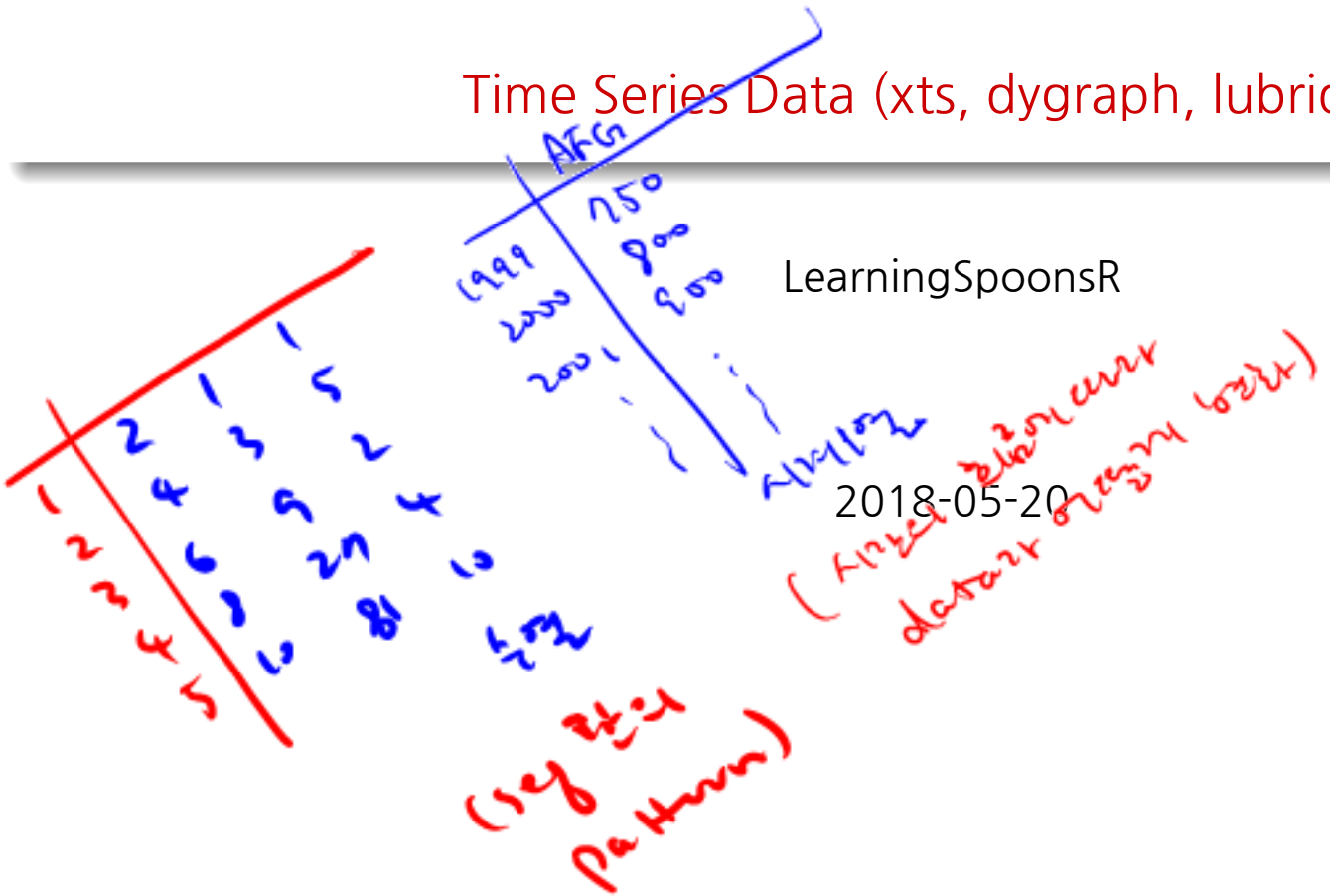
Time Series Data (xts, dygraph, lubridate)



LearningSpoonsR

2018-05-20

시계열 데이터

1. Time Series Data

   - 시간에 따라서 값이 변화하는 데이터를 시계열 데이터라고 함.

2. `xts`

   - `data.frame`의 확장 구조로써 시간 정보를 포함하는 데이터 구조 (Data Structure).
   - `data.frame`의 많은 명령어를 그대로 사용할 수 있음.
   - `data.frame`의 `rownames`에 해당하는 특성을 `index`라고 하는데,
   - `index`에 시간에 대한 정보가 포함되어 있음.
   - 즉, `index`는 `Date` 객체등 시간을 나타내는 데이터 타입.

3. `dygraph`

- R과 몇몇 소프트웨어에서 사용가능한 패키지
- `xts`개체를 쉽게 interactive한 plot으로 그릴 수 있음.
- plot에 `highlight`, `annotation`, `label` 등의 기능을 쉽게 추가할 수 있음.
- `rmarkdown`에서 `html`로 렌더하는 경우에 특히 유용함
- `ggplot2`보다 시계열 데이터 plotting에 대해서는 현재까지는 더 효과적
- `shiny`에서 사용할 수 있는 `renderDygraph` 함수를 제공
- https://rstudio.github.io/dygraphs/

4. `lubridate`

- `Date`객체등 시간을 나타내는 데이터 타입을 효과적으로 다룰 수 있는 패키지

Mini-Project: MSFT 주가 분석

MSFT 주가 불러오기

- Quandl은 글로벌 금융 데이터 베이스 제공업체입니다.
- Quandl은 **Quandl:Bloomberg = 위키피디아가:브리터니커**를 지향합니다.

```
source("LSR.R")
activate("Quandl")
Quandl.api_key("SD27xu59qZmj-YCnxwDm")
MSFT <- Quandl("WIKI/MSFT")
str(MSFT)
```

```
## 'data.frame':    8076 obs. of  13 variables:
##  $ Date       : Date, format: "2018-03-27" "2018-03-26" ...
##  $ Open       : num  94.9 90.6 89.5 91.3 92.9 ...
##  $ High       : num  95.1 94 90.5 91.8 94 ...
##  $ Low        : num  88.5 90.4 87.1 89.7 92.2 ...
##  $ Close      : num  89.5 93.8 87.2 89.8 92.5 ...
##  $ Volume     : num  53704562 55031149 42159397 37578166 23753263 ...
##  $ Ex-Dividend: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Split Ratio: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ Adj. Open  : num  94.9 90.6 89.5 91.3 92.9 ...
##  $ Adj. High  : num  95.1 94 90.5 91.8 94 ...
##  $ Adj. Low   : num  88.5 90.4 87.1 89.7 92.2 ...
##  $ Adj. Close : num  89.5 93.8 87.2 89.8 92.5 ...
##  $ Adj. Volume: num  53704562 55031149 42159397 37578166 23753263 ...
##  - attr(*, "freq")= chr "daily"
```

```
class(MSFT)
```

```
## [1] "data.frame"
```

- 설명의 편의성을 위해 MSFT객체에서 날짜, 거래량, 일일 종가만을 선택하여 진행합니다.

```
MSFT <- MSFT[,c(1,6,12)]
head(MSFT)
```

```
##         Date   Volume Adj. Close
## 1 2018-03-27 53704562      89.47
## 2 2018-03-26 55031149      93.78
## 3 2018-03-23 42159397      87.18
## 4 2018-03-22 37578166      89.79
## 5 2018-03-21 23753263      92.48
## 6 2018-03-20 21787780      93.13
```

```
class(MSFT)
```

```
## [1] "data.frame"
```

```
dim(MSFT)
```

```
## [1] 8076    3
```

- data.frame객체인 MSFT의 첫번째 컬럼이 시간 정보를 포함하고 있습니다.
- MSFT는 현재 data.frame객체입니다.

data.frame객체 MSFT를 xts객체로 변환

- 아래의 명령어와 같이 `xts()`함수를 이용해서 `data.frame`을 `xts`로 변환합니다.
- 시간의 정보가 해당하는 컬럼을 제외한 부분을 `MSFT[,-1]`와 같이 첫 번째 인수로 넣어주고…
- 시간의 정보에 해당하는 컬럼을 `order.by=` 를 이용해서 넣어줍니다.

```
activate("xts")
MSFT_xts <- xts(MSFT[,-1], order.by = as.Date(MSFT[,1]))
head(MSFT_xts)
```

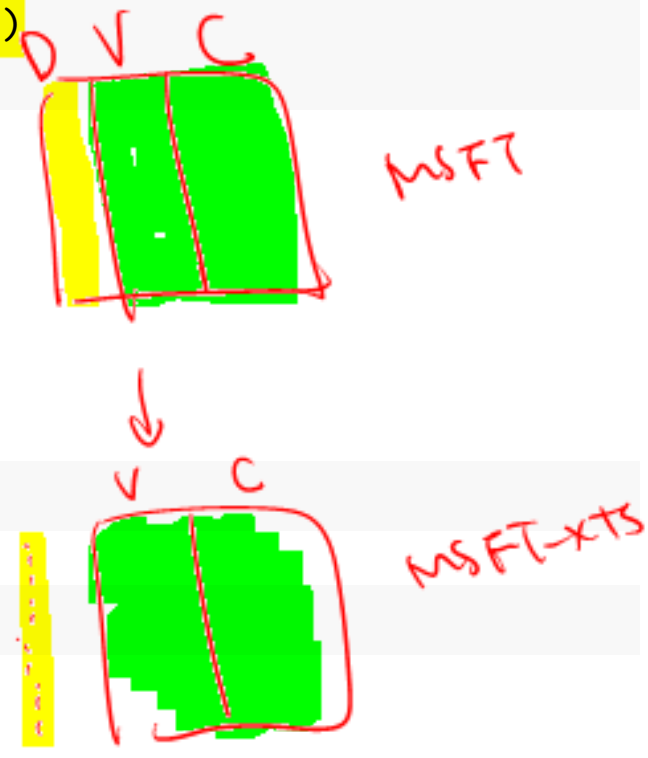
```
##            Volume Adj. Close
## 1986-03-13 3582600 0.08471998
## 1986-03-14 1070000 0.08703141
## 1986-03-17  462400 0.08818712
## 1986-03-18  235300 0.08645355
## 1986-03-19  166300 0.08529784
## 1986-03-20  202900 0.08356427
```

```
class(MSFT_xts)
```

```
## [1] "xts" "zoo"
```

```
dim(MSFT_xts)
```

```
## [1] 8076    2
```



- `xts`로 변환하면서 `dimension`이 줄었습니다. 그런데 모든 정보를 포함하고 있습니다?!
- `head()`의 결과를 보면 `xts`에서는 시간 정보를 마치 `data.frame`의 `rownames()`와 같은 방법으로 포함하고 있습니다.
- 시간의 정보를 이와 같이 색인, 즉, `index`로 만들기 때문에 정보의 손실없이 객체를 변환할 수 있습니다.
- R의 `data.frame`은 Python의 `Pandas` 패키지의 motivation이 되었다고 합니다.
- Python `Pandas`에서는 `Pandas.DataFrame`과 함께 `Pandas.Series`라는 `xts`에 대응되는 데이터 구조를 제공합니다.
- R과 Python에서 시계열 객체에 대한 문법의 구조도 거의 동일합니다.

xts객체를 data.frame로 변환

- (저는 한번도 필요를 느끼지 못했지만)
- xts객체를 data.frame으로 바꾸는 것도 당연히 가능합니다.

```
MSFT_df <- data.frame(index(MSFT_xts), MSFT_xts)
head(MSFT_df)
```

```
##             index.MSFT_xts.  Volume Adj..Close
## 1986-03-13      1986-03-13 3582600 0.06471998
## 1986-03-14      1986-03-14 1070000 0.06703141
## 1986-03-17      1986-03-17  462400 0.06818712
## 1986-03-18      1986-03-18  235300 0.06645355
## 1986-03-19      1986-03-19  166300 0.06529784
## 1986-03-20      1986-03-20  202900 0.06356427
```

```
class(MSFT_df)
```

```
## [1] "data.frame"
```

- data.frame에서 rownames가 거슬린다면 없애주면 됩니다.

```
rownames(MSFT_df) <- NULL
head(MSFT_df)
```

```
##   index.MSFT_xts.  Volume Adj..Close
## 1      1986-03-13 3582600 0.06471998
## 2      1986-03-14 1070000 0.06703141
## 3      1986-03-17  462400 0.06818712
## 4      1986-03-18  235300 0.06645355
## 5      1986-03-19  166300 0.06529784
## 6      1986-03-20  202900 0.06356427
```

dygraph로 그리기

- `xts`개체는 시간정보가 내재되어 있어서 다루기 편하고,
- `html`문서를 만들때에 `dygraph`를 이용할 수 있습니다.
- `dygraph`의 문법은 `ggplot`보다 더 쉽습니다.
- https://rstudio.github.io/dygraphs/

```
activate("dygraphs")
dygraph(MSFT_xts[,2]) %>% dyRangeSelector()
```

lubridate로 시계열 데이터 다루기

```
activate("lubridate")
theDay <- as.Date("2018-03-26")
```

- theDay의 한달 전은 언제입니까? “03”에서 1을 빼서 “02”로 바꾸면 되겠네요?

```
a <- as.numeric(substr(theDay, 6, 7))-1 # subtract month
activate("stringr")
lastMonthDay <- as.Date(paste(
  substr(theDay, 1, 4),
  str_pad(a, 2, pad = "0"), # fill with leading zero
  substr(theDay, 9, 10),
  sep = "-"))
lastMonthDay
```

```
## [1] "2018-02-26"
```

- “2018-01-15”의 한달 전은 언제입니까? 위의 코드로 해결이 안됩니다.
- “2018-03-31”의 한달 전은 언제입니까? 위의 코드로 해결이 안됩니다.
- lubridate의 months를 이용하면 아래의 연산이 가능합니다!!

```
theDay - months(1)
```

```
## [1] "2018-02-26"
```

- 이번달 1일

```
floor_date(theDay, "month")
```

```
## [1] "2018-03-01"
```

- 저번달 말일

```
floor_date(theDay, "month")-days(1)
```

```
## [1] "2018-02-28"
```

- 전년 말일

```
floor_date(theDay, "years")-1
```

```
## [1] "2017-12-31"
```

- 전년 동월 말일

```
ceiling_date(theDay-years(1))
```

```
## [1] "2017-03-26 00:00:01 UTC"
```

- 2018-03-26의 MSFT주가는 아래와 같이 확인이 가능합니다.

```
MSFT_xts[theDay,]
```

```
##              Volume Adj. Close
## 2018-03-26 55031149      93.78
```

```
MSFT_xts[index(MSFT_xts)==theDay,]
```

```
##              Volume Adj. Close
## 2018-03-26 55031149      93.78
```

- theDay의 과거 1개월전 주가가 궁금하다면…

```
MSFT_xts[theDay-months(1),]
```

```
##                Volume Adj. Close
## 2018-02-26 29760276       95.42
```

```
MSFT_xts[index(MSFT_xts)==(theDay-months(1)),]
```

```
##                Volume Adj. Close
## 2018-02-26 29760276       95.42
```

- theDay의 과거 1년전 주가가 궁금하다면…

```
MSFT_xts[theDay-years(1),]
```

```
##       Volume Adj. Close
```

```
MSFT_xts[index(MSFT_xts)==(theDay-years(1)),]
```

```
##       Volume Adj. Close
```

- weekdays(theDay-years(1))는 일요일입니다. 그래서 기록이 없습니다.
- 많은 시계열 데이터가 이와 같이 관찰값이 없는 시간이 많습니다.
- 처음에 시계열 데이터를 다루기가 이유는 이 때문이고, 코드가 **더러워** 지는 결과를 흔히 초래합니다.
- 그렇기 때문에 흔히 tidyverse (dplyr의 상위 패키지)의 fill과 같은 명령을 사용해서 데이터셋을 365일로 강제로 바꾸기도 합니다.
- 그런데 이 경우에는 관찰값이 있었던 날과 아닌 날이 구분이 안되므로 또 다른 문제를 야기하기도 합니다.
- 그렇다면 이런 경우에는 어떻게 해야할까요?

```
MSFT_xts[theDay-years(1),]
MSFT_xts[index(MSFT_xts)==(theDay-years(1)),]
```

- available은 theDay-years(1)시점에서 존재했을 기록들의 행번호 입니다.
- 그중에 가장 나중의 기록을 사용하면 됩니다.

```
available <- which(index(MSFT_xts)<=(theDay-years(1)))
MSFT_xts[max(available),]
```

```
##                Volume Adj. Close
## 2017-03-24 22617105   63.95089
```

- 위의 방식을 이용하면 관찰값이 있던 없던 해당 시점에 사용가능한 가장 마지막 관찰값을 불러올 수 있습니다.
- 이런 식으로 시계열 데이터를 다루는 방법은 모든 컴퓨터 언어와 엑셀에도 적용할 수 있습니다.
- 간단히 소개해드리는 KOSPI200 프로그램은 이와 같은 접근법을 사용합니다.