ABF\_Week1\_HW

**AI\_MBA 이태환**

**2023-12-27**

#install.packages("Quandl")  
#install.packages("highcharter")  
#install.packages("plotly")  
#install.packages("bigmemory")  
#install.packages("biganalytics")  
#install.packages("ff")  
#install.packages("biglm")  
library(Quandl)

library(tidyverse)

library(highcharter)

library(quantmod)

library(plotly)

library(bigmemory)

library(biganalytics)

library(ff)

library(biglm)

Quandl.api\_key("자신의 API 키 입력") # 변경 처리함

#Quandl 함수 설명

Quandl( code, type = c(“raw”, “ts”, “zoo”, “xts”, “timeSeries”), transform = c(““,”diff”, “rdiff”, “normalize”, “cumul”, “rdiff\_from”), collapse = c(““,”daily”, “weekly”, “monthly”, “quarterly”, “annual”), order = c(“desc”, “asc”), meta = FALSE, force\_irregular = FALSE, … )

해당 함수는 R 프로그래밍 언어에서 사용되는 함수로, 주로 금융 데이터를 가져오는 데 사용됩니다. 아래는 함수의 각 매개변수에 대한 설명입니다:

1. code: 데이터를 가져올 금융 기관 또는 데이터셋의 코드 또는 식별자를 지정하는 매개변수
2. type: 데이터의 출력 구조 유형을 지정하는 매개변수입니다.   
   “raw”: 원시 데이터  
   “ts”: 시계열 객체  
   “zoo”: Zoo 객체  
   “xts”: xts 객체  
   “timeSeries”: TimeSeries 객체.
3. transform: 데이터에 적용할 변환을 지정하는 매개변수  
   ““: 변환 없음  
   ”diff”: 차분  
   “rdiff”: 상대 차분  
   “normalize”: 정규화  
   “cumul”: 누적  
   “rdiff\_from”: 지정된 지점에서의 상대 차분
4. collapse: 데이터를 축소할 빈도를 지정하는 매개변수  
   ““: 축소하지 않음  
   ”daily”: 일일 데이터  
   “weekly”: 주간 데이터  
   “monthly”: 월간 데이터  
   “quarterly”: 분기별 데이터  
   “annual”: 연간 데이터.
5. order: 가져온 데이터의 정렬 순서를 지정하는 매개변수  
   “desc”: 내림차순  
   “asc”: 오름차순.
6. meta: 출력에 메타데이터를 포함할지 여부를 나타내는 논리형 매개변수
7. force\_irregular: 불규칙한 시계열을 강제로 생성할지 여부를 나타내는 논리형 매개변수
8. Slice and Dice the Data   
   ex) mydata = Quandl(“FRED/GDP”, start\_date=“2001-12-31”, end\_date=“2005-12-31”)
9. Request specific columns   
   ex) mydata = Quandl(c(“FRED/GDP.1”, “WIKI/AAPL.4”))

Qdata2 <- Quandl("NSE/OIL", collapse = "monthly")  
head(Qdata2)

## Date Open High Low Last Close Total Trade Quantity  
## 1 2019-01-31 172.05 174.95 172.05 174.55 174.55 431122  
## 2 2018-12-31 178.10 179.00 174.35 175.00 174.80 761462  
## 3 2018-11-30 190.00 195.80 185.35 186.75 186.25 6828327  
## 4 2018-10-31 196.15 198.60 194.00 196.95 197.60 426894  
## 5 2018-09-30 216.75 224.95 210.60 221.20 219.55 1139867  
## 6 2018-08-31 212.80 216.75 212.80 213.65 213.60 760196  
## Turnover (Lacs)  
## 1 749.99  
## 2 1343.75  
## 3 12935.68  
## 4 837.69  
## 5 2457.46  
## 6 1632.53

Qdata3 <- Quandl("FRED/GDP", start\_date = as.Date("2000-01-01"), end\_date = as.Date("2020-08-31"), type = "xts")  
head(Qdata3)

## [,1]  
## 2000 Q1 10002.18  
## 2000 Q2 10247.72  
## 2000 Q3 10318.17  
## 2000 Q4 10435.74  
## 2001 Q1 10470.23  
## 2001 Q2 10599.00

Qdata4 = Quandl("FRED/GDP", transform = "rdiff")  
head(Qdata4)

## Date Value  
## 1 2021-10-01 0.03404876  
## 2 2021-07-01 0.02028872  
## 3 2021-04-01 0.03188700  
## 4 2021-01-01 0.02610297  
## 5 2020-10-01 0.01603812  
## 6 2020-07-01 0.08528480

#Crude Oil Prices: West Texas Intermediate  
oil\_daily <- Quandl("FRED/DCOILWTICO", type = "raw", collapse = "daily", start\_date = "2008-01-01", end\_date = "2024-01-01")  
head(oil\_daily)

## Date Value  
## 1 2022-05-23 110.32  
## 2 2022-05-20 112.63  
## 3 2022-05-19 112.21  
## 4 2022-05-18 109.67  
## 5 2022-05-17 112.31  
## 6 2022-05-16 114.07

oil\_weekly <- Quandl("FRED/DCOILWTICO", type = "xts", collapse = "weekly", start\_date = "2008-01-01", end\_date = "2024-01-01")  
head(oil\_weekly)

## [,1]  
## 2008-01-06 97.90  
## 2008-01-13 92.74  
## 2008-01-20 90.55  
## 2008-01-27 90.37  
## 2008-02-03 89.03  
## 2008-02-10 91.77

oil\_monthly <- Quandl("FRED/DCOILWTICO", type = "xts", collapse = "monthly", start\_date = "2008-01-01", end\_date = "2024-01-01")  
head(oil\_monthly)

## [,1]  
## 1 2008 91.67  
## 2 2008 101.78  
## 3 2008 101.54  
## 4 2008 113.70  
## 5 2008 127.35  
## 6 2008 139.96

tail(index(oil\_monthly))

## [1] "12 2021" "1 2022" "2 2022" "3 2022" "4 2022" "5 2022"

mdy(last(index(oil\_monthly)))

## [1] "2022-05-20"

#mdy 함수는 날짜를 월-일-년(월/일/년) 형식으로 파싱하고 Date 객체로 반환하는 함수다.  
#index(oil\_monthly):시계열 데이터에서 시간 정보를 나타내는 인덱스를 추출합니다. 여기서는 날짜 또는 기간에 해당하는 값들이 추출된다.  
#last(...): 추출된 값 중에서 마지막 값을 선택합니다. 따라서 이 부분은 시계열 데이터의 마지막 시간 인덱스를 나타낸다.  
index(oil\_monthly) = seq(mdy("01/01/2008"), mdy(last(index(oil\_monthly))), by = "months")  
tail(index(oil\_monthly))

## [1] "2021-12-01" "2022-01-01" "2022-02-01" "2022-03-01" "2022-04-01"  
## [6] "2022-05-01"

head(index(oil\_monthly))

## [1] "2008-01-01" "2008-02-01" "2008-03-01" "2008-04-01" "2008-05-01"  
## [6] "2008-06-01"

hchart(oil\_monthly)

highchart(type = "stock") %>% hc\_add\_series(oil\_monthly, color = "cornflowerblue") %>%  
 hc\_yAxis(  
 title = list(text = "monthly price"),  
 labels = list(format = "${value}"), #y 축값에 $를 붙이라는 의미  
 opposite = FALSE #y title과 label이 왼쪽에 위치한다. //TRUE = 오른쪽에 위치  
 ) %>% hc\_add\_theme(hc\_theme\_flat())

Apple <- getSymbols("AAPL", from = "2020-01-01", to = "2023-12-31", auto.assign = F)  
SP500 <- getSymbols("^GSPC", from = "2020-01-01", to = "2023-12-31", auto.assign = F) #SP500 index  
Hanwha\_Life <- getSymbols("088350.KS", from = "2020-01-01", to = "2023-12-31", auto.assign = F)   
Samsung\_Life <- getSymbols("032830.KS", from = "2020-01-01", to = "2023-12-31", auto.assign = F)   
Samsung\_Elec <- getSymbols("005930.KS", from = "2020-01-01", to = "2023-12-31", auto.assign = F)   
#auto.assign = FALSE로 설정되어 있다면, 데이터를 받아온 후에는 현재 R 작업 환경에 자동으로 할당되지 않는다.  
#getSymbols("005930.KS", from = "2020-01-01", to = "2023-12-31", auto.assign = T) 로 할 경우, 데이터를 받아와서  
#005930.KS에 저장하게 된다.   
#getSymbols("005930.KS", from = "2020-01-01", to = "2023-12-31", auto.assign = T)

chart\_Series(Samsung\_Elec$`005930.KS.Close`) #tick mark: 숫자 1 옆. 특수문자로 시작할 경우 tick mark 붙여야 함  
**chart\_Series**(`005930.KS`**$**`005930.KS.Close`)

getSymbols("USD/KRW", src ="oanda", from = Sys.Date() - 180, to = Sys.Date())

## [1] "USD/KRW"

chart\_Series(USDKRW) #getSymbols의 auto.assign 기본값은 TRUE이므로, 위 명령을 실행하면 USDKRW 변수에 데이터를 저장한다.

#우리나라 실업율  
실업율 <- getSymbols("LRHUTTTTKRA156N", src = "FRED", from = "1998-01-01", to = "2021-12-31", auto.assign = F)  
chart\_Series(실업율)

SP500 <- getSymbols("^GSPC", from = "2010-01-01", to = "2023-12-31", auto.assign = F)  
plot(SP500[,4], col = "gray", type = "l", main = "S&P 500 Index")

rolling\_average = rollapply(SP500[,4], 100, mean) # 최근 100개의 평균을 구한다.  
lines(rolling\_average, col = "red", lwd = 2)

rolling\_volatility = rollapply(SP500[,4], 200, sd)  
plot(rolling\_volatility)

head(SP500)

## GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume GSPC.Adjusted  
## 2010-01-04 1116.56 1133.87 1116.56 1132.99 3991400000 1132.99  
## 2010-01-05 1132.66 1136.63 1129.66 1136.52 2491020000 1136.52  
## 2010-01-06 1135.71 1139.19 1133.95 1137.14 4972660000 1137.14  
## 2010-01-07 1136.27 1142.46 1131.32 1141.69 5270680000 1141.69  
## 2010-01-08 1140.52 1145.39 1136.22 1144.98 4389590000 1144.98  
## 2010-01-11 1145.96 1149.74 1142.02 1146.98 4255780000 1146.98

B <- new.env() # 새로운 환경이 생성되며, 이 환경은 변수, 함수, 그리고 다른 환경을 담을 수 있는 공간이 됩니다.  
getSymbols("BMW.DE", env = B, src = "yahoo", from = "2010-01-01", to = "2013-12-31")

## [1] "BMW.DE"

#2010년 1월 1일부터 2013년 12월 31일까지의 BMW 주식 데이터를 Yahoo Finance에서 가져와서 환경 B에 저장하는 작업을 수행합니다. 가져온 데이터는 B라는 환경 내에 "BMW.DE"라는 이름으로 변수로 저장됨  
BMW <- B$BMW.DE  
head(BMW)

## BMW.DE.Open BMW.DE.High BMW.DE.Low BMW.DE.Close BMW.DE.Volume  
## 2010-01-04 31.820 32.455 31.820 32.050 1808170  
## 2010-01-05 31.960 32.410 31.785 32.310 1564182  
## 2010-01-06 32.450 33.040 32.360 32.810 2218604  
## 2010-01-07 32.650 33.200 32.380 33.100 2026145  
## 2010-01-08 33.335 33.430 32.515 32.655 1925894  
## 2010-01-11 32.995 33.050 32.110 32.170 2157825  
## BMW.DE.Adjusted  
## 2010-01-04 18.19094  
## 2010-01-05 18.33851  
## 2010-01-06 18.62230  
## 2010-01-07 18.78690  
## 2010-01-08 18.53433  
## 2010-01-11 18.25905

chartSeries(BMW, multi.col = TRUE, theme = "white")

addBBands() #add the Bollinger Bands

addMACD() #the MACD trend-follwing momentum indicator

B.Return <- log(BMW$BMW.DE.Close / BMW$BMW.DE.Open)  
qqnorm(B.Return,  
 main = "Normal Probability Plot of BMW daily Log Returns",  
 xlab = "Theoretical Quantiles",  
 ylab = "Sample Quantiles")  
qqline(B.Return, col = "red", lwd = 2.5)

#Seasonal Decomposition of Times Series by Loess  
# stl 함수는 R의 시계열 데이터에 대해 STL (Seasonal-Trend decomposition using LOESS) 분해를 수행하는 함수입니다. STL은 시계열 데이터를 계절성, 추세, 그리고 나머지(residuals)로 분해하여 각 구성 요소를 개별적으로 추정합니다.  
#   
# 여기서 stl 함수의 몇 가지 주요 매개변수에 대한 설명  
#   
# - nottem: STL 분해를 수행할 시계열 데이터입니다. 여기서는 nottem이라는 변수가 해당 데이터를 나타내는 것으로 가정합니다.  
#   
# - s.window: 계절성 구성 요소를 추정할 때 사용되는 LOESS(smoothed time-series) 창의 크기입니다. "periodic"으로 설정하면 자동으로 계절성을 결정  
#   
# - t.window: 추세(trend) 구성 요소를 추정할 때 사용되는 LOESS 창의 크기  
#   
# - t.jump: 추세(trend)의 점프(jump) 또는 간격을 결정합니다. 이 매개변수는 추세를 추정할 때 사용되는 창을 얼마나 건너뛸지를 결정  
#   
# 이러한 설정을 사용하여 stl(nottem, s.window = "periodic", t.window = 70, t.jump = 1)는 nottem 시계열 데이터에 대한 STL 분해를 수행합니다. 계절성은 자동으로 결정되고, 추세는 LOESS 창 크기가 70이고 간격이 1인 설정으로 추정한다  
  
#t.jump 값을 크게 하면 trend 곡선이 변한다.  
plot(stl(nottem, s.window = "periodic", t.window = 70, t.jump = 1), main = "Average Monthly Temperatures at Nottingham, 1920-1939")

#Candl Stick Chart 그려보기  
df <- data.frame(Date = index(Samsung\_Elec), coredata(Samsung\_Elec))  
df <- tail(df, 30)  
  
fig <- df %>%  
 plot\_ly(  
 x = ~Date,  
 type = "candlestick",  
 open = ~X005930.KS.Open,  
 close = ~X005930.KS.Close,  
 high = ~X005930.KS.High,  
 low = ~X005930.KS.Low  
 )  
  
# 하단 range slider on/off설정 F:off, T:on  
fig <-  
 fig %>%  
 layout(title = "Basic Candlestick Chart(Samsung Electronics", xaxis = list(rangeslider = list(visible = F)))  
  
fig

# read.big.matrix는 R 프로그래밍 언어의 bigmemory 패키지에 속한 함수 중 하나입니다. bigmemory 패키지는 대용량 메모리 매핑을 통해 대규모 데이터를 효율적으로 다루는 데 사용되는 도구를 제공합니다.  
# read.big.matrix 함수는 대용량의 행렬 데이터를 효율적으로 메모리에 읽어오는 데 사용됩니다. 특히, 메모리에 맞지 않는 큰 데이터셋을 작은 조각으로 나누어 읽어들일 때 유용합니다.  
x <- read.big.matrix("bmw.csv", header = TRUE, type = "double")  
head(x)

## Index BMW.DE.Open BMW.DE.High BMW.DE.Low BMW.DE.Close BMW.DE.Volume  
## [1,] 1 31.820 32.455 31.820 32.050 1808170  
## [2,] 2 31.960 32.410 31.785 32.310 1564182  
## [3,] 3 32.450 33.040 32.360 32.810 2218604  
## [4,] 4 32.650 33.200 32.380 33.100 2026145  
## [5,] 5 33.335 33.430 32.515 32.655 1925894  
## [6,] 6 32.995 33.050 32.110 32.170 2157825  
## BMW.DE.Adjusted  
## [1,] 21.72463  
## [2,] 21.90087  
## [3,] 22.23979  
## [4,] 22.43635  
## [5,] 22.13472  
## [6,] 21.80597

x <- x[,-1]  
head(x)

## BMW.DE.Open BMW.DE.High BMW.DE.Low BMW.DE.Close BMW.DE.Volume  
## [1,] 31.820 32.455 31.820 32.050 1808170  
## [2,] 31.960 32.410 31.785 32.310 1564182  
## [3,] 32.450 33.040 32.360 32.810 2218604  
## [4,] 32.650 33.200 32.380 33.100 2026145  
## [5,] 33.335 33.430 32.515 32.655 1925894  
## [6,] 32.995 33.050 32.110 32.170 2157825  
## BMW.DE.Adjusted  
## [1,] 21.72463  
## [2,] 21.90087  
## [3,] 22.23979  
## [4,] 22.43635  
## [5,] 22.13472  
## [6,] 21.80597

class(x)

## [1] "matrix" "array"

# lapply 함수는 R에서 리스트에 적용되는 함수로, 주어진 리스트의 각 요소에 함수를 적용하여 리스트를 반환합니다. 여기서 1:10은 1부터 10까지의 정수로 이루어진 리스트를 나타냅니다. 따라서 lapply(1:10, function(i){bigkmeans(x, center = i, iter.max = 100, nstart = 10)})은 1부터 10까지의 정수를 가지고 있는 리스트에 대해 bigkmeans 함수를 각각 적용하는 것을 의미합니다.  
  
# 이 코드에서 사용된 함수와 매개변수들에 대한 설명은 다음과 같습니다:  
# 1:10: 1부터 10까지의 정수로 이루어진 리스트를 생성합니다.  
# function(i){...}: 리스트의 각 요소에 대해 실행될 함수를 정의합니다. 함수의 인자 i는 리스트의 각 요소를 나타냅니다.  
# bigkmeans(x, center = i, iter.max = 100, nstart = 10): bigkmeans는 대용량 데이터셋에 대해 k-평균 군집화를 수행하는 함수입니다.  
# x: 군집화를 수행할 데이터셋.  
# center = i: 클러스터의 개수를 i로 설정합니다. i는 1부터 10까지의 정수로 변화하면서 군집화가 수행됩니다.  
# iter.max = 100: 반복 횟수의 상한값을 설정합니다.  
# nstart = 10: 무작위 초기화를 10번 시도하며 가장 좋은 결과를 선택합니다.  
# 따라서 lapply(1:10, function(i){bigkmeans(x, center = i, iter.max = 100, nstart = 10)})은 군집화를 클러스터 수가 1부터 10까지 변화하면서 각각 수행하고, 그 결과를 리스트로 반환합니다. 반환된 리스트의 각 요소는 각 클러스터 수에 대한 군집화 결과입니다.  
Results <- lapply(1:10, function(i){bigkmeans(x, center = i, iter.max = 100, nstart = 10)})  
Within <- lapply(Results, function(x) sum(x$withinss)/(10^15))   
#10^15 로 나눈 이유는 Within cluster sum of squares by cluster 값이 {값}\*e^15로 표기 되기 때문에 e^15를 없애기 위함이다.   
plot(1:10, Within, type = "b", xlab = "Number of Clusters", ylab = "Within Sum of Square")

R3 <- bigkmeans(x, center = 3, iter.max = 100, nstart = 10)  
summary(R3)

## Length Class Mode   
## cluster 1017 -none- numeric  
## centers 18 -none- numeric  
## withinss 3 -none- numeric  
## size 3 -none- numeric

R3$cluster

## [1] 2 2 2 2 2 2 3 3 3 3 2 3 3 3 3 2 3 3 1 1 3 3 3 3 3 3 2 2 3 3 2 3 2 1 3 2 3  
## [38] 2 2 2 2 3 2 2 3 2 3 3 1 2 2 2 3 3 1 3 1 3 3 3 2 2 3 3 3 2 3 2 2 3 3 3 1 2  
## [75] 1 3 3 3 2 3 3 3 2 2 3 3 1 1 1 1 1 3 1 3 3 1 1 1 3 1 3 3 2 2 3 2 3 3 3 3 2  
## [112] 1 3 3 2 3 3 1 3 2 2 3 3 3 2 3 3 3 2 2 3 3 3 2 1 1 1 3 2 3 2 3 2 2 3 3 2 3  
## [149] 2 1 3 3 3 2 2 2 3 2 2 3 2 3 3 2 2 3 2 2 2 3 3 2 3 2 2 3 3 1 3 3 3 3 1 1 3  
## [186] 3 3 3 3 3 3 3 1 3 3 3 3 3 2 3 3 3 3 2 3 2 3 2 2 3 3 3 3 3 3 1 3 3 2 3 3 2  
## [223] 3 2 2 2 2 3 3 3 3 2 3 3 3 3 1 3 3 3 3 1 3 2 2 2 2 3 2 2 2 2 3 3 2 2 2 3 3  
## [260] 2 3 2 3 2 3 1 2 3 1 1 1 3 3 3 3 3 3 3 3 3 2 3 1 3 3 3 3 3 3 3 3 2 2 3 3 3  
## [297] 2 3 3 3 3 2 2 3 1 3 3 1 3 3 1 3 3 3 3 2 2 3 3 3 3 2 3 3 1 3 3 3 3 2 3 3 3  
## [334] 3 2 2 3 3 2 3 3 1 3 3 3 3 1 1 3 2 3 2 3 1 3 2 2 3 2 2 2 2 2 2 2 2 3 1 3 2  
## [371] 3 3 3 1 2 3 3 3 1 2 3 3 3 3 2 2 3 3 3 3 1 3 3 3 3 2 2 3 2 3 3 3 3 2 3 1 1  
## [408] 1 1 1 1 1 1 1 3 3 3 1 1 1 1 1 1 3 3 3 1 1 3 1 1 3 3 3 1 3 1 1 1 3 3 3 1 1  
## [445] 1 1 3 3 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 1 3 1 3 3 3 3 3 2 2 2 3  
## [482] 3 3 3 3 3 3 3 3 3 1 3 3 3 2 2 3 3 3 3 1 3 1 3 3 2 2 2 2 2 2 2 2 3 2 3 2 3  
## [519] 3 2 2 3 2 3 3 3 3 3 2 2 3 3 2 3 3 3 3 2 3 3 1 3 2 2 3 3 3 2 2 3 3 2 3 2 2  
## [556] 3 2 3 3 3 1 3 3 3 1 3 1 3 1 3 3 3 2 2 2 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2  
## [593] 3 3 3 2 3 1 3 3 3 3 1 3 2 3 1 3 1 3 3 3 3 2 2 3 3 3 1 3 3 3 3 2 3 3 3 3 1  
## [630] 3 3 2 3 2 3 3 2 2 3 2 3 2 3 3 2 2 2 3 3 2 2 2 3 3 3 3 3 3 3 3 3 1 1 3 3 3  
## [667] 2 2 2 2 2 2 2 3 2 2 2 3 2 2 2 2 3 3 2 3 3 1 3 2 2 3 3 3 2 3 2 1 1 2 3 3 3  
## [704] 2 3 2 3 3 2 2 2 3 2 2 2 3 3 1 2 3 3 2 2 2 2 3 3 3 3 3 1 3 3 2 2 2 2 3 3 2  
## [741] 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 2 2 2 3 3 2 2 2 2 2 2  
## [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 2 2 2 2 2 2 2 2 2  
## [815] 2 2 3 3 2 3 2 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 3 2 2 2 2 3 2 2  
## [852] 2 2 2 3 2 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 3 3 2  
## [889] 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2  
## [926] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2  
## [963] 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 1 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [1000] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2

R3$centers

## [,1] [,2] [,3] [,4] [,5] [,6]  
## [1,] 53.48832 54.67921 52.23121 53.52855 5243880 37.28641  
## [2,] 64.55363 65.11840 63.91027 64.58411 1623721 46.66433  
## [3,] 56.04169 56.87082 55.18087 56.04623 2871428 39.16213

colnames(x)

## [1] "BMW.DE.Open" "BMW.DE.High" "BMW.DE.Low" "BMW.DE.Close"   
## [5] "BMW.DE.Volume" "BMW.DE.Adjusted"

# read.csv.ffdf는 R 언어의 ff 패키지에서 제공되는 함수로, 대용량 CSV 파일을 메모리에 읽어들이기 위한 함수입니다. ff 패키지는 데이터를 파일로부터 읽거나 파일로 쓰는 등의 작업을 수행할 때 메모리를 효율적으로 활용할 수 있도록 도와주는 패키지입니다.  
# 여기서 read.csv.ffdf 함수는 ff 패키지에서 제공되는 대용량 데이터를 처리하기 위한 클래스인 ffdf 클래스의 객체로 데이터를 읽어들입니다.  
download.file("http://www.irs.gov/file\_source/pub/irs-soi/12zpallagi.csv", "soi.csv")  
x <- read.csv.ffdf(file="soi.csv", header = T)  
class(x)

## [1] "ffdf"

BigModel <- biglm(A02300~A00200+AGI\_STUB+NUMDEP+MARS2, data = x)  
summary(BigModel) #변수A02300: 국가에서 지불하는 실업수당당

## Large data regression model: biglm(A02300 ~ A00200 + AGI\_STUB + NUMDEP + MARS2, data = x)  
## Sample size = 166904   
## Coef (95% CI) SE p  
## (Intercept) 131.9412 44.3847 219.4977 43.7782 0.0026  
## A00200 -0.0019 -0.0019 -0.0018 0.0000 0.0000  
## AGI\_STUB -40.1597 -62.6401 -17.6794 11.2402 0.0004  
## NUMDEP 0.9270 0.9235 0.9306 0.0018 0.0000  
## MARS2 -0.1451 -0.1574 -0.1327 0.0062 0.0000

summary(BigModel)$rsq #대용량 데이터에 대한 선형 회귀 모델(biglm 객체)의 결정 계수 (R-squared) 값

## [1] 0.8609021