```
#################### Financial Econometrics - Spring 2021 ######################
#################### Karoll Gómez Portilla ######################
#################### Universidad Nacional de Colombia ######################
#################### Lab 3: CAPM Model ######################

installed <- dir(.libPaths()) # To check installed packages

# Packages
library(zoo)
library(sandwich)
library(gmm)
# Data set contains daily returns of twenty selected stocks from January 1993 to February 2009, the risk-free rate and the three factors of Fama and French
data(Finance)
head(Finance)
# Generating the CAPM variables
r <- Finance[1:500,1:5]
rm <- Finance[1:500,"rm"]
rf <- Finance[1:500,"rf"]
z <- as.matrix(r-rf)
zm <- as.matrix(rm-rf)
# Estimating the model by GMM
capm <- gmm(z~zm,x=zm)
summary(capm)
coef(capm)
# Testing the CAPM
library(carData)
library(car)
# Generating matrix R(5X10)
R <- cbind(diag(5),matrix(0,5,5))
# Generating matrix C(1X5)
c <- rep(0,5)
linearHypothesis(capm,R,c,test = "Chisq")
# Another way: estimating the restricted model and applying the J-test of over-identifying restrictions
capm2<-gmm(z~zm-1,cbind(1,zm))
specTest(capm2)

# Estimating the model by OLS
capm.ols <- lm(z~zm)
summary(capm.ols)

# Created CAPM Function
capm.tstats = function(r, mkrt) {
  # Fiting CAPM
  capm.fit = lm(r ~ mkrt)
  # Extract summary info
  capm.summary = summary(capm.fit)
  # Retrieve t-stat
  t.stat = coef(capm.summary)[1, 3]
  t.stat
}

tstats = apply(z,2, capm.tstats, zm)
tstats
# Test Hypothesis for 5% CI: H0: alpha=0
abs(tstats) > 2 # None is statistically significant
any(abs(tstats) > 2)


# Example for one stock
capm.ols.WMK = lm(z[,1]~zm)
# Plot CAPM
plot(zm, z[,1])
# Plot CAPM regression estimate
abline(capm.ols.WMK, col="blue")
# Create Axis
abline(h=0,v=0,lty=3)
# Placing beta & tstat values on the plot for APPL
alpha = coef(summary(capm.ols.WMK))[1,1]
a_tstat = coef(summary(capm.ols.WMK))[1,3]
beta = coef(summary(capm.ols.WMK))[2,1]
b_tstat = coef(summary(capm.ols.WMK))[2,3]
legend("topleft", legend=
         c(paste("alpha =",round(alpha,dig=2),"(",round(a_tstat, dig=2),")"),
           paste("beta =",round(beta,dig=2),"(",round(b_tstat,dig=2),")")), cex=1, bty="n")

# Beta risk compute by using the formula
beta = cov(r,rm)/var(rm)
beta

# Compute betas using a function
capm.betas = function(r,market) {
  capm.fit = lm(r~market)
  # Fit capm regression
  capm.beta = coef(capm.fit)[2]
  # Extract coefficients
  capm.beta
}
betas = apply(z,2,FUN=capm.betas,zm)
betas

# Generate mean returns
mu.hat = colMeans(r)
mu.hat

# Plot expected returns versus betas
plot(betas,mu.hat,main="Expected Return vs. Beta")

##### Estimate regression of Expected Return vs. Beta
sml.fit = lm(mu.hat~betas)
sml.fit
summary(sml.fit)

# Ideally intercept is zero and equals the excess market return
mean(zm)

# Plot Fitted Security MArket Line SML
plot(betas,mu.hat,main="Estimated SML")
abline(sml.fit)
legend("topright",1, "Estimated SML",1)




#################### Another example (Take from www.r-bloggers.com) #################
####### A Gentle Introduction to Finance using R: Efficient Frontier and CAPM #######
```

```r
library(data.table)
library(scales)
library(ggplot2)
library(labeling)
library(digest)
library(data.table)
library(scales)
library(ggplot2)

link <- "https://raw.githubusercontent.com/DavZim/Efficient_Frontier/master/data/fin_data.csv"
dt <- data.table(read.csv(link))
dt[, date := as.Date(date)]

# create indexed values
dt[, idx_price := price/price[1], by = ticker]

# plot the indexed values
ggplot(dt, aes(x = date, y = idx_price, color = ticker)) +
geom_line() +
# Miscellaneous Formatting
theme_bw() + ggtitle("Price Developments") +
xlab("Date") + ylab("Pricen(Indexed 2000 = 1)") +
scale_color_discrete(name = "Company")

# calculate the arithmetic returns
dt[, ret := price / shift(price, 1) - 1, by = ticker]

# summary table
# take only non-na values
tab <- dt[!is.na(ret), .(ticker, ret)]

# calculate the expected returns (historical mean of returns) and volatility (standard deviation of returns)
tab <- tab[, .(er = round(mean(ret), 4), sd = round(sd(ret), 4)), by = "ticker"]

ggplot(tab, aes(x = sd, y = er, color = ticker)) +
geom_point(size = 5) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Risk-Return Tradeoff") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, 0.03)) +
scale_x_continuous(label = percent, limits = c(0, 0.1))

### Calculating the Risk-Return Tradeoff of a Portfolio


# load the data
link <- "https://raw.githubusercontent.com/DavZim/Efficient_Frontier/master/data/mult_assets.csv"
df <- data.table(read.csv(link))

# calculate the necessary values:
# I) expected returns for the two assets
er_x <- mean(df$x)
er_y <- mean(df$y)

# II) risk (standard deviation) as a risk measure
sd_x <- sd(df$x)
sd_y <- sd(df$y)

# III) covariance
cov_xy <- cov(df$x, df$y)

# create 1000 portfolio weights (omegas)
x_weights <- seq(from = 0, to = 1, length.out = 1000)

# create a data.table that contains the weights for the two assets
two_assets <- data.table(wx = x_weights, wy = 1 - x_weights)

# calculate the expected returns and standard deviations for the 1000 possible portfolios
two_assets[, ':=' (er_p = wx * er_x + wy * er_y,
sd_p = sqrt(wx^2 * sd_x^2 +
wy^2 * sd_y^2 +
2 * wx * (1 - wx) * cov_xy))]
two_assets

# lastly plot the values
ggplot() +
geom_point(data = two_assets, aes(x = sd_p, y = er_p, color = wx)) +
geom_point(data = data.table(sd = c(sd_x, sd_y), mean = c(er_x, er_y)),
aes(x = sd, y = mean), color = "red", size = 3, shape = 18) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Possible Portfolios with Two Risky Assets") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, max(two_assets$er_p) * 1.2)) +
scale_x_continuous(label = percent, limits = c(0, max(two_assets$sd_p) * 1.2)) +
scale_color_continuous(name = expression(omega[x]), labels = percent)


#### Adding a Third Asset
# load the data
link <- "https://raw.githubusercontent.com/DavZim/Efficient_Frontier/master/data/mult_assets.csv"
df <- data.table(read.csv(link))

# calculate the necessary values:
# I) expected returns for the two assets
er_x <- mean(df$x)
er_y <- mean(df$y)
er_z <- mean(df$z)

# II) risk (standard deviation) as a risk measure
sd_x <- sd(df$x)
sd_y <- sd(df$y)
sd_z <- sd(df$z)

# III) covariance
cov_xy <- cov(df$x, df$y)
cov_xz <- cov(df$x, df$z)
cov_yz <- cov(df$y, df$z)

# create portfolio weights (omegas)
x_weights <- seq(from = 0, to = 1, length.out = 1000)

# create a data.table that contains the weights for the three assets
three_assets <- data.table(wx = rep(x_weights, each = length(x_weights)),
wy = rep(x_weights, length(x_weights)))
```

```r
three_assets[, wz := 1 - wx - wy]


# calculate the expected returns and standard deviations for the 1000 possible portfolios
three_assets[, ':=' (er_p = wx * er_x + wy * er_y + wz * er_z,
sd_p = sqrt(wx^2 * sd_x^2 +
wy^2 * sd_y^2 +
wz^2 * sd_z^2 +
2 * wx * wy * cov_xy +
2 * wx * wz * cov_xz +
2 * wy * wz * cov_yz))]

# take out cases where we have negative weights (shortselling)
three_assets <- three_assets[wx >= 0 & wy >= 0 & wz >= 0]
three_assets

# lastly plot the values
ggplot() +
geom_point(data = three_assets, aes(x = sd_p, y = er_p, color = wx - wz)) +
geom_point(data = data.table(sd = c(sd_x, sd_y, sd_z), mean = c(er_x, er_y, er_z)),
aes(x = sd, y = mean), color = "red", size = 3, shape = 18) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Possible Portfolios with Three Risky Assets") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, max(three_assets$er_p) * 1.2)) +
scale_x_continuous(label = percent, limits = c(0, max(three_assets$sd_p) * 1.2)) +
scale_color_gradientn(colors = c("red", "blue", "yellow"),
name = expression(omega[x] - omega[z]), labels = percent)

# Calculating the Efficient Frontier
calcEFParams <- function(rets) {

retbar <- colMeans(rets, na.rm = T)
covs <- var(rets, na.rm = T) # calculates the covariance of the returns
invS <- solve(covs)
i <- matrix(1, nrow = length(retbar))

alpha <- t(i) %*% invS %*% i
beta <- t(i) %*% invS %*% retbar
gamma <- t(retbar) %*% invS %*% retbar
delta <- alpha * gamma - beta * beta

retlist <- list(alpha = as.numeric(alpha),
beta = as.numeric(beta),
gamma = as.numeric(gamma),
delta = as.numeric(delta))

return(retlist)
}

# load data
link <- "https://raw.githubusercontent.com/DavZim/Efficient_Frontier/master/data/mult_assets.csv"
df <- data.table(read.csv(link))

abcds <- calcEFParams(df)
abcds

calcEFValues <- function(x, abcd, upper = T) {
alpha <- abcd$alpha
beta <- abcd$beta
gamma <- abcd$gamma
delta <- abcd$delta

if (upper) {
retval <- beta / alpha + sqrt((beta / alpha) ^ 2 - (gamma - delta * x ^ 2) / (alpha))
} else {
retval <- beta / alpha - sqrt((beta / alpha) ^ 2 - (gamma - delta * x ^ 2) / (alpha))
}

return(retval)
}

# calculate the risk-return tradeoff the two assets (for plotting the points)
df_table <- melt(df)[, .(er = mean(value),
sd = sd(value)), by = variable]

# plot the values
ggplot(df_table, aes(x = sd, y = er)) +
# add the stocks
geom_point(size = 4, color = "red", shape = 18) +
# add the upper efficient frontier
stat_function(fun = calcEFValues, args = list(abcd = abcds, upper = T), n = 10000,
color = "red", size = 1) +
# add the lower "efficient" frontier
stat_function(fun = calcEFValues, args = list(abcd = abcds, upper = F), n = 10000,
color = "blue", size = 1) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Efficient Frontier with Short-Selling") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, max(df_table$er) * 1.2)) +
scale_x_continuous(label = percent, limits = c(0, max(df_table$sd) * 1.2))

# Without Short-Selling
library(tseries)
link <- "https://raw.githubusercontent.com/DavZim/Efficient_Frontier/master/data/mult_assets.csv"
df <- data.table(read.csv(link))

df_table <- melt(df)[, .(er = mean(value),
sd = sd(value)), by = variable]

er_vals <- seq(from = min(df_table$er), to = max(df_table$er), length.out = 1000)

# find an optimal portfolio for each possible possible expected return
# (note that the values are explicitly set between the minimum and maximum of the expected returns per asset)
sd_vals <- sapply(er_vals, function(er) {
op <- portfolio.optim(as.matrix(df), er)
return(op$ps)
})

plot_dt <- data.table(sd = sd_vals, er = er_vals)

# find the lower and the upper frontier
minsd <- min(plot_dt$sd)
minsd_er <- plot_dt[sd == minsd, er]
plot_dt[, efficient := er >= minsd_er]
plot_dt
```

```r
ggplot() +
geom_point(data = plot_dt[efficient == F], aes(x = sd, y = er), size = 0.5, color = "blue") +
geom_point(data = plot_dt[efficient == T], aes(x = sd, y = er), size = 0.5, color = "red") +
geom_point(data = df_table, aes(x = sd, y = er), size = 4, color = "red", shape = 18) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Efficient Frontier without Short-Selling") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, max(df_table$er) * 1.2)) +
scale_x_continuous(label = percent, limits = c(0, max(df_table$sd) * 1.2))

# To directly compare the two options we can use the following code.
# combine the data into one plotting data.table called "pdat"
# use plot_dt with constraints
pdat1 <- plot_dt[, .(sd, er, type = "wo_short", efficient)]

# calculate the values without constraints
pdat2lower <- data.table(sd = seq(from = 0, to = max(pdat1$sd) * 1.2, length.out = 1000))
pdat2lower[, ':=' (er = calcEFValues(sd, abcds, F),
type = "short",
efficient = F)]

pdat2upper <- data.table(sd = seq(from = 0, to = max(pdat1$sd) * 1.2, length.out = 1000))
pdat2upper[, ':=' (er = calcEFValues(sd, abcds, T),
type = "short",
efficient = T)]

pdat <- rbindlist(list(pdat1, pdat2upper, pdat2lower))

# plot the values
ggplot() +
geom_line(data = pdat, aes(x = sd, y = er, color = type, linetype = efficient), size = 1) +
geom_point(data = df_table, aes(x = sd, y = er), size = 4, color = "red", shape = 18) +
# Miscellaneous Formatting
theme_bw() + ggtitle("Efficient Frontiers") +
xlab("Volatility") + ylab("Expected Returns") +
scale_y_continuous(label = percent, limits = c(0, max(df_table$er) * 1.2)) +
scale_x_continuous(label = percent, limits = c(0, max(df_table$sd) * 1.2)) +
scale_color_manual(name = "Short-Sells", values = c("red", "blue"), labels = c("Allowed", "Prohibited")) +
scale_linetype_manual(name = "Efficient", values = c(2, 1))
```