# HW4-Group7-Cohort2

*Cohort 2, Group 7 - Hyeuk Jung, Jiaqi Li, Xichen Luo, Huanyu Liu*

*February 24, 2019*

## Problem 1

### Question 1

```r
options(warn = -1)
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(readxl)
library(rugarch)
```

```
## Loading required package: parallel

##
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':
##
##     sigma
```

```r
library(fGarch)
```

```
## Loading required package: timeDate

## Loading required package: timeSeries

## Loading required package: fBasics
```

```r
library(sandwich)

##### Problem 1: Risk Management using Value-at-Risk and GARCH Models ----
raw <- read_xlsx("Currency_fund_prices.xlsx") %>% as.data.table; gc()
```

```
##           used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells 2213278 118.3    4400554 235.1         NA  3445933 184.1
## Vcells 3657052  28.0    8388608  64.0      16384  6164544  47.1
```

```r
data <- raw[, -c(2:5)]
data$lag_adj <- shift(data$`Adj Close`, 1)
data$log_ret <- log(data$`Adj Close` / data$lag_adj)

### Q1. Specify and estimate a parsimonious model for the conditional volatility of the daily log return

# GARCH(1,1)
garch_z <- ugarchspec(variance.model = list(model = "fGARCH",
                                            submodel = "GARCH",
                                            garchOrder = c(1,1)),
                         mean.model = list(armaOrder = c(0,0),
                                           include.mean = T),
                         distribution.model = "norm")
garch_t <- ugarchspec(variance.model = list(model = "fGARCH",
                                            submodel = "GARCH",
                                            garchOrder = c(1,1)),
                         mean.model = list(armaOrder = c(0,0),
                                           include.mean = T),
                         distribution.model = "std")

# I-GARCH(1,1)
igarch_z <- ugarchspec(variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                      mean.model = list(armaOrder = c(0,0), include.mean = T),
                      distribution.model = "norm")
igarch_t <- ugarchspec(variance.model = list(model = "iGARCH", garchOrder = c(1,1)),
                      mean.model = list(armaOrder = c(0,0), include.mean = T),
                      distribution.model = "std")

# E-GARCH(1,1)
egarch_z <- ugarchspec(variance.model = list(model = "eGARCH", garchOrder = c(1,1)),
                      mean.model = list(armaOrder = c(0,0), include.mean = T),
                      distribution.model = "norm")
egarch_t <- ugarchspec(variance.model = list(model = "eGARCH", garchOrder = c(1,1)),
                      mean.model = list(armaOrder = c(0,0), include.mean = T),
                      distribution.model = "std")

# GJR-GARCH(1,1)
gjrgarch_z <- ugarchspec(variance.model = list(model = "gjrGARCH", garchOrder = c(1,1)),
                       mean.model = list(armaOrder = c(0,0), include.mean = T),
                       distribution.model = "norm")
gjrgarch_t <- ugarchspec(variance.model = list(model = "gjrGARCH", garchOrder = c(1,1)),
                       mean.model = list(armaOrder = c(0,0), include.mean = T),
                       distribution.model = "std")

# Fitting the date to the models
test_data <- data[-1, "log_ret"]
garchfit_z <- ugarchfit(garch_z, test_data)
garchfit_t <- ugarchfit(garch_t, test_data)
igarchfit_z <- ugarchfit(igarch_z, test_data)
igarchfit_t <- ugarchfit(igarch_t, test_data)
egarchfit_z <- ugarchfit(egarch_z, test_data)
egarchfit_t <- ugarchfit(egarch_t, test_data)
gjrgarchfit_z <- ugarchfit(gjrgarch_z, test_data)
```

```r
gjrgarchfit_t <- ugarchfit(gjrgarch_t, test_data)

# Combine Information Criterion from each model
result <- do.call("rbind", list(infocriteria(garchfit_z)[1:2,],
                                infocriteria(garchfit_t)[1:2,],
                                infocriteria(igarchfit_z)[1:2,],
                                infocriteria(igarchfit_t)[1:2,],
                                infocriteria(egarchfit_z)[1:2,],
                                infocriteria(egarchfit_t)[1:2,],
                                infocriteria(gjrgarchfit_z)[1:2,],
                                infocriteria(gjrgarchfit_t)[1:2,])
                  ) #[1:2]

row.names(result) <- c("GARCH(1,1)-z", "GARCH(1,1)-t",
                       "I-GARCH(1,1)-z", "I-GARCH(1,1)-t",
                       "E-GARCH(1,1)-z", "E-GARCH(1,1)-t",
                       "GJR-GARCH(1,1)-z", "GJR-GARCH(1,1)-t")

#############################################################
#2,2 normal Garch-------------------------------------------
garch22n = ugarchspec(variance.model = list(model = "fGARCH",
                                            submodel = "GARCH",
                                            garchOrder = c(2,2)),
                      mean.model = list(armaOrder = c(0,0),
                                        include.mean = T),
                      distribution.model = "norm")
fit22n = ugarchfit(garch22n,test_data)

#2,2 std Garch-------------------------------------------
garch22t = ugarchspec(variance.model = list(model = "fGARCH",
                                            submodel = "GARCH",
                                            garchOrder = c(2,2)),
                      mean.model = list(armaOrder = c(0,0),
                                        include.mean = T),
                      distribution.model = "std")
fit22t = ugarchfit(garch22t,test_data)

#2,2 normal iGarch-------------------------------------------
garch22n_i = ugarchspec(variance.model = list(model = "iGARCH",
                                              submodel = "GARCH",
                                              garchOrder = c(2,2)),
                        mean.model = list(armaOrder = c(0,0),
                                          include.mean = T),
                        distribution.model = "norm")
fit22n_i = ugarchfit(garch22n_i,test_data)

#2,2 normal iGarch-------------------------------------------
garch22t_i = ugarchspec(variance.model = list(model = "iGARCH",
                                              submodel = "GARCH",
                                              garchOrder = c(2,2)),
                        mean.model = list(armaOrder = c(0,0),
                                          include.mean = T),
                        distribution.model = "std")
```

```r
fit22t_i = ugarchfit(garch22t_i,test_data)

#2,2 normal eGarch--------------------------------------------
garch22n_e = ugarchspec(variance.model = list(model = "eGARCH",
                                              submodel = "GARCH",
                                              garchOrder = c(2,2)),
                        mean.model = list(armaOrder = c(0,0),
                                          include.mean = T),
                        distribution.model = "norm")
fit22n_e = ugarchfit(garch22n_e,test_data)

#2,2 normal eGarch--------------------------------------------
garch22t_e = ugarchspec(variance.model = list(model = "eGARCH",
                                              submodel = "GARCH",
                                              garchOrder = c(2,2)),
                        mean.model = list(armaOrder = c(0,0),
                                          include.mean = T),
                        distribution.model = "std")
fit22t_e = ugarchfit(garch22t_e,test_data)

#2,2 normal gjrGarch--------------------------------------------
garch22n_gjr = ugarchspec(variance.model = list(model = "gjrGARCH",
                                                submodel = "GARCH",
                                                garchOrder = c(2,2)),
                          mean.model = list(armaOrder = c(0,0),
                                            include.mean = T),
                          distribution.model = "norm")
fit22n_gjr = ugarchfit(garch22n_gjr,test_data)

#2,2 normal gjrGarch--------------------------------------------
garch22t_gjr = ugarchspec(variance.model = list(model = "gjrGARCH",
                                                submodel = "GARCH",
                                                garchOrder = c(2,2)),
                          mean.model = list(armaOrder = c(0,0),
                                            include.mean = T),
                          distribution.model = "std")
fit22t_gjr = ugarchfit(garch22t_gjr,test_data)


order22 = do.call("rbind", list(infocriteria(fit22n)[1:2,],
                                infocriteria(fit22t)[1:2,],
                                infocriteria(fit22n_i)[1:2,],
                                infocriteria(fit22t_i)[1:2,],
                                infocriteria(fit22n_e)[1:2,],
                                infocriteria(fit22t_e)[1:2,],
                                infocriteria(fit22n_gjr)[1:2,],
                                infocriteria(fit22t_gjr)[1:2,]))

row.names(order22) <- c("GARCH(2,2)-z", "GARCH(2,2)-t",
                       "I-GARCH(2,2)-z", "I-GARCH(2,2)-t",
                       "E-GARCH(2,2)-z", "E-GARCH(2,2)-t",
                       "GJR-GARCH(2,2)-z", "GJR-GARCH(2,2)-t")
```

```
############################################################

All = rbind(result,order22)
print(All)
```

```
##                       Akaike     Bayes
## GARCH(1,1)-z     -7.161066 -7.151230
## GARCH(1,1)-t     -7.226537 -7.214243
## I-GARCH(1,1)-z   -7.162009 -7.154633
## I-GARCH(1,1)-t   -7.227331 -7.217495
## E-GARCH(1,1)-z   -7.168923 -7.156628
## E-GARCH(1,1)-t   -7.234664 -7.219910
## GJR-GARCH(1,1)-z -7.163986 -7.151691
## GJR-GARCH(1,1)-t -7.229426 -7.214673
## GARCH(2,2)-z     -7.164782 -7.150029
## GARCH(2,2)-t     -7.227130 -7.209918
## I-GARCH(2,2)-z   -7.159983 -7.147688
## I-GARCH(2,2)-t   -7.225389 -7.210635
## E-GARCH(2,2)-z   -7.170991 -7.151320
## E-GARCH(2,2)-t   -7.233353 -7.211222
## GJR-GARCH(2,2)-z -7.163854 -7.144183
## GJR-GARCH(2,2)-t -7.228251 -7.206121
```

```
which(min(result[,"Akaike"]) == result[,"Akaike"])
```

```
## E-GARCH(1,1)-t
##              6
```

```
which(min(result[,"Bayes"]) == result[,"Bayes"])
```

```
## E-GARCH(1,1)-t
##              6
```

```
which(min(order22[,"Akaike"]) == order22[,"Akaike"])
```

```
## E-GARCH(2,2)-t
##              6
```

```
which(min(order22[,"Bayes"]) == order22[,"Bayes"])
```

```
## E-GARCH(2,2)-t
##              6
```

```
All[c("E-GARCH(1,1)-t","E-GARCH(2,2)-t"),]
```

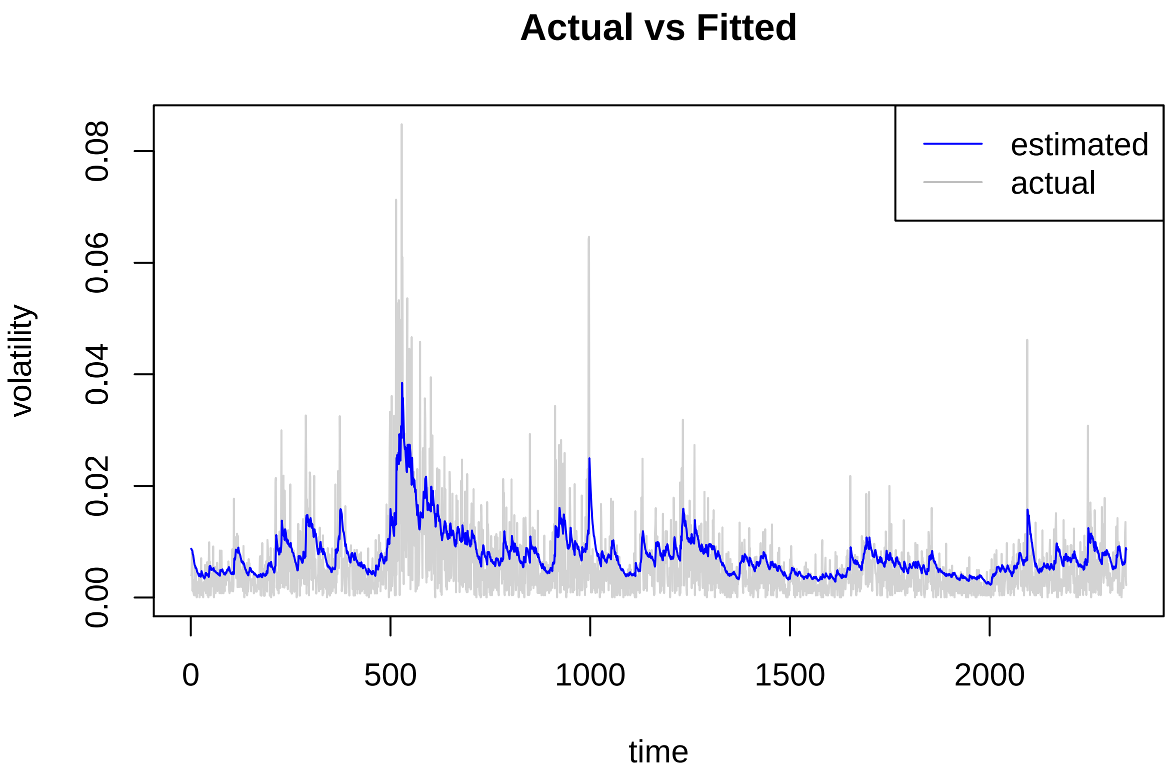
```
##                     Akaike     Bayes
## E-GARCH(1,1)-t -7.234664 -7.219910
## E-GARCH(2,2)-t -7.233353 -7.211222
```

```
# E-GARCH(1,1)-t has the lowest AIC and BIC values
```

**Based on the AIC and BIC, we find that E_Garch model with the order of (1,1) with studnet t distribution fits the conditional volatility of the daily log returns best.**

```
mu = mean(test_data$log_ret)
V=c()
for (i in 1:length(test_data$log_ret)){
  V = c(V,(mu - test_data$log_ret[i])^2)
}
```

```r
plot(abs(test_data$log_ret-mu),
     col = "lightgray", type = "l", xlab = "time",
     ylab = "volatility",
     main = "Actual vs Fitted")
lines(egarchfit_t@fit$sigma, type = "l",col = "blue")
legend("topright",legend = c("estimated","actual"),
       lty = 1, col = c("blue","grey"))
```



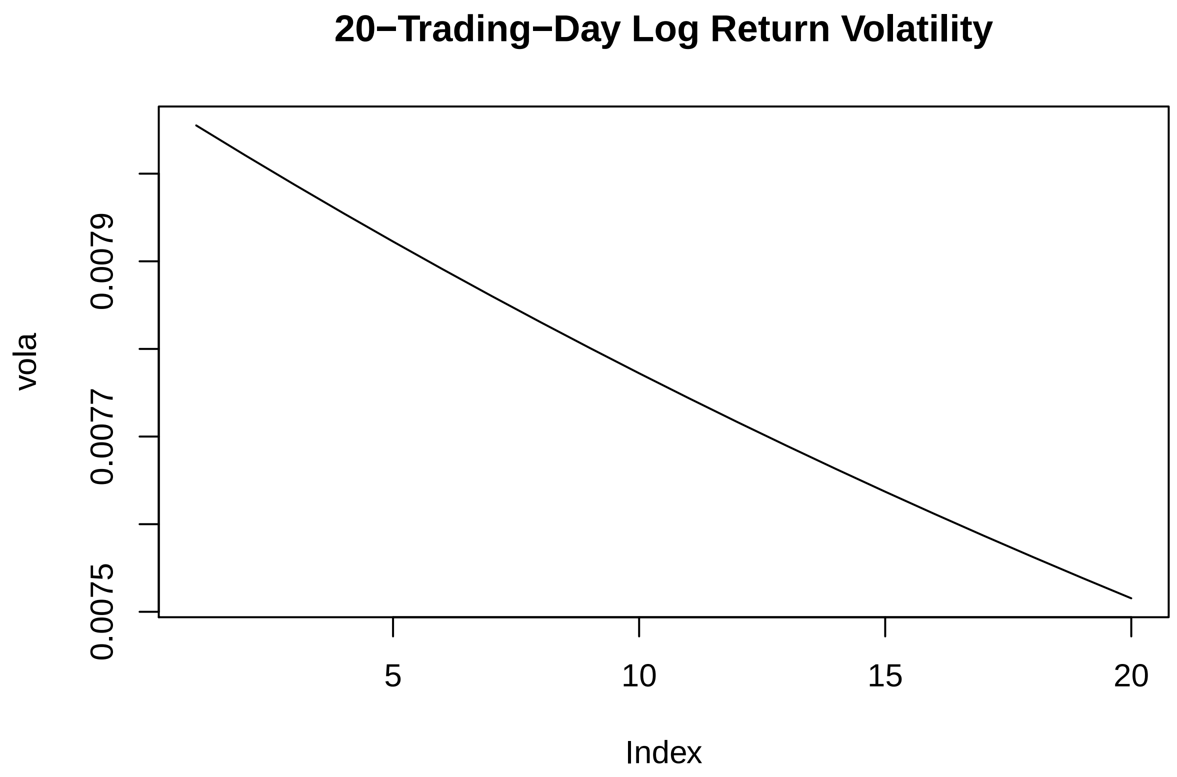
Based on the graph above, we can tell that the model fits the volatility data well.

**Question 2**

```r
### Q2. Develop a forecast for the 20-trading-day log return
#volatility on Jan 11, 2016 (end of day)
# -> sum of each of the 20 individual days' expected variance
# -> sqrt
forecast <- ugarchforecast(egarchfit_t, n.ahead = 20)
vola = as.vector(sigma(forecast))
volatility_pred = matrix(vola, nrow = 1,
                         ncol = length(sigma(forecast)))
colnames(volatility_pred) = c("T+1","T+2","T+3","T+4","T+5",
                              "T+6","T+7","T+8","T+9","T+10",
                              "T+11","T+12","T+13","T+14","T+15",
                              "T+16","T+17","T+18","T+19","T+20")
print(volatility_pred)
```

```
##                T+1         T+2         T+3         T+4         T+5
## [1,] 0.008055066 0.008020843 0.007987361 0.007954599 0.007922542
##                T+6         T+7         T+8         T+9        T+10
## [1,] 0.00789117 0.007860468 0.007830419 0.007801007 0.007772217
##             T+11        T+12        T+13        T+14        T+15
## [1,] 0.007744034 0.007716443 0.00768943 0.007662981 0.007637083
##             T+16        T+17        T+18        T+19        T+20
## [1,] 0.007611723 0.007586889 0.007562567 0.007538747 0.007515416
```

```
#plot(forecast) # have to make a plot selection (unconditional)
plot(vola,
     main = "20-Trading-Day Log Return Volatility",
     type = "l")
```



```
volatility <- sqrt(sum(vola^2))
print(volatility)
```

```
## [1] 0.03474748
```

All the volatilities of each forecasting step are shown above; the exact 20-trading-day log return volatility is approximately **0.035**. The number is calculated as the square root of the sum of each day's variance:

$$\sigma = \sqrt{\sum_{t=1}^{20} Var[log(return)]} = 0.035$$

**Question 2**

**1.**

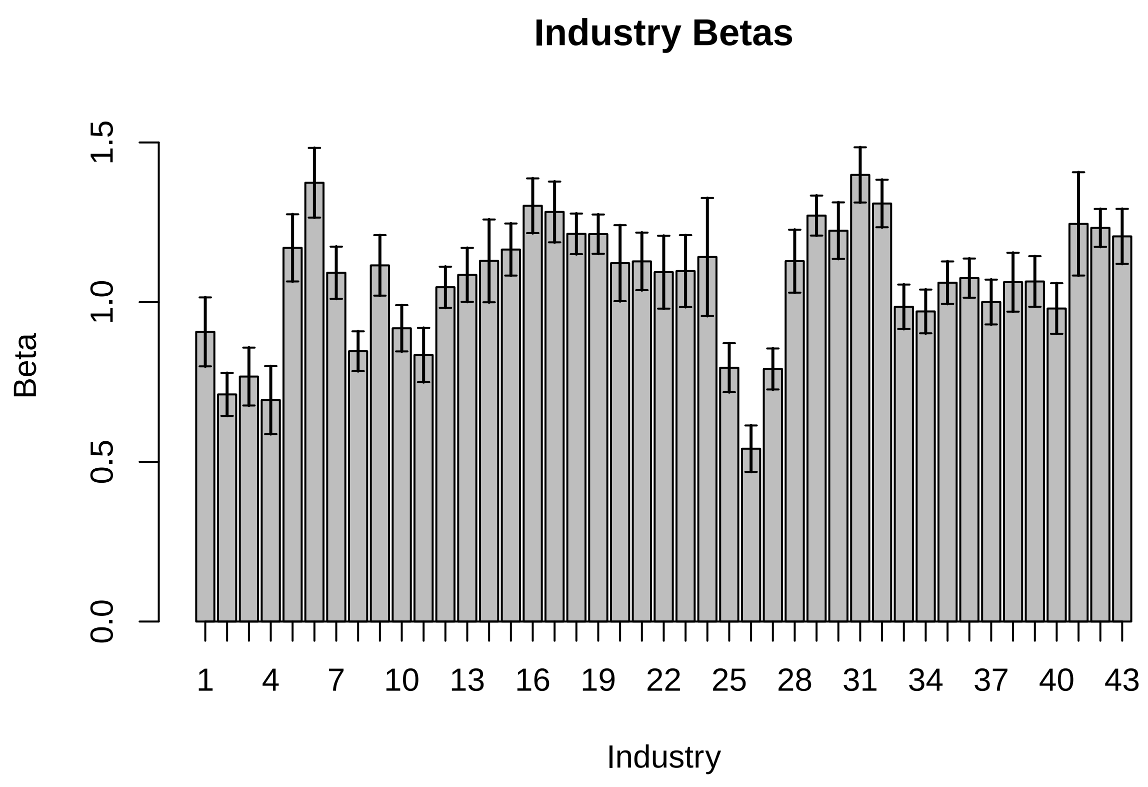
```
##### Problem 2: The Single Factor (Market) Model ------------------------
industry_48 <- read.csv("48_Industry_Portfolios_vw.csv", header = T) %>% as.data.table; gc()
```

```
##          used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells 2400137 128.2    4400554 235.1         NA  4400554 235.1
## Vcells 4709618  36.0   10146329  77.5      16384  8388608  64.0
```

```r
famafrench <- read.csv("F-F_Research_Data_Factors.csv", header = T) %>% as.data.table; gc()
```

```
##           used  (Mb) gc trigger  (Mb) limit (Mb) max used  (Mb)
## Ncells 2400144 128.2    4400554 235.1         NA  4400554 235.1
## Vcells 4716677  36.0   10146329  77.5      16384  8388608  64.0
```

```r
#industry_48[, Date := as.Date(as.character(X), "%Y%m")] -> Does not work

# Subset the data: year 1960~2015 & without -99.99 and missing values
industry_48$year <- as.numeric(substr(industry_48$X, 1, 4)) # get year info of each row
industry <- industry_48[ (industry_48$year >= 1960) & (industry_48$year < 2016), ] # year condition
industry[industry == -99.99] <- NA # changing -99.99 values to NA
industry[industry == -999] <- NA
industry_filtered <- industry %>% select_if( ~ !any(is.na(.)) ) # removing columns with NAs

# merge industry data and Rf from Fama-French data
data2 <- merge(industry_filtered, famafrench, by = "X")

### Q1. Regress the industry excess return on the market excess return
regression <- function(x) {
  out <- lm( (x-data2$RF) ~ data2$Mkt.RF )
  summary <- summary(out)

  alpha <- summary$coefficients[1, 1] # intercept for (c)
  beta <- summary$coefficients[2, 1] # beta for (a)                     #out$coef[2]
  hac <- sqrt(vcovHC(out, type = "HC")[2, 2]) # standard error for (a)  #print(hac)
  rsquared <- summary$r.squared # R^2 for (b)

  return(c(alpha, beta, hac, rsquared)) # does not return column names
}

industry_coef <- sapply(data2[, 2:44], regression) #data2[, 2:44]
row.names(industry_coef) <- c("Intercept", "Beta", "HAC", "R-Squared")

# Q1 (a). Bar plot
plots <- barplot(height = industry_coef["Beta", ], ylim = c(0, 1.6), xaxt = "n",
                 main = "Industry Betas", ylab = "Beta", xlab = "Industry", border = "black",
                 legend.text = F)
axis(1, at = plots, labels = 1:ncol(industry_coef))
segments(plots, industry_coef["Beta", ] - 2*industry_coef["HAC", ],
         plots, industry_coef["Beta", ] + 2*industry_coef["HAC", ], lwd = 1.5)
arrows(plots, industry_coef["Beta", ] - 2*industry_coef["HAC", ],
       plots, industry_coef["Beta", ] + 2*industry_coef["HAC", ], lwd = 1, angle = 90,
       code = 3, length = 0.03)
```



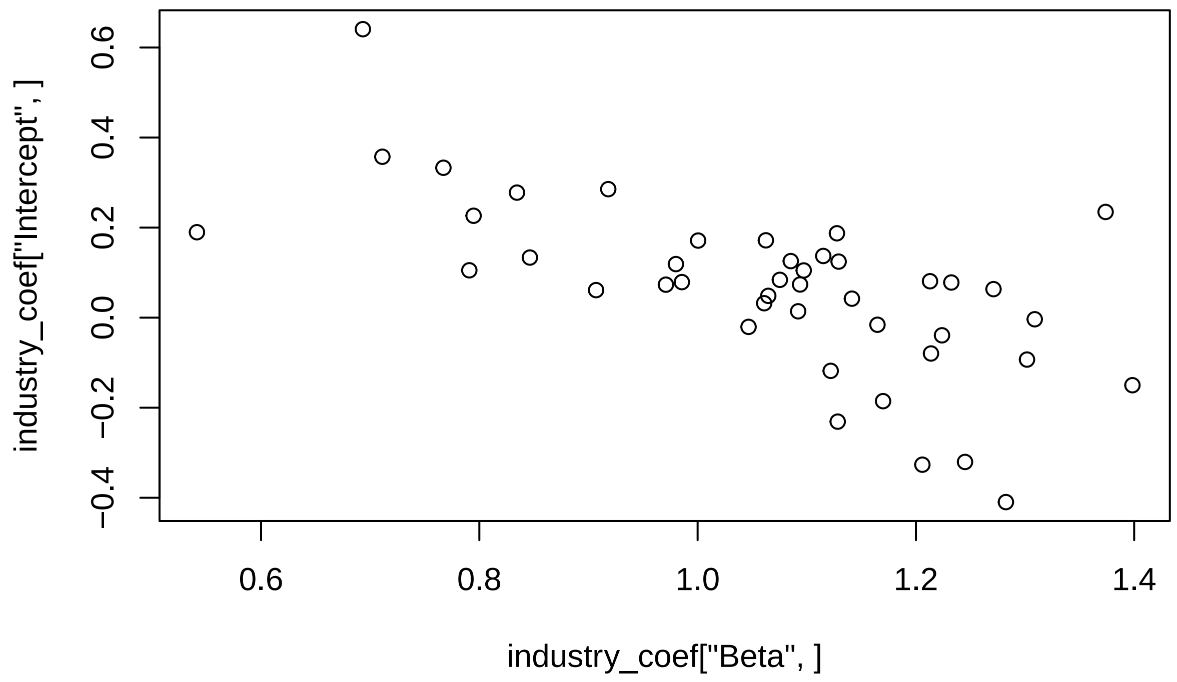
```r
# Q1 (b). Range of Beta and R-Squared
beta_range <- c(min = min(industry_coef["Beta", ]),
                mean = mean(industry_coef["Beta", ]),
                max = max(industry_coef["Beta", ]))
rsquared_range <- c(min = min(industry_coef["R-Squared", ]),
                    mean = mean(industry_coef["R-Squared", ]),
                    max = max(industry_coef["R-Squared", ]))
beta_range
```

```
##       min      mean       max
## 0.5412037 1.0648543 1.3984123
```

```r
rsquared_range
```

```
##       min      mean       max
## 0.2505185 0.5766409 0.8007488
```

```r
# Q1 (c). Plot Intercept vs. Beta
plot(industry_coef["Beta", ], industry_coef["Intercept", ])
```



```r
#abline(lm(industry_coef["Beta", ] ~ industry_coef["Intercept", ]))
```

The plot shows a down trending patern, which could also be a proof of the failure of CAPM.

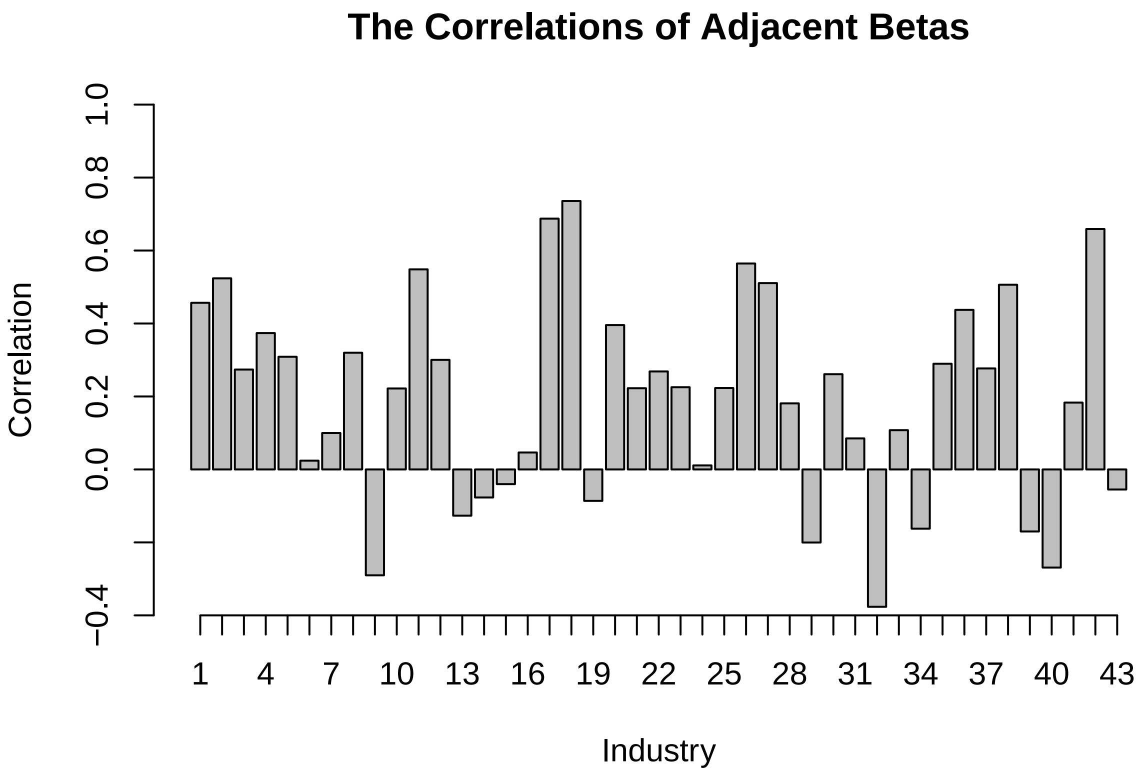
Alpha is the active return on an investment, gauges the performance of an investment against a market index or benchmark which is considered to represent the market's movement as a whole. An alpha of 1.0 means the mutual fund or investment outperformed its benchmark index by 1 percent. Conversely, an alpha of -1.0 means the mutual fund or investment underperformed its benchmark index by 1%. Beta is a measure of the volatility, or systematic risk, of a security or a portfolio in comparison to the market as a whole.

The patern tells that the higher the beta is, the lower the alpha is. However, there is not any causality between these two. When alpha is higher or lower than the benchmark, the volatility, which is the beta, could be high.

**2.**

```r
### Q2. Rolling regression of 5 years of data
# 11 betas for the each industry
years <- seq(from = 1960, to = 2015, by = 5)
industry_coef_5y <- data.frame()
for( i in 1:(length(years)-1) ) {
  # regression function to get the betas
  regression2 <- function(x) {
    out <- lm( (x-temp$RF) ~ temp$Mkt.RF )
    beta <- out$coef[2]
    return(beta)
  }

  temp <- data2[ (data2$year >= years[i] & data2$year < years[i+1]), ]
  industry_coef_5y <- rbind(industry_coef_5y, sapply(temp[, 2:44], regression2))
}
```

```r
# Result of betas
colnames(industry_coef_5y) <- colnames(industry_coef)
#industry_coef_5y

industry_corr <- data.frame()
# Calculate correlations of adjacent betas for each industry
correlation <- function(x) {
  len <- length(x) # 11
  return(cor( x = x[1:(len-1)], y = x[2:len] ))
}

# Result of correlations
industry_corr <- sapply(industry_coef_5y, correlation)
plots2 <- barplot(industry_corr, xaxt = "n", main = "The Correlations of Adjacent Betas",
                  ylim = c(-0.4, 1), ylab = "Correlation", xlab = "Industry")
axis(1, at = plots2, labels = 1:length(industry_corr))
```



According to the plot, most of the betas are highly correlated, and to be specific, most of them are positively correlated.

Reasons that betas are not the same across 5-year periods: The industry is embracing innovation or revolution, which shape the business style of this industry. Also, the risk profile of this industry changes over time, which could lead to the change of beta.