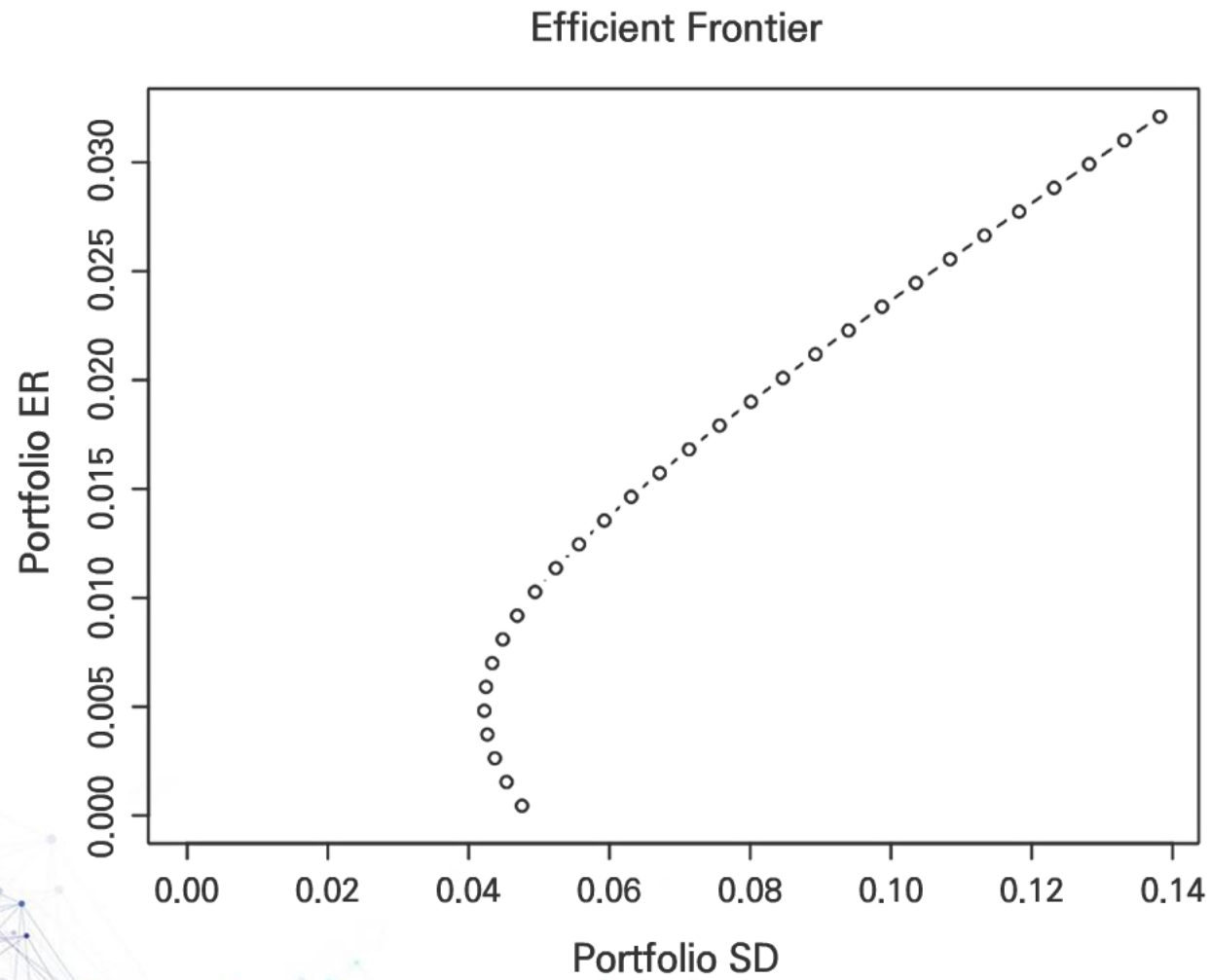
## Efficient Frontier

```
ef <- efficient.frontier(er, covmat, alpha.min=-2,
                         alpha.max=1.5, nport=30)
ef
```

```
> ef
Call:
efficient.frontier(er = er, cov.mat = covmat, nport = 30, alpha.min = -2,
    alpha.max = 1.5)

Frontier portfolios' expected returns and standard deviations
   port 1 port 2 port 3 port 4 port 5 port 6 port 7 port 8 port 9 port 10 port 11 port 12
ER 0.0321 0.0310 0.0299 0.0288 0.0277 0.0266 0.0256 0.0245 0.0234  0.0223  0.0212  0.0201
SD 0.1382 0.1332 0.1282 0.1232 0.1182 0.1133 0.1084 0.1035 0.0987  0.0940  0.0893  0.0846
   port 13 port 14 port 15 port 16 port 17 port 18 port 19 port 20 port 21 port 22 port 23
ER  0.0190  0.0179  0.0168  0.0157  0.0146  0.0135  0.0125  0.0114  0.0103  0.0092  0.0081
SD  0.0801  0.0757  0.0713  0.0671  0.0631  0.0593  0.0557  0.0524  0.0494  0.0469  0.0448
   port 24 port 25 port 26 port 27 port 28 port 29 port 30
ER  0.0070  0.0059  0.0048  0.0037  0.0026  0.0015  0.0005
SD  0.0433  0.0425  0.0422  0.0427  0.0437  0.0454  0.0476
```

출처 : 교수자 제공

## Efficient Frontier



## Adjustment with the Plot

```
par(family="AppleGothic") # Dealing with broken
Korean fonts on Mac
plot(ef, plot.assets=T)
points(gmin.port$sd, gmin.port$er, col="blue")
points(tan.port$sd, tan.port$er, col="red")
sr.tan = (tan.port$er - rfree)/tan.port$sd
abline(a=rfree, b=sr.tan)
```

## Adjustment with the Plot