



# 03. 데이터 마이닝과 dplyr 패키지

3주차

성현곤



충북대학교 도시공학과  
Dept. of URBAN ENGINEERING

# 목차

- 데이터 불러오기와 저장하기
  - 데이터의 종류와 불러오기(저장하기)
  - 내장 데이터 불러오기
  - 작업경로(폴더) 확인 및 변경
  - 외장 데이터 불러오기와 저장하기
    - 텍스트 파일(.txt, .csv, ...)
    - 엑셀 스프레드 시트(.xls, .xlsx)
- 데이터 마이닝 기초
  - 데이터 탐구하기
  - 데이터 정렬하기
  - 데이터 병합하기
  - 데이터 변환하기
  - 데이터 추출하기
  - 기타 유용한 함수들: (...) with, attach, detach
- dplyr 패키지를 활용한 데이터 마이닝
  - dplyr 패키지 소개
  - 추출하기: select()
  - 특정 조건에 맞는 데이터 추출하기: filter()
  - 정렬하기: arrange()
  - 변수명 변환하기: rename()
  - 데이터 수정 및 변형: mutate(), transmute()
  - 집단별 데이터 분할하기: group\_by()
  - 데이터 마이닝 간결하게 수행하기: %>% (파이프 연산자)
  - 데이터 요약하기: summarise()
  - 응용 실습
  - tbl\_df() 소개

# 데이터 불러오기와 저장하기

## 데이터의 종류와 함수

- R은 거의 모든 자료 활용 가능
  - 내장 데이터
    - 기본 데이터세트
  - 외장 데이터
    - 일반 텍스트 파일
    - 엑셀 스프레드 시트
    - 기타 각종 프로그램 데이터 파일
    - 데이터 베이스
    - 웹페이지
    - 비정형 텍스트 파일 등
- 기본 데이터 세트(내장 데이터)
  - R 데이터 세트
    - 리스트와 간단한 설명 확인
    - `help(package= " datasets ")`
  - 패키지와 함께 있는 파일

```
> ### 내장 데이터 불러오기(적재하기)
> library(help=datasets) # 내장 데이터 목록보기
```

패키지 'datasets'에 대한 정보

설명:

```
Package: datasets
Version: 3.5.3
Priority: base
Title: The R Datasets Package
Author: R Core Team and contributors worldwide
Maintainer: R Core Team <R-core@r-project.org>
Description: Base R datasets.
License: Part of R 3.5.3
Built: R 3.5.3; ; 2019-03-11 08:33:40 UTC; windows
```

인덱스:

AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJsales	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide Uptake in Grass Plants
ChickWeight	Weight versus age of chicks on different diets
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Students
Harman23.cor	Harman Example 2.3
Harman74.cor	Harman Example 7.4
Indometh	Pharmacokinetics of Indomethacin
InsectSprays	Effectiveness of Insect Sprays
JohnsonJohnson	Quarterly Earnings per Johnson & Johnson Share
LakeHuron	Level of Lake Huron 1875-1972
LifeCycleSavings	Intercountry Life-Cycle Savings Data
Loblolly	Growth of Loblolly pine trees
Nile	Flow of the River Nile
Orange	Growth of Orange Trees

# 데이터 불러오기와 저장하기

## 데이터의 종류와 함수

- 외장 데이터 불러오기와 함수
  - 외장 데이터 종류와 함수

Format	Typical Extension	Import Package	Export Package	Installed by Default					
					"XBASE" database files	.dbf	<a href="#">foreign</a>	<a href="#">foreign</a>	Yes
Comma-separated data	.csv	<a href="#">data.table</a>	<a href="#">data.table</a>	Yes	Weka Attribute-Relation File Format	.arff	<a href="#">foreign</a>	<a href="#">foreign</a>	Yes
Pipe-separated data	.psv	<a href="#">data.table</a>	<a href="#">data.table</a>	Yes	Data Interchange Format	.dif	<a href="#">utils</a>		Yes
Tab-separated data	.tsv	<a href="#">data.table</a>	<a href="#">data.table</a>	Yes	Fortran data	no recognized extension	<a href="#">utils</a>		Yes
SAS	.sas7bdat	<a href="#">haven</a>	<a href="#">haven</a>	Yes	Fixed-width format data	.fwf	<a href="#">utils</a>	<a href="#">utils</a>	Yes
SPSS	.sav	<a href="#">haven</a>	<a href="#">haven</a>	Yes	gzip comma-separated data	.csv.gz	<a href="#">utils</a>	<a href="#">utils</a>	Yes
Stata	.dta	<a href="#">haven</a>	<a href="#">haven</a>	Yes	CSVY (CSV + YAML metadata header)	.csvy	<a href="#">csvy</a>	<a href="#">csvy</a>	No
SAS XPORT	.xpt	<a href="#">haven</a>	<a href="#">haven</a>	Yes	EViews	.wf1	<a href="#">hexView</a>		No
SPSS Portable	.por	<a href="#">haven</a>		Yes	Feather R/Python interchange format	.feather	<a href="#">feather</a>	<a href="#">feather</a>	No
Excel	.xls	<a href="#">readxl</a>		Yes	Fast Storage	.fst	<a href="#">fst</a>	<a href="#">fst</a>	No
Excel	.xlsx	<a href="#">readxl</a>	<a href="#">openxlsx</a>	Yes	JSON	.json	<a href="#">jsonlite</a>	<a href="#">jsonlite</a>	No
R syntax	.R	<a href="#">base</a>	<a href="#">base</a>	Yes	Matlab	.mat	<a href="#">rmatio</a>	<a href="#">rmatio</a>	No
Saved R objects	.RData, .rda	<a href="#">base</a>	<a href="#">base</a>	Yes	OpenDocument Spreadsheet	.ods	<a href="#">readODS</a>	<a href="#">readODS</a>	No
Serialized R objects	.rds	<a href="#">base</a>	<a href="#">base</a>	Yes	HTML Tables	.html	<a href="#">xml2</a>	<a href="#">xml2</a>	No
Epiinfo	.rec	<a href="#">foreign</a>		Yes	Shallow XML documents	.xml	<a href="#">xml2</a>	<a href="#">xml2</a>	No
Minitab	.mtp	<a href="#">foreign</a>		Yes	YAML	.yaml	<a href="#">yaml</a>	<a href="#">yaml</a>	No
Systat	.syd	<a href="#">foreign</a>		Yes	Clipboard	default is tsv	<a href="#">clipr</a>	<a href="#">clipr</a>	No
					<a href="#">Google Sheets</a>	as Comma-separated data			

# 데이터 불러오기와 저장하기

## 내장 데이터 불러오기

- 기본 데이터 세트(내장 데이터)
  - iris # 데이터 세트 이름 입력으로 바로 사용가능
  - co2 # 바로 함수 또는 명령어로 활용 가능
    - str(co2)

```
> co2
```

```
Grouped Data: uptake ~ conc | Plant
  Plant      Type Treatment conc uptake
1  Qn1      Quebec nonchilled  95  16.0
2  Qn1      Quebec nonchilled 175  30.4
3  Qn1      Quebec nonchilled 250  34.8
4  Qn1      Quebec nonchilled 350  37.2
5  Qn1      Quebec nonchilled 500  35.3
6  Qn1      Quebec nonchilled 675  39.2
7  Qn1      Quebec nonchilled 1000 39.7
8  Qn2      Quebec nonchilled  95  13.6
9  Qn2      Quebec nonchilled 175  27.3
10 Qn2      Quebec nonchilled 250  37.1
11 Qn2      Quebec nonchilled 350  41.8
12 Qn2      Quebec nonchilled 500  40.6
13 Qn2      Quebec nonchilled 675  41.4
```

```
> iris # 데이터 세트 이름 입력으로 바로 사용가능
```

```
      Sepal.Length Sepal.Width Petal.Length Petal.Width   species
1         5.1         3.5         1.4         0.2    setosa
2         4.9         3.0         1.4         0.2    setosa
3         4.7         3.2         1.3         0.2    setosa
4         4.6         3.1         1.5         0.2    setosa
5         5.0         3.6         1.4         0.2    setosa
6         5.4         3.9         1.7         0.4    setosa
7         4.6         3.4         1.4         0.3    setosa
8         5.0         3.4         1.5         0.2    setosa
9         4.4         2.9         1.4         0.2    setosa
10        4.9         3.1         1.5         0.1    setosa
11        5.4         3.7         1.5         0.2    setosa
12        4.8         3.4         1.6         0.2    setosa
13        4.8         3.0         1.4         0.1    setosa
14        4.3         3.0         1.1         0.1    setosa
15        5.8         4.0         1.2         0.2    setosa
16        5.7         4.4         1.5         0.4    setosa
17        5.4         3.9         1.3         0.4    setosa
18        5.1         3.5         1.4         0.3    setosa
19        5.7         3.8         1.7         0.3    setosa
20        5.1         3.8         1.5         0.3    setosa
21        5.4         3.4         1.7         0.2    setosa
```

```
> str(co2) # 데이터 구조 확인하기
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData'
 $ Plant      : ord.factor w/ 12 levels "Qn1"<"Qn2"<"Qn3"
 $ Type       : Factor w/ 2 levels "Quebec","Mississippi"
 $ Treatment  : Factor w/ 2 levels "nonchilled","chilled"
 $ conc       : num  95 175 250 350 500 675 1000 95 175 13
 $ uptake     : num  16 30.4 34.8 37.2 35.3 39.2 39.7 13
 - attr(*, "formula")=Class 'formula' language uptake ~
 .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "outer")=Class 'formula' language ~Treatment
 .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
 .. $ x: chr "Ambient carbon dioxide concentration"
```

## 작업경로(폴더) 확인 및 변경

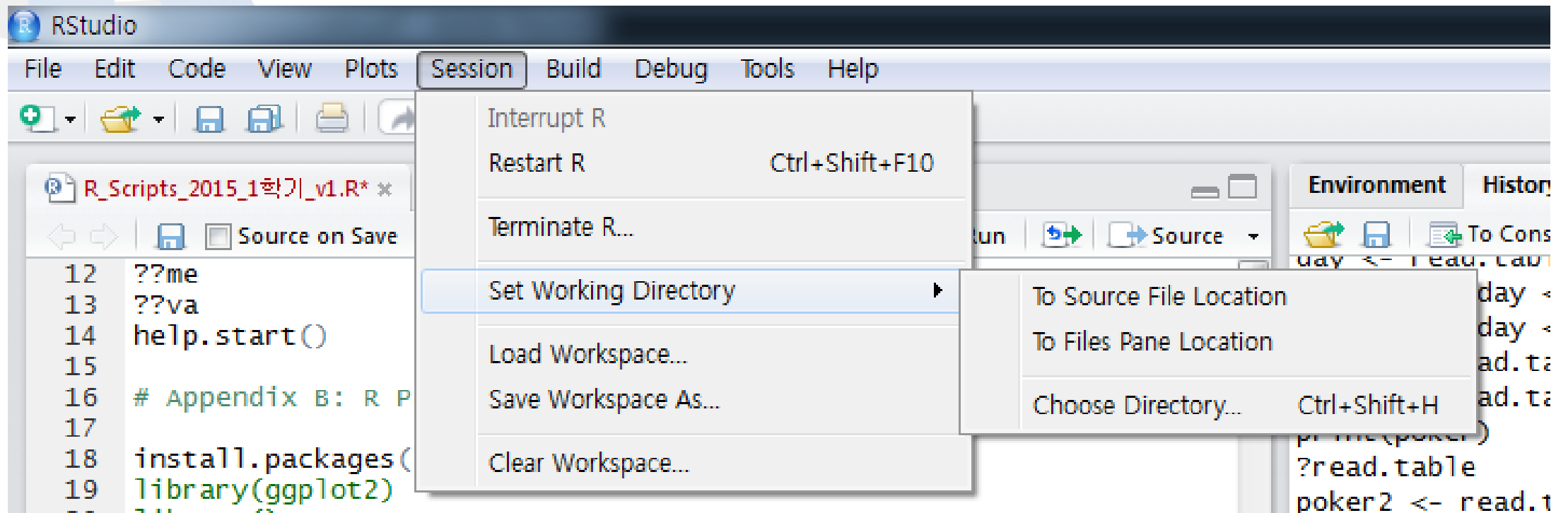
- 작업 경로의 확인 및 변경
  - 외장 데이터는 불러오고자 하는 파일과 그 경로를 알아야 함
  - 작업 경로(폴더) 내에서는 경로 설정 불필요
- 현재 작업 디렉토리 확인: `getwd()`
- 작업 디렉토리의 변경 설정: `setwd()`
  - 폴더(경로) 구분자 사용 주의
    - `'\'` 는 사용 불가능(오류 발생)
    - `'/'` 또는 `'\\'` 사용
  - 파일 디렉토리가 루트 디렉토리에서 시작하지 않을 경우 현재 디렉토리에서 시작 가정
- 작업 디렉토리의 파일 확인: `list.files()`

```
> ##### 작업경로(폴더) 확인하고 변경하기
> ?getwd # 현재 설정된 경로 확인
> getwd()
[1] "E:/강의/수치해석/3주차_R_데이터마이닝_dplyr패키지"
> ?setwd # 앞으로 작업하고자 하는 폴더 또는 그 파일이 있는 폴더의 경로로 설정
> setwd("E:/강의/수치해석/3주차_R_데이터마이닝_dplyr패키지") # 오류 발생
Error: '\? is an unrecognized escape in character string starting "'E:\?
> setwd("E:\\강의\\수치해석\\3주차_R_데이터마이닝_dplyr패키지")
> setwd("E:/강의/수치해석/3주차_R_데이터마이닝_dplyr패키지")
> getwd() # 바뀐 경로 확인
[1] "E:/강의/수치해석/3주차_R_데이터마이닝_dplyr패키지"
```

```
> list.files() # 작업 폴더(경로) 내 파일 목록 확인하기
[1] "~$03_R 데이터마이닝과 dplyr 패키지_v1_2019-09-08.pptx"
[2] "~$선수과목강좌_05_R_컴퓨터프로그래밍_03_R_programming_practi
[3] "03_R 데이터마이닝과 dplyr 패키지_v1_2019-09-08.pptx"
[4] "03_데이터마이닝과 dplyr 패키지_v1_2019-09-06.R"
[5] "d_sigungu.xlsx"
[6] "data_aprt_sale_충북.txt"
[7] "data_by_sigungu.csv"
[8] "data_by_sigungu.xlsx"
[9] "data_by_sigungu_2018.csv"
[10] "data_by_sigungu_2018.xlsx"
[11] "데이터_아파트(매매)_실거래가_충남대전세종.csv"
[12] "데이터_아파트(매매)_실거래가_충남대전세종.xlsx"
[13] "데이터_아파트매매가격.csv"
[14] "데이터_아파트매매가격.xlsx"
[15] "선수과목강좌_04_R_시작하기_입문_v2.docx"
```

## 작업경로(폴더) 확인 및 변경

- Rstudio의 메뉴바로 설정



# 데이터 불러오기와 저장하기

## 외장 데이터 불러오기와 저장하기

- 일반 텍스트 파일
  - 일반 문서에 테이블 형태로 데이터 저장
    - 콤마, 탭, |, 혹은 문자 또는 기호로 각각의 칸을 구분
    - 혼란을 피하기 위하여 하나의 파일에 하나의 기본 구분 규칙 적용
    - 모든 텍스트 파일은 ".txt" 확장자
    - 때로는 구분 기호의 종류를 확실히 하기 위하여 특별한 확장자 사용
      - 예: 콤마 구분기호 파일 확장자 = .csv(comma-separated value)
- 일반 텍스트 파일
  - 불러오기: read.table() 함수
  - 저장하기: write.table() 함수

```
> #### 텍스트 파일 불러오기 저장하기
> ?read.table
> ?read.csv
```

read.table (utils)

R Documentation

### Data Input

#### Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

#### Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"",
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
  row.names, col.names, as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
  dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
  dec = ",", fill = TRUE, comment.char = "", ...)
```

```
> ?write.table
> ?write.csv
```

write.table prints its required argument x (after converting it to a data frame if it is not one nor a matrix) to a file or [connection](#).

#### Usage

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
  eol = "\n", na = "NA", dec = ".", row.names = TRUE,
  col.names = TRUE, qmethod = c("escape", "double"),
  fileEncoding = "")

write.csv(...)
write.csv2(...)
```



# 데이터 불러오기와 저장하기

## 외장 데이터 불러오기와 저장하기

- 일반 텍스트 파일
- read.table() 인수(arguments)
  - file, header, sep, quote, na.string, skip, nrow, .....
  - (file =) "파일경로/파일이름.확장자"
  - sep: 파일 구분기호 지정
  - header: 첫 번째 변수값이 변수이름인지 지정
    - header=TRUE, FALSE
  - na.string: 결측기호 표시 및 확인
    - na.string="." # "."을 NA로 표시
  - skip 또는 nrow: 데이터 세트 특정구간 부터 시작하여 마치기
    - skip : 몇 줄 부터 건너뛰어 데이터 읽어드림
    - nrow : 정해진 줄 수 만큼 읽고 마침
      - 첫번째 header행은 nrow에 포함되지 않음

```
> #### 텍스트 파일 불러오고 저장하기
> ?read.table
> ?read.csv
```

read.table {utils}

R Documentat

### Data Input

#### Description

Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file.

#### Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
  dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
  row.names, col.names, as.is = !stringsAsFactors,
  na.strings = "NA", colClasses = NA, nrows = -1,
  skip = 0, check.names = TRUE, fill = !blank.lines.skip,
  strip.white = FALSE, blank.lines.skip = TRUE,
  comment.char = "#",
  allowEscapes = FALSE, flush = FALSE,
  stringsAsFactors = default.stringsAsFactors(),
  fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
  dec = ".", fill = TRUE, comment.char = "", ...)
```

```
read.csv2(file, header = TRUE, sep = ";", quote = "\"",
  dec = ",", fill = TRUE, comment.char = "", ...)
```

## 외장 데이터 불러오기와 저장하기

- 데이터의 숫자가 아닌 문자열을 불러 들일 때, R은 자동으로 요인으로 불러들임

- 한 개의 파일에서 요인이 아닌 문자로 불러들이고자 할 때의 인수 옵션
  - "stringAsFactors=FALSE" 옵션 지정

```
House <- read.table("I:/R_analysis/house.txt", sep=";", header=TRUE, stringAsFactors=FALSE)
```

- 여러 개의 데이터 세터를 불러 들일 경우,
  - options(stringAsFactors=FALSE)
- 요인으로 불러들이고자 할 때,
  - 기본(default)이므로, 설정하지 않아도 됨
  - 또는 stringAsFactors=TRUE

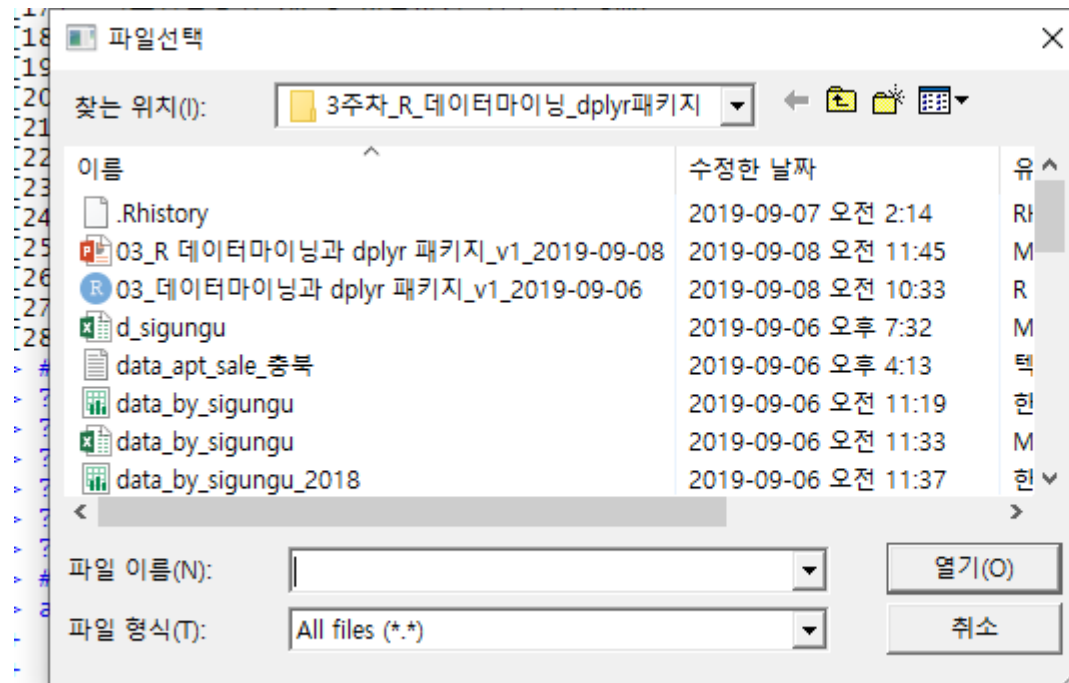
# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- 충청북도 아파트 실거래가 자료(2018.09~2019.08.31) 불러오기

```
> apt_sale <- read.table('데이터_아파트매매가격.csv', # 현재 작업 폴더에 있는 파일 이름
+                       sep="," # 구분자 = ","
+                       , header = TRUE, # 첫행을 변수명으로
+                       stringsAsFactors = FALSE) # 명목변수(열)은 요인이 아닌 문자로 불러오기
```

```
> apt_sale <- read.table(file.choose()) # 파일 이름을 모를 때 사용 또는 list.files() 함수 사용하여 확인
```



LYR\_package\_v3\_2019-06-03.pdf"  
LYR\_package\_v3\_2019-06-03.pptx  
019-04-23.R"

```
stringsAsFactors = FALSE) # 명목변수(열)은 요인이 아닌 문자로 불러오기
> apt_sale <- read.table(file.choose()) # 파일 이름을 모를 때 사용 또는 list.files() 함수 사용하여 확인
```

## 외장 데이터 불러오고 저장하기

- 데이터 확인하기

- 데이터 구조: str()

```
> str(apt_sale) # 데이터 구조 확인하기
'data.frame': 13309 obs. of 14 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price : int  4800 4500 4000 4000 4000 4000 4000 4000 4000 4000
 $ area_m2  : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8
 $ floor_no : int  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built : int  1998 1998 1998 1998 1998 1998 1998 1998 1998
 $ ym_sale  : int  201811 201811 201812 201812 201812 201812 201812
 $ day_sale : int  6 13 10 10 10 10 10 10 10 10 ...
 $ urban    : chr  "읍" "읍" "읍" "읍" ...
 $ apt_complex : chr  "두진백로" "두진백로" "두진백로" "두진백로" ...
 $ address   : chr  "충청북도 괴산군 괴산읍 대사리" "충청북도 괴산군 괴신
 $ address_bunji : chr  "177-1" "177-1" "177-1" "177-1" ...
 $ address_sido : chr  "충청북도" "충청북도" "충청북도" "충청북도" ...
 $ address_sigungu : chr  "괴산군" "괴산군" "괴산군" "괴산군" ...
 $ address_dong : chr  "괴산읍" "괴산읍" "괴산읍" "괴산읍" ...
```

- 데이터 열 속성 확인

- is.character()

- is.factor()

- 데이터 수정

- 요인으로 데이터 변환: as.factor()

- ";"로 두개 명령문 한줄에서 동시 실행 가능

```
> is.character(apt_sale$urban) # 데이터의 열($, 키)로 내부 열 urban 문자인 지 여부 확인
[1] TRUE
> apt_sale$urban <- as.factor(apt_sale$urban) # 문자형 변수를 요인형 변수로 변환하기
> is.character(apt_sale$urban) ; is.factor(apt_sale$urban) # 문자, 요인변수 확인,
[1] FALSE
[1] TRUE
```

## 외장 데이터 불러오고 저장하기

- 일반 텍스트 파일

- 저장하기

- write.table()

- 기존 파일명이 같은 경우에는 덮어쓰기가 되므로, 저장할 파일 이름 명명시 주의 필요

```
> ##### 데이터 텍스트 파일로 저장하기
> write.table(aprt_sale, # 저장할 객체(데이터) 이름
+             'data_aprt_sale_충북.txt', # 저장하고자 하는 파일 이름 명명(다른 폴더 저장시 해당 경로 설정)
+             sep=",") # 구분자 설정
```

- 저장한 파일 확인하기

- file.exists()

- list.files()

```
> file.exists('data_aprt_sale_충북.txt')
[1] TRUE
> list.files()
[1] "~$03_R 데이터마이닝과 dplyr 패키지_v2_2019-09-08.pptx"
[2] "~$선수과목강좌_05_R_컴퓨터프로그래밍_03_R_programming_pr
[3] "03_R 데이터마이닝과 dplyr 패키지_v1_2019-09-08.pptx"
[4] "03_R 데이터마이닝과 dplyr 패키지_v2_2019-09-08.pptx"
[5] "03_데이터마이닝과 dplyr 패키지_v1_2019-09-06.R"
[6] "d_sigungu.xlsx"
[7] "data_aprt_sale_충북.txt"
[8] "data_by_sigungu.csv"
[9] "data_by_sigungu.xlsx"
... ..
```

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저

- 일반 텍스트 파일

- “readr” 패키지로 데이터 불러오기와 저장하기 활용

```
> install.packages("readr")
WARNING: Rtools is required to build R packages but
fore proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/hgs_home/Documents
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/con
Content type 'application/zip' length 1585643 bytes
downloaded 1.5 MB

package 'readr' successfully unpacked and MD5 sums c
The downloaded binary packages are in
C:\Users\Public\Documents\ESTsoft\CreatorTem
> library(readr)
> ?readr
```

```
# The easiest way to get readr is to install the whole tidyverse:
install.packages("tidyverse")
```

```
# Alternatively, install just readr:
install.packages("readr")
```

```
# Usage
```

```
library(tidyverse) or library(readr)
```

### readr: Read Rectangular Text Data

#### Description

The goal of 'readr' is to provide a fast and friendly way to read rectangular data (like 'csv', 'tsv', and 'fwf'). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes.

#### Author(s)

Maintainer: Jim Hester [james.hester@rstudio.com](mailto:james.hester@rstudio.com)

#### Authors:

- Hadley Wickham [hadley@rstudio.com](mailto:hadley@rstudio.com)
- Romain Francois

#### Other contributors:

- R Core Team (Date time code adapted from R) [contributor]
- RStudio [copyright holder, funder]



## 외장 데이터 불러오고 저장하기

- 일반 텍스트 파일
  - “readr” 패키지로 데이터 불러오기 활용
    - The goal of readr is to provide **a fast** and friendly way to read rectangular data (like csv, tsv, and fwf).
    - readr supports seven file formats with seven read\_ functions:
      - [read\\_csv\(\)](#): comma separated (CSV) files
      - [read\\_tsv\(\)](#): tab separated files
      - [read\\_delim\(\)](#): general delimited files
      - [read\\_fwf\(\)](#): fixed width files
      - [read\\_table\(\)](#): tabular files where columns are separated by white-space.
      - [read\\_log\(\)](#): web log files

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- “readr” 패키지로 데이터 불러오기
  - 데이터 불러오기: read\_csv()

```
> apt_sale2 <- read_csv('데이터_아파트매매가격.csv', # 파일 경로와 이름
+                       locale=locale('ko',encoding='euc-kr')) # 한글 설정
Parsed with column specification:
cols(
  id = col_double(),
  apt_price = col_double(),
  area_m2 = col_double(),
  floor_no = col_double(),
  year_built = col_double(),
  ym_sale = col_double(),
  day_sale = col_double(),
  urban = col_character(),
```

- 데이터 확인하기: head(), str(), View()

```
> head(apt_sale2) # 6째 행까지 데이터 확인
# A tibble: 6 x 14
   id apt_price area_m2 floor_no year_built ym_sale day_sale urb
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1     1     4800   39.8       2     1998   201811      6   이진백로
2     2     4500   39.8       6     1998   201811     13   이진백로
3     3     4000   39.8       7     1998   201812     10   이진백로
4     4     4000   39.8       9     1998   201812     10   이진백로
5     5     4000   39.8       3     1998   201812     10   이진백로
6     6     4000   39.8       7     1998   201812     10   이진백로
# ... with 1 more variable: address_dong <chr>
> str(apt_sale2) # 데이터 구조 확인하기
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':      13
 $ id          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price   : num  4800 4500 4000 4000 4000 4000 4000 4000 4000 4
 $ area_m2     : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 3
 $ floor_no    : num  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built  : num  1998 1998 1998 1998 1998 1998 ...
 $ ym_sale     : num  201811 201811 201812 201812 201812 ...
 $ day_sale    : num  6 13 10 10 10 10 10 10 10 10 ...
 $ urban       : chr  "은" "은" "은" "은"
```

> view(apt\_sale2) # 새로운 창에서 데이터 확인

	id	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban	apt_complex	a
1	1	4800	39.8000	2	1998	201811	6	은	이진백로	은
2	2	4500	39.8000	6	1998	201811	13	은	이진백로	은
3	3	4000	39.8000	7	1998	201812	10	은	이진백로	은
4	4	4000	39.8000	9	1998	201812	10	은	이진백로	은
5	5	4000	39.8000	3	1998	201812	10	은	이진백로	은
6	6	4000	39.8000	7	1998	201812	10	은	이진백로	은
7	7	4000	39.8000	5	1998	201812	10	은	이진백로	은



## 외장 데이터 불러오고 저장하기

- “readr” 패키지로 데이터 저장하기

- `write_csv()` Write a data frame to a delimited file

### Description

This is about twice as fast as `write.csv()`, and never writes row names.  
`output_column()` is a generic method used to coerce columns to suitable output.

### Usage

```
write_delim(x, path, delim = " ", na = "NA", append = FALSE,
  col_names = !append, quote_escape = "double")

write_csv(x, path, na = "NA", append = FALSE, col_names = !append,
  quote_escape = "double")

> write_csv(apt_sale2, # 저장할 객체
+           "test.csv") # 저장할 파일 이름 명명
> file.exists('test.csv')
[1] TRUE
```

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- 엑셀 스프레드 시트(파일) 불러오기
  - 엑셀 파일 입출력을 위한 패키지들 많음
    - "readxl"과 "xlsx" 패키지 활용
  - Importing Excel files into R using readxl package
    - `install.packages("readxl")` # 엑셀 불러오고 저장하기 패키지 설치
    - `library("readxl")` # 엑셀 패키지 불러오기(요청) 또는
    - `library(readxl)`
- R 프로그램 자체는 자바가 없어도 실행이 가능하지만 대부분의 패키지를 사용하기 위해서는 자바가 필요
  - <https://www.java.com/ko/>
  - 32bit vs. 64bit : 64 bit 버전 수동 설치
  - <https://www.java.com/en/download/manual.jsp>
- Importing Excel files using xlsx package
  - `install.packages("xlsx")`
  - `library("xlsx")`

```
> install.packages("readxl") # 엑셀 불러오고 저장하기 패키지 설치
Installing package into 'C:/Users/hyun/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
also installing the dependencies 'magrittr', 'rematch', 'prettyunits', 'cellranger'
```

```
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/magrittr_1.5.zip'
Content type 'application/zip' length 157877 bytes (154 KB)
downloaded 154 KB
```

```
> library("readxl") # 엑셀 패키지 불러오기(요청)
> library(readxl)
```

```
> install.packages("xlsx")
WARNING: Rtools is required to build R packages but is not currently installed

https://cran.rstudio.com/bin/windows/Rtools/
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/xlsx_0.6.1.zip'
Content type 'application/zip' length 460695 bytes (449 KB)
downloaded 449 KB
```

```
package 'xlsx' successfully unpacked and MD5 sums checked
```

```
The downloaded binary packages are in
C:\Users\user\AppData\Local\Temp\RtmpcNodCH\downloaded_packages
```

```
> library(xlsx) # 자바(java)와 매칭되어야 하므로, 최신 자바버전 및 64bit 설치 필요
```

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- readxl package로 엑셀 파일 불러오기

> ?read\_excel

```
> data_sigungu <- read.xlsx("data_by_sigungu_2018.xlsx", # file: 파일 경로와 이름
+                           sheetName="data", # 불러들일 시트 이름
+                           header=TRUE, # 첫행을 열의 변수명으로 함
+                           encoding="UTF-8") # 한글 코드 인식
```

```
> str(data_sigungu) # 데이터 구조 확인하기
'data.frame': 261 obs. of 16 variables:
 $ id          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ sido        : Factor w/ 17 levels "강원도","경기도",...: 9 9 9 9
 $ sigungu     : Factor w/ 236 levels "가평군","강남구",...: 190 191
 $ area        : num  23.91 9.96 21.87 16.86 17.06 ...
 $ pop_density : num  6387 12574 10529 17750 20935 ...
 $ pop_tot     : num  154770 125709 229161 304808 357703 ...
 $ pop_male    : num  75967 62253 110878 150368 174414 ...
 $ pop_female  : num  78803 63456 118283 154440 183289 ...
 $ age_average : Factor w/ 132 levels "-","35.9","36.2",...: 48 54
 $ r_aged      : num  17.1 17.1 16 13.6 12.5 ...
 $ pop_net_move : num  -204 143 -89 57 -608 -469 -429 -716 -521 -3
 $ housing     : num  45729 40177 71680 87830 85638 ...
 $ apt_sale_price : num  102 104 105 105 104 ...
 $ apt_junse_price : num  102 101 100 100 102 ...
 $ car         : num  0.33 0.41 0.34 0.34 0.28 0.28 0.28 0.27 0.2
 $ accident_per_1000car: num  18.82 18.72 14.64 8.54 8.72 ...
```

### Read xls and xlsx files

#### Description

Read xls and xlsx files

read\_excel() calls `excel_format()` to determine if path is xls or xlsx, based on the file extension and the file itself, in that order. Use `read_xls()` and `read_xlsx()` directly if you know better and want to prevent such guessing.

#### Usage

```
read_excel(path, sheet = NULL, range = NULL, col_names = TRUE,
  col_types = NULL, na = "", trim_ws = TRUE, skip = 0,
  n_max = Inf, guess_max = min(1000, n_max),
  progress = readxl_progress(), .name_repair = "unique")
```

```
read_xls(path, sheet = NULL, range = NULL, col_names = TRUE,
  col_types = NULL, na = "", trim_ws = TRUE, skip = 0,
  n_max = Inf, guess_max = min(1000, n_max),
  progress = readxl_progress(), .name_repair = "unique")
```

```
read_xlsx(path, sheet = NULL, range = NULL, col_names = TRUE,
  col_types = NULL, na = "", trim_ws = TRUE, skip = 0,
  n_max = Inf, guess_max = min(1000, n_max),
  progress = readxl_progress(), .name_repair = "unique")
```

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- xlsx package로 엑셀 파일 불러오기
  - read\_xlsx2() : 불러들이는 속도 빠름. 그러나 요인으로 인식

```
> data_sigungu2 <- read.xlsx2("data_by_sigungu_2018.xlsx", # file: 파일 경로와 이름
+                               sheetIndex= 1, # 첫번째 시트를 불러들임
+                               header=TRUE) # 첫행을 열의 변수명으로 함
> str(data_sigungu2) # 모든 변수들이 요인(Factor)으로 인식됨
'data.frame': 261 obs. of 16 variables:
 $ id          : Factor w/ 261 levels "1","10","100",...: 1 112 185 196 207 2...
 $ sido        : Factor w/ 17 levels "강원도","경기도",...: 9 9 9 9 9 9 9 9 9 9 ...
 $ sigungu      : Factor w/ 236 levels "가평군","강남구",...: 190 191 161 110 28 ...
 $ area         : Factor w/ 229 levels "", "10.21", "1009.24",...: 57 215 52 34 ...
 $ pop_density  : Factor w/ 229 levels "", "10212.16",...: 173 23 8 64 85 94 89 ...
 $ pop_tot      : Factor w/ 261 levels "1004081", "100615",...: 29 23 62 113 14 ...
 $ pop_male     : Factor w/ 261 levels "100613", "102876",...: 241 235 11 62 89 ...
 $ pop_female   : Factor w/ 261 levels "100652", "101050",...: 243 235 26 66 95 ...
 $ age_average  : Factor w/ 132 levels "-", "35.9", "36.2",...: 48 54 48 40 36 4 ...
 $ r_aged       : Factor w/ 214 levels "", "10.18", "10.88",...: 83 82 73 44 29 ...
 $ pop_net_move : Factor w/ 193 levels "", "-1", "-105",...: 25 114 101 167 85 6 ...
```

- 숫자형 변수로 변환: as.numeric()

```
> data_sigungu2$area <- as.numeric(data_sigungu2$area) # 시군구 면적: 요인형 변수를 숫자형 변수로 변환하기
> data_sigungu2$pop_density <- as.numeric(data_sigungu2$pop_density) # 인구밀도 : 요인형 변수를 숫자형 변수로 변환하기
> str(data_sigungu2)
'data.frame': 261 obs. of 16 variables:
 $ id          : Factor w/ 261 levels "1","10","100",...: 1 112 185 196 207 218 229 240 251 2 ...
 $ sido        : Factor w/ 17 levels "강원도","경기도",...: 9 9 9 9 9 9 9 9 9 9 ...
 $ sigungu      : Factor w/ 236 levels "가평군","강남구",...: 190 191 161 110 28 67 192 111 5 64 ...
 $ area         : num 57 215 52 34 36 25 43 60 55 50 ...
 $ pop_density  : num 173 23 8 64 85 94 89 73 32 56 ...
 $ pop_tot      : Factor w/ 261 levels "1004081", "100615",...: 29 23 62 113 142 140 162 182 120 134 ...
 $ pop_male     : Factor w/ 261 levels "100613", "102876",...: 241 235 11 62 89 91 114 125 70 79 ...
```

# 데이터 불러오기와 저장하기

## 외장 데이터 불러오고 저장하기

- 엑셀 파일 저장하고 확인하기

- 저장하기: `write.xlsx()`
- 확인하기:
  - `file.exists('d_sigungu.xlsx')`
  - `list.files()` # 폴더내 파일 확인

```
> ?write.xlsx # write a data.frame to an Excel workbook
```

Write a data.frame to an Excel workbook.

### Description

Write a data.frame to an Excel workbook.

### Usage

```
write.xlsx(x, file, sheetName="Sheet1",  
           col.names=TRUE, row.names=TRUE, append=FALSE, showNA=TRUE, pas  
  
write.xlsx2(x, file, sheetName="Sheet1",  
            col.names=TRUE, row.names=TRUE, append=FALSE, password=NULL, .
```

```
> write.xlsx(data_sigungu2, # 저장할 객체 이름  
+           'd_sigungu.xlsx', # 저장할 엑셀 파일 이름  
+           sheetName="data") # 엑셀 시트 이름  
> file.exists('d_sigungu.xlsx')  
[1] TRUE  
> list.files()
```

## 연습문제 01

- 다음 중 R에서 rJava 패키지와 연동되어야 작동하는 함수(명령어)를 모두 고르시오.
  - ① read.table
  - ② read.csv
  - ③ write.csv
  - ④ write.xlsx
  - ⑤ read.xlsx
  - ⑥ read\_xlsx
  - ⑦ read\_xls
  - ⑧ read\_excel

## 연습문제 02

- 아래와 같은 폴더내에 있는 csv파일을 불러들여 x라는 객체에 할당하고자 할 때, 사용하게 될 올바른 명령문(함수)를 적으시오.
  - I:\강의\수치해석\3주차\데이터\_아파트매매가격.csv

## 데이터 탐구하기

- 데이터 불러오기

```
> apt <- read.table('데이터_아파트매매가격.csv',  
+                   sep=";", # 구분자 = ";",  
+                   header = TRUE, # 첫행을 변수명으로  
+                   stringsAsFactors = TRUE) # 명목변수(열)은 요인으로 불러오기
```

- 데이터 확인하기

- str()
- head()
- tail()

```
> str(apt) # 데이터 구조  
'data.frame': 13309 obs. of 14 variables:  
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...  
 $ apt_price : int  4800 4500 4000 4000 4000 4000 4000 4000 ...  
 $ area_m2  : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 ...  
 $ floor_no : int  2 6 7 9 3 7 5 9 8 5 ...  
 $ year_built : int  1998 1998 1998 1998 1998 1998 1998 1998 ...  
 $ ym_sale   : int  201811 201811 201812 201812 201812 201812 201812 201812 ...  
 $ day_sale  : int  6 13 10 10 10 10 10 10 10 10 ...  
 $ urban     : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
```

```
> head(apt) # 첫 행부터 6번째 행까지 데이터 확인  
  id apt_price area_m2 floor_no year_built ym_sale day_sale  
1  1      4800    39.8        2      1998  201811         6  
2  2      4500    39.8        6      1998  201811        13  
3  3      4000    39.8        7      1998  201812        10  
4  4      4000    39.8        9      1998  201812        10  
5  5      4000    39.8        3      1998  201812        10  
6  6      4000    39.8        7      1998  201812        10
```

```
> tail(apt) # 맨 마지막행 부터 6번째 행까지 데이터 확인  
      id apt_price area_m2 floor_no year_built ym_sale  
13304 13304    11300    84.68        15      2002  201907  
13305 13305    10700    84.68         3      2002  201907  
13306 13306    10700    84.68         7      2002  201907  
13307 13307    12800    84.68        14      2002  201908  
13308 13308    12500    84.68        14      2002  201908  
13309 13309    12300    84.68         2      2002  201908
```



## 데이터 탐구하기

- 데이터 확인하기

- View() : Invoke a spreadsheet-style data viewer on a matrix-like **R** object.

- 'V'는 대문자

```
> View(apt) # 창으로 데이터 확인
```

	id	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban	apt_complex
1	1	4800	39.8000	2	1998	201811	6	읍	두진백로
2	2	4500	39.8000	6	1998	201811	13	읍	두진백로
3	3	4000	39.8000	7	1998	201812	10	읍	두진백로
4	4	4000	39.8000	9	1998	201812	10	읍	두진백로

- summary()

```
> summary(apt) # 요약통계
```

id		apt_price		area_m2		floor_no	
Min.	: 1	Min.	: 1250	Min.	: 14.98	Min.	: -1.000
1st Qu.:	3328	1st Qu.:	7700	1st Qu.:	58.93	1st Qu.:	4.000
Median :	6655	Median :	13500	Median :	60.00	Median :	8.000
Mean :	6655	Mean :	14553	Mean :	68.92	Mean :	8.835
3rd Qu.:	9982	3rd Qu.:	19500	3rd Qu.:	84.89	3rd Qu.:	12.000
Max.	:13309	Max.	:108000	Max.	:244.07	Max.	:48.000

## 데이터 탐구하기

- 데이터 분포 및 요약 통계

- `hist(apt$apt_price)` # 아파트 실거래 가격(만원) 히스토그램

- `?quantile()` # produces quantiles corresponding to the given probabilities

- `quantile(apt$apt_price, probs = seq(0, 1, 0.1))` # 10분위

- `quantile(apt$apt_price, probs = seq(0.9, 1, 0.01))` # 90분위 부터 1% 씩 가격 확인

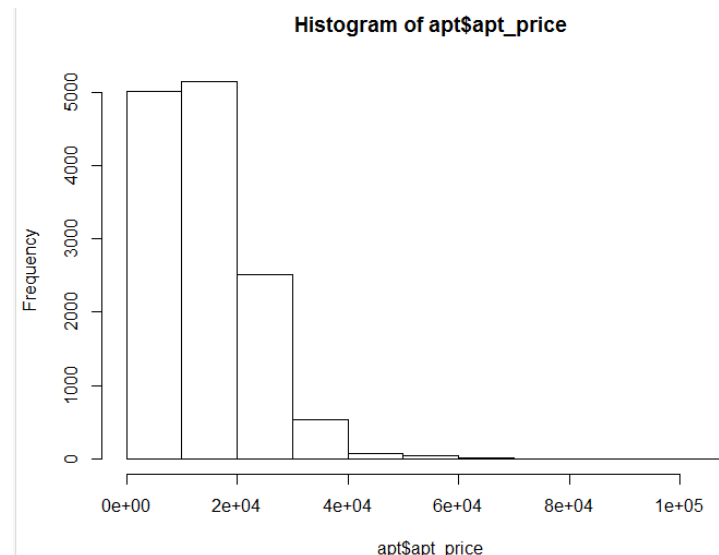
```
> quantile(apt$apt_price, probs = seq(0, 1, 0.1)) # 10분위
```

0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
1250	4700	6800	8500	10900	13500	15370	18000	21800	25500	108000

```
> quantile(apt$apt_price, probs = seq(0.9, 1, 0.01)) # 90분위 부터 1% 씩 가격 확인
```

90%	91%	92%	93%	94%	95%	96%	97%	98%	99%	100%
25500	26000	27000	28000	29000	30000	31500	33876	36200	39492	108000

```
> hist(apt$apt_price) # 아파트 실거래 가격(만원) 히스토그램
```



## 데이터 탐구하기

- 자료의 크기와 길이 확인
  - `nrow(apt)` # 행의 갯수
  - `ncol(apt)` # 열의 갯수
  - `dim(apt)` # 행과 열의 갯수

```
> nrow(apt) # 행의 갯수
[1] 13309
> ncol(apt) # 열의 갯수
[1] 14
> dim(apt) # 행과 열의 갯수
[1] 13309      14
```

## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)
  - 데이터 접근: 색인([ ]) 또는 키(\$)로 접근 가능
  - \$
    - d\$colname: 데이터 프레임 d의 colname에 저장된 데이터
    - d\$colname[n]: 데이터 프레임 d의 colname의 n번째 행
- d[m, n, drop=TRUE]
  - 데이터 프레임 d의 m행 n열 컬럼에 저장된 데이터
  - d[n] : n번째 요소, 숫자 또는 셀의 이름
  - d[-n] : n번째 요소를 제외한 나머지
  - d[start:end] : 벡터의 start부터 end까지의 값을 반환함. 시작과 끝 포함

## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)

- 예:

```
> apt[1, 2] # 첫째행과 두번째 열에 있는 값은?  
[1] 4800  
> apt[13309, -2:-14] # 13309행과 2번째 부터 14번째 열을 제외한 열들에 있는 값은?  
[1] 13309  
> apt[1:2, 3:5] # 1:2행과 3:5열에 있는 값들은?  
  area_m2 floor_no year_built  
1    39.8        2    1998  
2    39.8        6    1998  
> apt$floor_no[3] # apt 객체 중 층수(floor_no) 열에서 3번째 행에 있는 값은?  
[1] 7  
> apt$floor_no[c(-1:-2, -4:-13309)] # 위와 동일  
[1] 7
```

## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)
  - 이상치 또는 특정 자료 확인
    - which() # Which indices are TRUE?

```
> which(apt$floor_no %in% -1) # -1이 포함된 행 번호 반환
[1] 1882 1883 1886 1889 1891
> apt[c(1882, 1883, 1886, 1889, 1891 ), ]
      id apt_price area_m2 floor_no year_built ym_sale
1882 1882      2500   56.88       -1      1989  201902
1883 1883      5000   56.88       -1      1989  201902
1886 1886      2000   56.88       -1      1989  201905
1889 1889      3500   56.88       -1      1989  201907
1891 1891      2400   38.25       -1      1989  201907
```

## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)

- 이상치 또는 특정 자료 확인

- %in%:

- i %in% j : i 객체에서 j객체에 포함되어 있는 색인 번호 반환

```
> which(apt$floor_no %in% -1) # -1이 포함된 행 번호 반환
```

```
[1] 1882 1883 1886 1889 1891
```

- 특정 조건에 맞는 자료 확인

```
> i <- which(apt$apt_price > 60000) # 실거래가가 6억 이상인 아파트 색인
```

```
> i
```

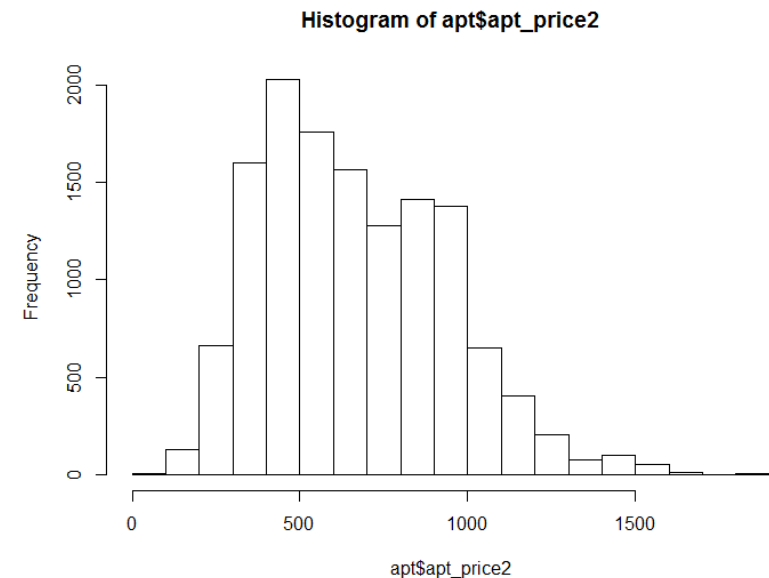
```
[1] 9147 9156 10299 10303 10316 10323 10324 10349 10361
```

```
> apt[i, c(2:9, 14)] # 해당 거래 자료 보기
```

	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban	apt_complex	address_dong
9147	63000	131.941	24	2015	201809	28	동	대원칸타빌3차아파트	주성동
9156	67000	131.941	23	2015	201901	10	동	대원칸타빌3차아파트	주성동
10299	68000	162.626	45	2010	201810	18	동	신영지웰시티 1차	북대동
10303	60800	152.652	6	2010	201811	3	동	신영지웰시티 1차	북대동
10316	97300	196.976	42	2010	201901	13	동	신영지웰시티 1차	북대동
10323	105000	196.976	42	2010	201902	13	동	신영지웰시티 1차	북대동
10324	64000	152.652	24	2010	201902	20	동	신영지웰시티 1차	북대동
10349	108000	196.976	41	2010	201905	25	동	신영지웰시티 1차	북대동
10361	64500	152.652	39	2010	201907	19	동	신영지웰시티 1차	북대동

## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)
  - 특정 조건에 맞는 자료 확인
    - 새로운 변수(열) 생성
      - 평당 가격 변수 생성
        - <-
      - 요약통계
      - 히스토그램



```
> apt$apt_price2 <- apt$apt_price/apt$area_m2*3.3 # 평당 아파트 가격(만원) 변수 생성
> summary(apt$apt_price2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  85.48  444.44  627.35  666.15  866.05 1825.47
> hist(apt$apt_price2)
```



## 데이터 탐구하기

- 색인(index) 또는 키(\$)로 데이터 접근하기(찾아보기)
- 특정 조건에 맞는 자료 확인
  - 아파트 가격이 평당 1600만원 이상 자료 확인

- which()

```
> j <- which(apt$apt_price2 >= 1600) # 아파트 가격이 평당 1600만원 이상으로 거래된 자료 행 번호
> apt[j, c(2:10, 15)] # 해당 거래 자료 보기
```

	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban	apt_complex
4193	60000	108.4651	29	2018	201905	11	동	청주 센트럴자이
9156	67000	131.9410	23	2015	201901	10	동	대원칸타빌3차아파트
9998	39800	80.1350	39	2015	201809	27	동	두산위브지웰시티2차
10075	39000	80.1350	39	2015	201905	12	동	두산위브지웰시티2차
10108	39000	80.1350	34	2015	201907	6	동	두산위브지웰시티2차
10112	39400	80.4990	39	2015	201907	8	동	두산위브지웰시티2차
10121	39500	80.1350	30	2015	201907	23	동	두산위브지웰시티2차
10143	39500	80.4990	31	2015	201908	16	동	두산위브지웰시티2차
10316	97300	196.9760	42	2010	201901	13	동	신영지웰시티 1차
10323	105000	196.9760	42	2010	201902	13	동	신영지웰시티 1차
10349	108000	196.9760	41	2010	201905	25	동	신영지웰시티 1차

```
> apt[j, -11:-14 ] # 해당 거래 자료 보기
```

	id	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban	apt_complex
4193	4193	60000	108.4651	29	2018	201905	11	동	청주 센트럴자이
9156	9156	67000	131.9410	23	2015	201901	10	동	대원칸타빌3차아파트
9998	9998	39800	80.1350	39	2015	201809	27	동	두산위브지웰시티2차
10075	10075	39000	80.1350	39	2015	201905	12	동	두산위브지웰시티2차
10108	10108	39000	80.1350	34	2015	201907	6	동	두산위브지웰시티2차

# 데이터 마이닝 기초

## 데이터 탐구하기

- 문자(명목)변수 탐구하기
  - 요인(factor) 변수 고유값 및 빈도수 테이블 생성과 확인
    - str(apt)
    - unique(apt\$urban) # 읍면동 고유값 확인
    - table(apt\$urban) # 빈도
  - unique(apt\$address\_sigungu) # 시군구 고유값 확인
  - table(apt\$address\_sigungu) # 빈도
  - table(apt[ c('urban', 'address\_sigungu')])

```
> unique(apt$urban) # 읍면동 고유값 확인
[1] 읍 면 동
Levels: 동 면 읍
> table(apt$urban) # 빈도
```

동	면	읍
8848	1332	3129

```
>
> unique(apt$address_sigungu) # 시군구 고유값 확인
[1] 괴산군 단양군 보은군 영동군 옥천군 음성군 제천시 증평군
[13] 청주흥덕구 충주시
Levels: 괴산군 단양군 보은군 영동군 옥천군 음성군 제천시 증평군 진천군 청주상당구 청주서원구 청주시
> table(apt$address_sigungu) # 빈도
```

괴산군	단양군	보은군	영동군	옥천군	음성군	제천시	증평군
48	135	102	198	213	897	1264	

```
>
> table(apt[ c('urban', 'address_sigungu')])
```

		address_sigungu											
		괴산군	단양군	보은군	영동군	옥천군	음성군	제천시	증평군	진천군	청주상당구	청주서원구	청주시
urban	동	0	0	0	0	0	0	1215	0	0	1468		
	면	1	0	6	8	4	579	0	3	175	7		
	읍	47	135	96	190	209	318	49	228	529	0		

## 데이터 탐구하기

- 순서(서열)형 요인 변수 생성과 확인

- ordered()
- str()
- is.ordered()
- nlevels()

```
> apt$day_sale <- ordered(apt$day_sale) # 순서형 팩터 생성
> str(apt$day_sale)
  Ord.factor w/ 31 levels "1"<"2"<"3"<"4"<...: 6 13 10 10 10 10 10 10 10 10
> is.ordered(apt$day_sale) # 순서형 팩터 여부 판단
[1] TRUE
> nlevels(apt$day_sale) # 순서형 팩터 라벨 갯수 반환
[1] 31
```

## 데이터 탐구하기

- 하위데이터의 요약 통계: aggregate()
  - aggregate(x, by, FUN, ..., simplify = TRUE, drop = TRUE)
    - i ~ j : j 집단(그룹)별로 i에 대하여 요약 통계를 구하라
      - one ~ one, one ~ many, many ~ one, and many ~ many

```
> aggregate(apt_price ~ urban, FUN=mean, data=apt) # 읍면동(urban) 별 평균 실거래 가격
```

	urban	apt_price
1	북촌	14724.22
2	북촌	12624.81
3	북촌	14888.65

## 데이터 탐구하기

- 하위데이터의 요약 통계: aggregate()
  - aggregate(x, by, FUN, ..., simplify = TRUE, drop = TRUE)
  - Manty to one: cbind() 함수 활용
  - Manty to many

```
> ?cbind # Combine R Objects by Columns: 행별 결합은 rbind()
> x <- aggregate(cbind(apt_price, apt_price2)
+               ~ urban, FUN=mean, data=apt) # 읍면동(urban)별 평균 거래 가격과 평당 가격
> x
  urban apt_price apt_price2
1   동   14724.22    674.3314
2 읍면   12624.81    612.8632
3 읍   14888.65    665.6805
> x2 <- aggregate(cbind(apt_price, apt_price2)
+               ~ address_sigungu + urban, FUN=mean, data=apt) # 시군구별 읍면동(urban)별 평균 거래 가격과 평당 가격
> head(x2)
address_sigungu urban apt_price apt_price2
1   제천시   동   11466.14    549.6755
2 청주상당구   동   14801.04    656.3173
3 청주서원구   동   15068.47    701.4172
4 청주청원구   동   16971.64    740.2430
5 청주흥덕구   동   18577.32    845.0420
6   충주시   동   10719.81    509.1005
```

## 데이터 탐구하기

- 하위데이터의 요약 통계: aggregate()
  - aggregate(x, by, FUN, ..., simplify = TRUE, drop = TRUE)
  - 색인으로 추출

```
> aggregate(apr[, 'apt_price'], apr[, 'urban', drop=FALSE], mean)
  urban      x
1   동 14724.22
2   면 12624.81
3   읍 14888.65
> aggregate(apr[, c('apt_price', 'apt_price2')], apr[, 'urban', drop=FALSE], mean, na.rm=TRUE)
  urban apt_price apt_price2
1   동  14724.22    674.3314
2   면  12624.81    612.8632
3   읍  14888.65    665.6805
> aggregate(apr[, c('apt_price', 'apt_price2')], apr[, 'ym_sale', drop=FALSE], mean, na.rm=TRUE)
  ym_sale apt_price apt_price2
1  201809  14744.04    676.6967
2  201810  15813.51    708.7854
3  201811  14470.37    662.3380
4  201812  13886.61    660.3361
5  201901  14776.15    658.9688
6  201902  14433.34    653.5126
7  201903  13958.03    636.7984
8  201904  14037.45    639.5075
9  201905  14350.37    653.7078
10 201906  14151.45    665.0134
11 201907  14492.23    666.2532
12 201908  15235.40    706.1628
```

## 데이터 탐구하기

- 하위데이터의 요약 통계: aggregate()
  - aggregate(x, by, FUN, ..., simplify = TRUE, drop = TRUE)
    - 'by' 인수 활용 예

```
> y <- aggregate(apt[ , c(2:5, 15)], # 집계화하고자 하는 데이터 프레임
+               by = list(apt$year_built), # 그룹화할 변수들의 리스트
+               FUN = mean) # 계산하고자 하는 함수( Maximum, minimum, count, standard deviation and sum)
> head(y)
```

	Group.1	apt_price	area_m2	floor_no	year_built	apt_price2
1	1977	10016.667	64.15333	1.666667	1977	515.9706
2	1978	8728.571	64.46286	3.142857	1978	452.9652
3	1979	5104.545	48.50227	3.363636	1979	350.0775
4	1980	3705.926	47.60963	2.925926	1980	237.7294
5	1981	4508.919	43.93459	3.243243	1981	342.3330
6	1982	8215.000	64.60550	3.450000	1982	411.6612

## 데이터 탐구하기

- 행과 열의 집단별 요약통계: lapply, sapply, tapply
  - apply() # Apply Functions Over Array Margins: apply(X, MARGIN, FUN, ...)
  - lapply() # Apply a Function over a List or Vector
  - sapply() # a user-friendly version and wrapper of lapply by default returning a vector, matrix, or others
  - tapply() # Apply a Function Over a Ragged Array

```
> ?tapply # Apply a Function Over a Ragged Array
> tapply(apt$apt_price, apt$urban, mean) # 읍면동별 평균 거래 가격
      등      면      읍
14724.22 12624.81 14888.65
> tapply(apt[, 'apt_price'], apt[, 'urban'], mean) # 읍면동별 평균 거래 가격
      등      면      읍
14724.22 12624.81 14888.65
```



## 연습문제 03

- 전용면적(area\_m2)이 30m2이하이면서 아파트 가격(apt\_price)이 2000만원 이하로 거래된 자료 행 번호를 which()로 추출하였을 때, 총 몇개의 아파트 거래건수가 있는 지 적으시오.
  - 사용할 데이터(객체)는 다음과 같음
    - `apt <- read.table('데이터_아파트매매가격.csv', sep=";", header = TRUE, stringsAsFactors = TRUE)`

## 연습문제 03

- 전용면적(area\_m2)이 30m2이하이면서 아파트 가격(apt\_price)이 2000만원 이하로 거래된 자료 행 번호를 which()로 추출하였을 때, 총 몇개의 아파트 거래건수가 있는 지 적으시오.
  - 사용할 데이터(객체)는 다음과 같음
  - `apt <- read.table('데이터_아파트매매가격.csv', sep=";", header = TRUE, stringsAsFactors = TRUE)`

## 연습문제 04

- 시군구(address\_sigungu)별 평균 아파트 가격을 알기 위하여 `tapply()`로 사용하여 구하시오. 이 때 평균 아파트 거래가격(`apt_price`)이 가장 높은 시군구를 적으시오. `cjdwnt`

## 데이터 정렬, 병합, 변환, 추출하기

- 대용량 데이터를 다룰 때, 이들 데이터의 정렬, 병합, 추출, 재구조화는 분석 단계 이전에서 중요한 역할을 함
  - 데이터 구조의 병합(merge), 정렬(sort), 재구조화(reshape)의 실습을 하는 것은 중요
- 데이터 정렬:
  - `sort()` : 데이터 프레임에는 적용 안됨(벡터만 가능)
  - `order()`
    - 데이터의 특정 변수들의 값들을 정렬(오름차순 및 내림차순)하여 데이터 탐구 및 추출을 용이하기 하는 절차적 변환
    - 사용법: `order(..., na.last = TRUE, decreasing = FALSE, method = c("auto", "shell", "radix"))`
      - 정렬 방법: 오름차순 정렬과 내림차순 정렬

## 데이터 정렬, 병합, 변환, 추출하기

- 정렬: sort()
  - 벡터만 가능

```
> x <- apt$apt_price
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1250    7700   13500   14553   19500   108000
> head(sort(x)) # 올림차순 정렬
[1] 1250 1250 1250 1250 1300 1300
> head(sort(x, decreasing=TRUE)) # 내림차순 정렬
[1] 108000 105000  97300  68000  67000  64500
```

- 정렬: order()
  - apt[order(apt\$apt\_price) , 1: 3]

## 데이터 정렬, 병합, 변환, 추출하기

- 정렬: order()

```
> ?order
> x <- apt[order(apt$apt_price) # apt_price에 대하여 오름차순 정렬
+           , 1:3] # 1:3 열만 추출
> head(x)
      id apt_price area_m2
511  511      1250  39.950
512  512      1250  39.950
513  513      1250  39.950
514  514      1250  39.950
189  189      1300  50.185
2637 2637      1300  42.540
> x <- apt[order(-apt$apt_price) # id에 대하여 내림차순 정렬
+           , 1:3] # 1:3 열만 추출
> head(x)
      id apt_price area_m2
10349 10349    108000 196.976
10323 10323    105000 196.976
10316 10316     97300 196.976
10299 10299     68000 162.626
9156   9156     67000 131.941
10361 10361     64500 152.652
> x <- apt[order(-apt$area_m2, -apt$apt_price) # area_m2와 apt_price에 대하여 내림차순 정렬
+           , 1:5] # 첫번째 열부터 다섯번째 열까지만
> head(x)
      id apt_price area_m2 floor_no year_built
12280 12280     31000 244.0710       18      2007
10745 10745     46700 219.0313       14      2005
11094 11094     25600 203.1600        6      1995
10349 10349    108000 196.9760       41      2010
10323 10323    105000 196.9760       42      2010
10316 10316     97300 196.9760       42      2010
```

## 데이터 정렬, 병합, 변환, 추출하기

- 데이터 구조 변환: reshape
  - from Wide to Long
  - Data from long to wide

**Wide**



**Long**

id	gender	race	trt	day1	day2	day3
1	F	0	0	19.81310	18.05777	14.84996
2	M	0	0	17.91846	18.75825	15.30547
3	M	0	0	17.22526	19.79218	15.10622

id	gender	race	trt	day	amt
1	F	0	0	day1	19.81310
1	F	0	0	day2	18.05777
1	F	0	0	day3	14.84996
2	M	0	0	day1	17.91846
2	M	0	0	day2	18.75825
2	M	0	0	day3	15.30547
3	M	0	0	day1	17.22526
3	M	0	0	day2	19.79218
3	M	0	0	day3	15.10622

## 데이터 정렬, 병합, 변환, 추출하기

- 데이터 구조 변환: reshape()

- Wide → Long

```
reshape(wide, direction = "long")
```

- Long → Wide

```
reshape(wide, idvar = "Subject", varying = list(2:12),  
        v.names = "conc", direction = "long")
```

```
reshape(df, timevar = "visit", idvar = "id", direction = "wide")
```

- 인수:

- direction = "wide" 또는 "long"
  - idvar = long 유형에서 동일 집단의 다수 행(레코드)을 가진 변수(들)
  - varying = 하나의 새로운 변수로 통합할 변수들 목록
  - v.names = long 유형으로 변환하였을(할) 때의 새로운 변수명
  - times = 숫자대신에 통합되어질 고유 변수명 사용
  - timevar = 새로이 만들어진 변수인 time의 이름 부여



## 데이터 정렬, 병합, 변환, 추출하기

- 데이터 구조 변환: reshape()
  - 예: wide to long # 매매와 전세 가격을 하나의 열로 재구조화
    - Long <- reshape(d, varying=c("apt\_sale\_price", "apt\_junse\_price"), v.names="price", timevar="type", times=c("apt\_sale\_price", "apt\_junse\_price"), direction="long")
    - str(Long)
    - head(Long)
    - View(Long)

```
> Long <- reshape(d, # 재구조화하고자 하는 데이터
+                 varying=c("apt_sale_price", "apt_junse_price"), # 하나의 변수로 통합할 변수
+                 v.names="price", # 새로운 변수명
+                 timevar="type", # 통합되어 만들어진 새로운 변수 이름 부여
+                 times=c("apt_sale_price", "apt_junse_price"), # 숫자 대신 새로운 변수(
+                 direction="long") # 재구조화 방향
> str(Long)
'data.frame': 522 obs. of 3 variables:
 $ id : num 1 2 3 4 5 6 7 8 9 10 ...
 $ type : chr "apt_sale_price" "apt_sale_price" "apt_sale_price" "apt_sale_price"
 $ price: num 102 104 105 105 104 ...
- attr(*, "reshapeLong")=List of 4
 ..$ varying:List of 1
 .. ..$ price: chr "apt_sale_price" "apt_junse_price"
 .. ..- attr(*, "v.names")= chr "price"
 .. ..- attr(*, "times")= chr "apt_sale_price" "apt_junse_price"
 ..$ v.names: chr "price"
 ..$ idvar : chr "id"
 ..$ timevar: chr "type"
> head(Long)
      id      type price
1. apt_sale_price 1 apt_sale_price 102.4
2. apt_sale_price 2 apt_sale_price 103.6
3. apt_sale_price 3 apt_sale_price 105.0
4. apt_sale_price 4 apt_sale_price 105.2
5. apt_sale_price 5 apt_sale_price 103.9
6. apt_sale_price 6 apt_sale_price 102.1
```

## 데이터 정렬, 병합, 변환, 추출하기

- 부분 데이터 또는 특정 조건에 맞는 데이터 추출: subset()
  - 벡터, 행렬, 데이터 프레임에서 부분 데이터 또는 특정 조건을 충족하는 하위데이터 추출할 때 유용하게 사용되는 함수(명령문)
  - 하위(부분) 데이터 추출

```
> sub_d <- subset(apt, # 추출할 데이터(색체)
+                 select = id: urban) # id열부터 urban 열까지만 데이터 선택
> str(sub_d)
'data.frame':  13309 obs. of  8 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price : int  4800 4500 4000 4000 4000 4000 4000 4000 4000 4800 ..
 $ area_m2   : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 ..
 $ floor_no  : int  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built: int  1998 1998 1998 1998 1998 1998 1998 1998 1998 1998 ..
 $ ym_sale   : int  201811 201811 201812 201812 201812 201812 201812 201812 201
 $ day_sale  : int   6 13 10 10 10 10 10 10 10 10 ...
 $ urban     : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
```

## 데이터 정렬, 병합, 변환, 추출하기

- 부분 데이터 또는 특정 조건에 맞는 데이터 추출: subset()
  - 조건 및 하위(부분) 데이터 추출

```
> d2 <- subset(apt, # 추출할 데이터(객체)
+               urban=="동", # 조건: urban이 "동"인 데이터 # 추출할 열(변수) 목록
+               select = c("urban", "apt_price", "area_m2"),)
> str(d2)
'data.frame':  8848 obs. of  3 variables:
 $ urban      : Factor w/ 3 levels "동","면","읍": 1 1 1 1 1 1 1 1 1 1 ...
 $ apt_price  : int  22500 22000 21500 22500 21300 21500 21500 15500 18900 2:
 $ area_m2    : num  84.4 84.4 84.4 84.4 84.4 ...
> table(d2$urban) # 빈도

동   면   읍
8848   0   0
> a <- subset(apt,
+             floor_no >20 & ym_sale <= 201812, # 층수가 20층 초과이면서 거래년월이 201812보다 이하
+             select = c(-day_sale: -address_dong)) # day_sale 열부터 address_dong 열 제외하고 선택
> str(a)
'data.frame':  309 obs. of  6 variables:
 $ id         : int  1244 1246 1297 1299 2318 2319 2320 2322 4183 5189 ...
 $ apt_price  : int  19800 16600 15400 16000 24200 17800 16900 17000 31710 26000 ...
 $ area_m2    : num  84.5 74.4 59.9 59.9 78.4 ...
 $ floor_no   : int   22 23 21 21 26 32 34 21 22 23 ...
 $ year_built : int  2014 2014 2014 2014 2017 2017 2017 2017 2018 2013 ...
 $ ym_sale    : int  201810 201811 201809 201810 201809 201809 201809 201811 201812 201810 ...
```

# 데이터 마이닝 기초

## 데이터 정렬, 병합, 변환, 추출하기

- 좌우 데이터(프레임) 병합: merge()
  - 사용법: merge(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all, sort = TRUE, suffixes = c(".x", ".y"), incomparables = NULL, ...)

```
> str(apt) # 아파트 거래 자료의 시도, 시군구
'data.frame': 13309 obs. of 14 variables:
 $ id          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price   : int  4800 4500 4000 4000 4000 4000 4000 4000
 $ area_m2     : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8
 $ floor_no    : int  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built  : int  1998 1998 1998 1998 1998 1998 1998 1998
 $ ym_sale     : int  201811 201811 201812 201812 201812 20181
 $ day_sale    : int  6 13 10 10 10 10 10 10 10 10 ...
 $ urban       : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3
 $ apt_complex : Factor w/ 802 levels "(1028-0)","(142-1)",...
 $ address     : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사
 $ address_bunji : Factor w/ 873 levels "1","1-1","1-16",...: 216
 $ address_sido : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1
 $ address_sigungu : Factor w/ 14 levels "괴산군","단양군",...: 1 1 1
 $ address_dong : Factor w/ 107 levels "가경동","감곡면",...: 9 9
```

```
> str(data_sigungu) # 시군구 통계자료의 시도, 시군구
'data.frame': 261 obs. of 16 variables:
 $ id          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ sido        : Factor w/ 17 levels "강원도","경기도",...: 9 9 9 9 9 9 9 9 9
 $ sigungu     : Factor w/ 236 levels "가평군","강남구",...: 190 191 161 110 2
 $ area        : num  23.91 9.96 21.87 16.86 17.06 ...
 $ pop_density : num  6387 12574 10529 17750 20935 ...
 $ pop_tot     : num  154770 125709 229161 304808 357703 ...
 $ pop_male    : num  75967 62253 110878 150368 174414 ...
 $ pop_female  : num  78803 63456 118283 154440 183289 ...
 $ age_average : Factor w/ 132 levels "-", "35.9", "36.2",...: 48 54 48 40 36
 $ r_aged      : num  17.1 17.1 16 13.6 12.5 ...
 $ pop_net_move : num  -204 143 -89 57 -608 -469 -429 -716 -521 -308 ...
 $ housing     : num  45729 40177 71680 87830 85638 ...
 $ apt_sale_price : num  102 104 105 105 104 ...
 $ apt_junse_price : num  102 101 100 100 102 ...
 $ car         : num  0.33 0.41 0.34 0.34 0.28 0.28 0.28 0.27 0.24 0.29 ...
 $ accident_per_1000car : num  18.82 18.72 14.64 8.54 8.72 ...
```

## 데이터 정렬, 병합, 변환, 추출하기

- 좌우 데이터(프레임) 병합:  
merge()

- 아파트 거래자료(x)와 시군구 통계 자료(y) 좌우 병합

- 병합 기준 id

- 매칭된 두 데이터만 병합

```
> m1 <- merge(apt, data_sigungu, # merge할 두 데이터 프레임(x, y)
+             by.x = c("address_sido", "address_sigungu"), # x데이터의 병합 기준 id
+             by.y = c("sido", "sigungu")) # y 객체의 병합 기준 id
> str(m1)
'data.frame': 5490 obs. of 28 variables:
 $ address_sido      : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu   : Factor w/ 14 levels "괴산군","단양군",...: 1 1 1 1 1 1 1 1 1 1 .
 $ id.x              : int  1 7 8 9 10 11 12 13 14 2 ...
 $ apt_price         : int  4800 4000 4000 4000 4800 4000 4000 4000 6300 4500 ...
 $ area_m2           : num  39.8 39.8 39.8 39.8 39.8 39.8 ...
 $ floor_no          : int  2 5 9 8 5 10 1 1 5 6 ...
 $ year_built        : int  1998 1998 1998 1998 1998 1998 1998 1998 1985 1998 ...
 $ ym_sale           : int  201811 201812 201812 201812 201901 201901 201901 201901
 $ day_sale          : int  6 10 10 10 10 10 11 22 22 18 13 ...
 $ urban             : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
 $ apt_complex       : Factor w/ 802 levels "(1028-0)","(142-1)",...: 167 167 167 167
 $ address           : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사리",...: 1 1 1 1
 $ address_bunji     : Factor w/ 873 levels "1","1-1","1-16",...: 216 216 216 216 216
 $ address_dong      : Factor w/ 107 levels "가경동","감곡면",...: 9 9 9 9 9 9 9 9 9 9
 $ id.y              : num  153 153 153 153 153 153 153 153 153 153 ...
 $ area              : num  407 407 407 407 407 ...
 $ pop_density       : num  172 172 172 172 172 ...
 $ pop_tot           : num  73677 73677 73677 73677 73677 ...
 $ pop_male          : num  38314 38314 38314 38314 38314 ...
 $ pop_female        : num  35363 35363 35363 35363 35363 ...
 $ age_average       : Factor w/ 132 levels "-", "35.9", "36.2",...: 109 109 109 109 10
 $ r_aged            : num  31.1 31.1 31.1 31.1 31.1 ...
 $ pop_net_move      : num  391 391 391 391 391 391 391 391 391 391 ...
 $ housing           : num  16380 16380 16380 16380 16380 ...
 $ apt_sale_price    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ apt_junse_price   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ car               : num  0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 ...
 $ accident_per_1000car: num  9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 ...
> View(m1)
```

## 데이터 정렬, 병합, 변환, 추출하기

- 좌우 데이터(프레임) 병합: merge()
  - 아파트 거래자료(x)와 시군구 통계 자료(y) 좌우 병합
    - 병합 기준 id
  - all.x = TRUE
    - 매칭되지 않아도 x 객체 모두 남겨둠
  - all.y = TRUE
    - 매칭되지 않아도 y 객체 모두 남겨둠

```
> m2 <- merge(apt, data_sigungu,
+             by.x = c("address_sido", "address_sigungu"),
+             by.y = c("sido", "sigungu"),
+             all.x=TRUE) # x데이터 프레임은 매칭 되지 않아도 모두 남겨둠
> str(m2)
'data.frame': 13309 obs. of 28 variables:
 $ address_sido      : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu   : Factor w/ 14 levels "괴산군","단양군",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ id.x              : int  1 7 8 9 10 11 12 13 14 2 ...
 $ apt_price         : int  4800 4000 4000 4000 4800 4000 4000 4000 6300 4500 ...
 $ area_m2           : num  39.8 39.8 39.8 39.8 39.8 39.8 ...
 $ floor_no          : int  2 5 9 8 5 10 1 1 5 6 ...
 $ year_built        : int  1998 1998 1998 1998 1998 1998 1998 1998 1998 1985 1998 ...
 $ ym_sale           : int  201811 201812 201812 201812 201901 201901 201901 2019 ...
 $ day_sale          : int  6 10 10 10 10 11 22 22 18 13 ...
 $ urban             : Factor w/ 3 levels "등","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
 $ apt_complex       : Factor w/ 802 levels "(1028-0)","(142-1)",...: 167 167 167 ...
 $ address           : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사리",...: 1 1 1 ...
 $ address_bunji     : Factor w/ 873 levels "1","1-1","1-16",...: 216 216 216 216 ...
 $ address_dong       : Factor w/ 107 levels "가경동","감곡면",...: 9 9 9 9 9 9 9 9 9 ...
 $ id.y              : num  153 153 153 153 153 153 153 153 153 153 ...
 $ area              : num  407 407 407 407 407 ...
 $ pop_density       : num  172 172 172 172 172 ...
 $ pop_tot           : num  73677 73677 73677 73677 73677 ...
 $ pop_male          : num  38314 38314 38314 38314 38314 ...
 $ pop_female        : num  35363 35363 35363 35363 35363 ...
 $ age_average       : Factor w/ 132 levels "-", "35.9", "36.2",...: 109 109 109 109 ...
 $ r_aged            : num  31.1 31.1 31.1 31.1 31.1 ...
 $ pop_net_move      : num  391 391 391 391 391 391 391 391 391 391 ...
 $ housing           : num  16380 16380 16380 16380 16380 ...
 $ apt_sale_price    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ apt_junse_price   : num  NA NA NA NA NA NA NA NA NA NA ...
 $ car               : num  0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 0.57 ...
 $ accident_per_1000car: num  9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 9.49 ...
> view(m2)
```



## 데이터 정렬, 병합, 변환, 추출하기

- 상하 데이터(프레임) 병합(붙이기): `rbind()`

- Combine R Objects by Rows

- 데이터 구조가 동일할 때 주로 사용

- 데이터 구조가 다를 경우
  - 일치하는 데이터만 추출하여 실행

- 예: 아파트 실거래가 자료

- 충청북도 자료(x)
- 충청남도, 세종시, 대전시 자료(y)
- `rbind(x, y)`

```
> apt_cb <- apt # 충청북도 아파트 거래 사료
> str(apt_cb)
'data.frame': 13309 obs. of 14 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price : int  4800 4500 4000 4000 4000 4000 4000 4000 4000 4800 ...
 $ area_m2  : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 ...
 $ floor_no  : int  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built : int  1998 1998 1998 1998 1998 1998 1998 1998 1998 1998 ...
 $ ym_sale   : int  201811 201811 201812 201812 201812 201812 201812 201812 2018 ...
 $ day_sale  : int  6 13 10 10 10 10 10 10 10 10 ...
 $ urban     : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
 $ apt_complex : Factor w/ 802 levels "(1028-0)","(142-1)",...: 167 167 167 ...
 $ address   : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사리",...: 1 1 1 ...
 $ address_bunji : Factor w/ 873 levels "1","1-1","1-16",...: 216 216 216 216 ...
 $ address_sido : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu : Factor w/ 14 levels "괴산군","단양군",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ address_dong : Factor w/ 107 levels "가경동","감곡면",...: 9 9 9 9 9 9 9 9 9 9
```

```
> apt_cn <- read.table('데이터_아파트(매매)_실거래가_충남대전세종.csv', # 붙이고자 하는 데이터 파일명
+                      sep=";", # 구분자 = ";"
+                      header = TRUE, # 첫행을 변수명으로
+                      stringsAsFactors = TRUE) # 명목변수(열)은 요인으로 불러오기
```

```
> str(apt_cn)
'data.frame': 44109 obs. of 13 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price : int  14700 17000 15900 15500 16700 16900 16200 11000 17950 16000 ...
 $ area_m2  : num  76 85 76 85 85 ...
 $ floor_no  : int  2 11 7 10 3 9 8 1 6 5 ...
 $ year_built : int  2002 2002 2002 2002 2002 2002 2002 2002 2002 2002 ...
 $ ym_sale   : int  201809 201809 201810 201810 201810 201811 201811 201812 201901 2019 ...
 $ day_sale  : int  19 19 6 8 10 13 20 4 31 9 ...
 $ urban     : Factor w/ 3 levels "동","면","읍": 1 1 1 1 1 1 1 1 1 1 ...
 $ apt_complex : Factor w/ 1729 levels "(296-1)","(335-118)",...: 110 110 110 110 110 110 ...
 $ address_bunji : Factor w/ 1625 levels "1","1-1","1-1113",...: 1352 1352 1352 1352 1352 13 ...
 $ address_sido : Factor w/ 3 levels "대전광역시","세종특별자치시",...: 3 3 3 3 3 3 3 3 3 3 ...
 $ address_sigungu : Factor w/ 48 levels "계룡시","고운동",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ address_dong : Factor w/ 247 levels "", "가수원동",...: 31 31 31 31 31 31 31 31 31 31 ...
```

## 데이터 정렬, 병합, 변환, 추출하기

- 상하 데이터(프레임) 병합(붙이기): rbind()

- Combine R Objects by Rows

- 아파트 실거래가 자료

- 충청북도 자료(apt\_cb2)

- 일치하지 않은 열 제외한 데이터 셋 추출

- 충청남도, 세종시, 대전시 자료(apt\_cn)

- rbind(apt\_cb2, apt\_cn)

```
> apt_cb2 <- subset(apt_cb, select = -address) # address 열만 제외한 데이터 셋 추출 후 할당
> apt_cncb <- rbind(apt_cb2, apt_cn)
> str(apt_cncb)
'data.frame': 57418 obs. of 13 variables:
 $ id          : int  1 2 3 4 5 6 7 8 9 10 ...
 $ apt_price   : int  4800 4500 4000 4000 4000 4000 4000 4000 4000 4800 ...
 $ area_m2     : num  39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 39.8 ...
 $ floor_no    : int  2 6 7 9 3 7 5 9 8 5 ...
 $ year_built  : int  1998 1998 1998 1998 1998 1998 1998 1998 1998 1998 ...
 $ ym_sale     : int  201811 201811 201812 201812 201812 201812 201812 201812 201812 201812 ...
 $ day_sale    : int  6 13 10 10 10 10 10 10 10 10 ...
 $ urban       : Factor w/ 3 levels "동","면","읍": 3 3 3 3 3 3 3 3 3 3 ...
 $ apt_complex : Factor w/ 2436 levels "(1028-0)","(142-1)",...: 167 167 167 167 167 1 ...
 $ address_bunji : Factor w/ 2270 levels "1","1-1","1-16",...: 216 216 216 216 216 216 2 ...
 $ address_sido : Factor w/ 4 levels "충청북도","대전광역시",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu : Factor w/ 57 levels "괴산군","단양군",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ address_dong  : Factor w/ 348 levels "가경동","감곡면",...: 9 9 9 9 9 9 9 9 9 9 ...
```



## 데이터 정렬, 병합, 변환, 추출하기

- 알아두면 유용한 함수들: (...), with, attach, detach
  - (...)
    - 문장 시작("(")과 끝(")")의 괄호는 할당 결과를 바로 실행하여 보여 줌
    - (...)의 있고 없음의 차이
      - `k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 )`
      - `(k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 ))`

```
> k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 )
> k
[1] 473 972 973 976 983 987 988 989 2907
> (k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 ))
[1] 473 972 973 976 983 987 988 989 2907
```

## 데이터 정렬, 병합, 변환, 추출하기

- 알아두면 유용한 함수들: (...) with, attach, detach

- (...)

- 문장 시작("(")과 끝(")")의 괄호는 할당 결과를 바로 실행하여 보여 줌

- (...)의 있고 없음의 차이

- `k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 )`

- `(k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 ))`

- with()

```
> k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 )
> k
[1] 473 972 973 976 983 987 988 989 2907
> (k <- which(apt$apt_price <= 2000 & apt$area_m2 <=30 ))
[1] 473 972 973 976 983 987 988 989 2907
```

- attach()

- detach()

## 데이터 정렬, 병합, 변환, 추출하기

- 알아두면 유용한 함수들: (...) with, attach, detach

- with()

- with(data, expression, .....)

- with 함수 내("( )")의 키(\$) 사용을 간결하게 하기 위한 함수

- With 함수가 있고 없음의 차이 확인

- 데이터(객체) 명 한번만 사용하고, \$ 사용 불필요

- with()로 더욱 간편하게 접근함. 그러나 해당 명령에서만 유효

```
> ?with()
> a1 <- apt_cncb$apt_price[apt_cncb$floor_no >20 & apt_cncb$area_m2 <60]
> a2 <- with(apt_cncb, apt_price[floor_no >20 & area_m2 <60]) # with()로 더욱
```

## 기타 유용한 함수들: (...) with, attach, detach

- 기타 알아두면 유용한 함수들: (...) with, attach, detach
  - attach(): Attach Set of R Objects to Search Path
    - 데이터(객체)를 attach()로 설정하면, "객체\$"를 사용하지 않고 바로 접근(사용) 가능
      - with()함수는 해당 함수 내에서만 작동
      - attach()함수는 detach()함수를 적용하기 이전에는 계속 사용 가능
        - with()함수 보다 편리하게 데이터 내 접근하여 사용

```
> ?attach # Attach Set of R Objects to Search Path
> summary(mtcars$mpg) # 내장 데이터에 있는 mpg파일 요약통계
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
10.40  15.43   19.20   20.09   22.80   33.90
> attach(mtcars) # "mtcars" 객체(데이터) 바로 접근 적용
> summary(mpg) # "mtcars$" 없이 데이터 접근
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
10.40  15.43   19.20   20.09   22.80   33.90
> detach() # 해제
> summary(mpg) # 해제 후에는 접근 안됨
Error in summary(mpg) : object 'mpg' not found
```

## 연습문제 05

- 아파트 전용면적(area\_m2)이 100m2 이하이면서, 그 가격(apt\_price)이 2억이하로 거래된 자료만 추출하되, apt\_complex 열 제외하고 선택하였을 때, 이 때 관측치(행)의 갯수와 열(변수)의 갯수를 적으시오. 이 때 사용할 함수는 subset()임
  - 사용할 데이터(객체)는 다음과 같음
    - apt <- read.table('데이터\_아파트매매가격.csv', sep=";", header = TRUE, stringsAsFactors = TRUE)

## 연습문제 06

- 데이터(mtcars)의 hp 변수의 요약통계를 살펴보고자 한다. 올바르게 사용한 명령문을 모두 고르시오.
  - ① `summary(hp)`
  - ② `summary(mtcars$ hp)`
  - ③ `with(mtcars, hp)`
  - ④ `attach(mtcars) ; summary(mtcars$ hp)`
  - ⑤ `attach(mtcars) ; summary(hp)`
  - ⑥ `detach(mtcars) ; summary(hp)`

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- 기존 데이터 가공 방법들 검토
  - 데이터프레임의 일부를 추출하기 위한 `subset()` 함수와 `[` 과 `$` 운영자의 이용 가능
  - 그러나 이들 함수와 운영자들은 추출, 재정렬, 집계화 등의 작업에서의 쉽지 않음
    - tedious operations
    - not very intuitive
- The dplyr package
  - designed to mitigate a lot of these problems and to provide a highly optimized set of routines specifically for dealing with data frames

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- The dplyr Package
  - Not provide any "new" functionality to R per se
  - could already be done with base R
  - but it greatly simplifies existing functionality in R.

```
# install dplyr package
install.packages("tidyverse")
install.packages("dplyr")
```

```
library(dplyr)
?dplyr
```

dplyr-package {dplyr}

R Documentation

### dplyr: a grammar of data manipulation

#### Description

dplyr provides a flexible grammar of data manipulation. It's the next iteration of plyr, focused on tools for working with data frames (hence the *d* in the name).



# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- dplyr package의 장점
  - a **flexible** grammar of **data manipulation**
  - focused on tools for working with **data frames** (hence the *d* in the name).
  - three main goals
    - Identify the most important data manipulation **verbs** and make them **easy to use** from R.
      - `select()`, `filter()`, `arrange()`, `rename()`, `mutate()`, `summarise()/summarize()`, `%>%`
    - Provide blazing **fast performance** for in-memory data by writing key pieces in C++ (using Rcpp)
    - Use the same interface to work with data no matter where it's stored, whether in **a data frame, a data table or database**
      - **relational databases that allow you to operate on very very large data frames**

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- dplyr package의 특징
  - Hadley Wickham가 작성한 데이터 처리에 특화된 R 패키지
    - **plyr**도 편리하긴 했지만 모든 함수가 R로 작성되어서 처리 속도가 느림
    - **dplyr**은 C++로 작성되어 불필요한 함수를 불러오지 않기 때문에 처리 속도 빠름
  - 다음 형식의 데이터 사용가능
    - data.table : data.table 패키지와 사용
    - 각종 데이터베이스 : 현재 MySQL, PostgreSQL, SQLite, BigQuery를 지원
    - 데이터 큐브 : dplyr 패키지 내부에 실험적으로 내장됨

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- dplyr package 함수와 유사 R 함수 비교

함수명	내용	유사함수
filter()	지정한 조건식에 맞는 데이터 추출	subset()
select()	열의 추출	data[, c("Year", "Month")]
mutate()	열 추가	transform()
arrange()	정렬	order(), sort()
summarise()	집계	aggregate()

출처: <https://wsyang.com/2014/02/introduction-to-dplyr/>

- group\_by() 함수
  - 집단별로 다양한 집계를 추가로 활용 가능

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- The key “**verbs**” provided by the dplyr package
  - `select()`
    - return a subset of the **columns of a data frame**, using a flexible notation
  - `filter()`
    - extract **a subset of rows** from a data frame based on logical conditions
  - `arrange()`
    - **reorder** rows of a data frame
  - `rename()`
    - **rename** variables in a data frame
  - `mutate()`, `transmute()`
    - **add new variables/columns** or transform existing variables
  - `summarise()` 또는 `summarize()`
    - **generate summary statistics of different variables** in the data frame, possibly within strata
  - `%>%`
    - the “pipe” operator is used to **connect multiple verb actions together into a pipeline**
    - 파이프(pipe) 연산자를 사용해서 dplyr 동사를 순열로 조합하여 연결

# dplyr 패키지를 활용한 데이터 마이닝

## dplyr 패키지 소개

- 공통으로 적용되는 dplyr 함수 특징들
  - 첫번째 인수는 데이터 프레임(data frame)
  - 이후의 일련의 인수들은 해당 데이터로 해야 할 작업 설정과 키(\$) 없이 열(변수)에 접근할 수 있음
  - 함수 실행의 반환 결과는 새로운 데이터 프레임
  - 데이터 프레임은 한 줄당 한 개의 관측치로 구성되어야 함
  - Data frames must be properly formatted and annotated for this to all be useful.
    - In particular, the data must be tidy.
    - In short, there should be **one observation per row**, and each column should represent a feature or characteristic of that observation.

# dplyr 패키지를 활용한 데이터 마이닝

## 추출하기: select()

- 열(변수)별로 지정한 열(변수)들을 선택하는 함수
- 유용한 기능들
  - X:y : x열 부터 y열까지만 선택하여 추출
  - c(x, y:z) : x와 y열부터 z열만 선택하여 추출
  - -x : x열만 제외하고 모두 선택
  - -x : -y : x열부터 y열까지 제외하고 모두 선택
  - select() 에서만 사용되는 특별 기능들
    - [starts\\_with\(\)](#), [ends\\_with\(\)](#), [contains\(\)](#)
    - [matches\(\)](#)
    - [num\\_range\(\)](#)
    - [one\\_of\(\)](#)
    - [everything\(\)](#)
    - [group\\_cols\(\)](#)

# dplyr 패키지를 활용한 데이터 마이닝

## 추출하기: select()

### • 실습

```
> sub_apt <- select(apt, # 수출할 객체명
+                   floor_no, year_built, ym_sale) # 추출할 열 이름(들)
> head(sub_apt)
  floor_no year_built ym_sale
1         2      1998  201811
2         6      1998  201811
3         7      1998  201812
4         9      1998  201812
5         3      1998  201812
6         7      1998  201812

> sub_apt <- select(apt, # 추출할 객체명
+                   apt_price:urban) # apt_price열부터 urban열까지 선택
> head(sub_apt)
  apt_price area_m2 floor_no year_built ym_sale day_sale urban
1     4800    39.8         2      1998  201811         6     음
2     4500    39.8         6      1998  201811        13     음
3     4000    39.8         7      1998  201812        10     음
4     4000    39.8         9      1998  201812        10     음
5     4000    39.8         3      1998  201812        10     음
6     4000    39.8         7      1998  201812        10     음
```

```
> sub_apt <- select(apt,
+                   -(apt_price:urban)) # apt_price열부터 urban열까지 제외하고 모두 선택
> head(sub_apt)
  id apt_complex address address_bunji address_sido address_dong
1  1 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군
2  2 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군
3  3 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군
4  4 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군
5  5 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군
6  6 두진백로 충청북도 괴산군 괴산읍 대사리 177-1 충청북도 괴산군

> sub_apt <- select(apt,
+                   ends_with("e")) # e로 끝나는 이름을 가진 열들만 선택
> str(sub_apt)
'data.frame': 13309 obs. of 3 variables:
 $ apt_price: int  4800 4500 4000 4000 4000 4000 4000 4000 4000 4800 ...
 $ ym_sale : int  201811 201811 201812 201812 201812 201812 201812 201812 201812 201812 ...
 $ day_sale: int   6 13 10 10 10 10 10 10 10 10 ...

> sub_apt <- select(apt,
+                   starts_with("address")) # address로 시작하는 이름을 가진 열들만 선택
> str(sub_apt)
'data.frame': 13309 obs. of 5 variables:
 $ address      : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사리",...: 1 1 1 1 1
 $ address_bunji: Factor w/ 873 levels "1","1-1","1-16",...: 216 216 216 216 216
 $ address_sido : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu: Factor w/ 14 levels "괴산군","단양군",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ address_dong : Factor w/ 107 levels "가경동","감곡면",...: 9 9 9 9 9 9 9 9 9 9 ..
```

dplyr 패키지를 활용한 데이터 마이닝

# 특정 조건에 맞는 데이터 추출하기: filter()

- Return rows with matching conditions
  - choose rows/cases where conditions are true.
  - similar to the existing subset() function in R but is quite a bit faster
- **Useful filter functions**
  - ==, >, >= etc
  - &, |, !, xor()
  - is.na()
  - between(), near()



# dplyr 패키지를 활용한 데이터 마이닝

## I 특정 조건에 맞는 데이터 추출하기: filter()

- 실습

```
> ?filter # Return rows with matching conditions
> apt.f <- filter(apt,
+               apt_price >= 10000) # 1억이상 아파트 가격에 거래된 자료들만 추출
> summary(apt.f$apt_price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
10000   13950   17500   19261   23200   108000
> str(apt.f)
'data.frame':   8493 obs. of  14 variables:
 $ id      : int  17 18 19 20 21 22 23 24 25 26 ...
 $ apt_price : int  16000 15300 12600 15500 16100 15500 15300 21000 21800
 $ area_m2  : num  60 60 60 60 60 ...

> apt.f <- filter(apt,
+               apt_price >= 10000 & urban == "읍") # 1억이상 아파트 가격에 거래되면서 읍지역에 해당하는 자료들만 추출
> table(apt.f$urban) # 빈도

   등   면   읍
   0    0 1968
> str(apt.f)
'data.frame':   1968 obs. of  14 variables:
 $ id      : int  17 18 19 20 21 22 23 24 25 26 ...
 $ apt_price : int  16000 15300 12600 15500 16100 15500 15300 21000 21800 21300 ...

> apt.f <- filter(apt,
+               apt_price >= 10000 | urban == "등") # 1억이상 아파트 가격이거나 등지역에 해당하는 자료들만 추출
> table(apt.f$urban) # 빈도

   등   면   읍
8848  784 1968
> str(apt.f)
'data.frame':   11600 obs. of  14 variables:
 $ id      : int  17 18 19 20 21 22 23 24 25 26 ...
 $ apt_price : int  16000 15300 12600 15500 16100 15500 15300 21000 21800 21300 ...
 $ area_m2  : num  60 60 60 60 60 ...
```

## 변수명 변환하기: rename()

- rename() 변수명 변환하기
  - 모든 변수들은 변하지 않음
  - R에서 변수명 변환하기는 어려운 작업임
  - rename() 함수는 변수명 변환을 쉽게 하기 위한 도구
- 실습

```
> ?rename() # select/rename variables by name
> head(apt[ , 2:5], 4) # 2번째 부터 5번째 열까지, 4개의 행 데이터셋 확인하기
  apt_price area_m2 floor_no year_built
1     4800    39.8        2     1998
2     4500    39.8        6     1998
3     4000    39.8        7     1998
4     4000    39.8        9     1998
> apt.r <- rename(apt, # 바꾸고자 하는 객체 명
+                  price = apt_price, area = area_m2, floor = floor_no, b_year = year_built) # 변수명 바꿈
> head(apt.r[ , 2:5], 4)
  price area floor b_year
1  4800  39.8     2   1998
2  4500  39.8     6   1998
3  4000  39.8     7   1998
4  4000  39.8     9   1998
```

# dplyr 패키지를 활용한 데이터 마이닝

## 정렬하기: arrange()

- Arrange rows by variables
  - reorder rows of a data frame according to one of the variables/columns
  - simplifies the process
- 실습

```
> ## arrange()
> ?arrange # order tbl rows by an expression involving its variables.
> apt <- arrange(apt, # 정렬할 데이터 명
+               apt_price, area_m2, year_built) # 오름차순 정렬
> head(select(apt, apt_price, area_m2, year_built), 3) # 확인1
  apt_price area_m2 year_built
1      1250   39.95      1993
2      1250   39.95      1993
3      1250   39.95      1993
> tail(select(apt, apt_price, area_m2, year_built), 3) # 확인2
  apt_price area_m2 year_built
13307     97300  196.976     2010
13308    105000  196.976     2010
13309    108000  196.976     2010
```

```
> ?desc # Descending order
> apt <- arrange(apt,
+               desc(apt_price, area_m2)) # 내림차순 정렬
> head(select(apt, apt_price, area_m2), 3)
  apt_price area_m2
1    108000  196.976
2    105000  196.976
3     97300  196.976
> tail(select(apt, apt_price, area_m2), 3)
  apt_price area_m2
13307     1250   39.95
13308     1250   39.95
13309     1250   39.95
```

# dplyr 패키지를 활용한 데이터 마이닝

## 데이터 수정 및 변형: mutate(), transmute()

- mutate()와 transmute()의 차이
  - Mutate()함수는 기존 데이터를 그대로 유지하면서 새로운 변수를 추가
  - Transmute() 함수는 기존 데이터를 제거하고 새로이 생성된 변수만 추가
  - 기존 변수명이 같을 경우에는 새로운 변수가 추가되면서 기존 변수에 덮어쓰기를 하게 되므로 주의
- mutate()와 transmute()에서 유용한 기능들
  - [+](#), [-](#), [log\(\)](#), etc., for their usual mathematical meanings
  - [lead\(\)](#), [lag\(\)](#)
  - [dense\\_rank\(\)](#), [min\\_rank\(\)](#), [percent\\_rank\(\)](#), [row\\_number\(\)](#), [cume\\_dist\(\)](#), [ntile\(\)](#)
  - [cumsum\(\)](#), [cummean\(\)](#), [cummin\(\)](#), [cummax\(\)](#), [cumany\(\)](#), [cumall\(\)](#)
  - [na\\_if\(\)](#), [coalesce\(\)](#)
  - [if\\_else\(\)](#), [recode\(\)](#), [case\\_when\(\)](#)

# dplyr 패키지를 활용한 데이터 마이닝

## 데이터 수정 및 변형: mutate(), transmute()

- 실습

- mutate()

```
> apt <- mutate(apt, # 추가할 객체 넣
+               mean_diff_price = apt_price - mean(apt_price, na.rm = TRUE)) # 아파트 평균 가격의 차이
> head(select(apt, apt_price, mean_diff_price), 3 )
  apt_price mean_diff_price
1   108000      93447.24
2   105000      90447.24
3    97300      82747.24
> summary(apt$apt_price) # 아파트 평균 요약통계
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1250   7700   13500   14553   19500   108000
> summary(apt$mean_diff_price)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-13303  -6853  -1053      0    4947   93447
```

- 93447.24 = 108000-14553

- transmute()

```
> apt.tr <- transmute(apt, # 추가할 객체 넣
+                     mean_diff_price = apt_price - mean(apt_price, na.rm = TRUE), # 아파트 평균 가격의 차이
+                     mean_diff_area = area_m2 - mean(area_m2, na.rm = TRUE)) # 아파트 평균 전용 면적의 차이
> head(apt.tr, 5) # 첫번째 부터 다섯 째 행까지만 데이터 보여주기
  mean_diff_price mean_diff_area
1      93447.24      128.05387
2      90447.24      128.05387
3      82747.24      128.05387
4      53447.24      93.70387
5      52447.24      63.01887
```

# 집단별 데이터 분할하기: group\_by()

- 변수(들)별 분할(집계화)
  - group\_by() takes an existing tbl and converts it into a grouped tbl where operations are performed "by group".
    - 집단별로 요약 통계를 생성할 때 사용
    - 복귀
      - ungroup() : removes grouping.

## • Usage

- group\_by(.data, ..., add = FALSE, .drop = FALSE)
- ungroup(x, ...)

```
.data  a tbl
...    Variables to group by. All tbls accept variable names. Some tbls will accept functions
        of variables. Duplicated groups will be silently dropped.

add    When add = FALSE, the default, group_by() will override existing groups. To add
        to the existing groups, use add = TRUE.

.drop  When .drop = TRUE, empty groups are dropped.

x      A tbl()
```

# dplyr 패키지를 활용한 데이터 마이닝

## 집단별 데이터 분할하기: group\_by()

- 실습
  - 데이터를 집단으로 분할: group\_by()
  - 건축년도별 데이터를 리스트로 분할

```
> years <- group_by(apt, # 변수의 집단별 분할할 객체명
+                   year_built) # 건축 연도별 매트릭스 형태의 리스트 형성
> str(years) #
Classes 'grouped_df', 'tbl_df', 'tbl' and 'data.frame': 13309 obs. of  15 v
 $ id          : int  10349 10323 10316 10299 9156 10361 10324 9147 1030
 $ apt_price    : int  108000 105000 97300 68000 67000 64500 64000 63000
 $ area_m2      : num  197 197 197 163 132 ...
 $ floor_no     : int  41 42 42 45 23 39 24 24 6 29 ...
 $ year_built   : int  2010 2010 2010 2010 2015 2010 2010 2015 2010 2018
 $ ym_sale      : int  201905 201902 201901 201810 201901 201907 201902 2
 $ day_sale     : int  25 13 13 18 10 19 20 28 3 11 ...
 $ urban        : Factor w/ 3 levels "동","면","읍": 1 1 1 1 1 1 1 1 1 1 .
 $ apt_complex  : Factor w/ 802 levels "(1028-0)","(142-1)",...: 373 373 3
 $ address      : Factor w/ 181 levels "충청북도 괴산군 괴산읍 대사리",...: 142
 $ address_bunji : Factor w/ 873 levels "1","1-1","1-16",...: 426 426 426 4
 $ address_sido  : Factor w/ 1 level "충청북도": 1 1 1 1 1 1 1 1 1 1 ...
 $ address_sigungu: Factor w/ 14 levels "괴산군","단양군",...: 13 13 13 13 12
 $ address_dong  : Factor w/ 107 levels "가경동","감곡면",...: 40 40 40 40 90
 $ mean_diff_price: num  93447 90447 82747 53447 52447 ...
- attr(*, "groups")=Classes 'tbl_df', 'tbl' and 'data.frame':  43 obs. of
 ..$ year_built: int  1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 ..
 ..$ .rows      :List of 43
 .. ..$ : int  7811 8584 8821
 .. ..$ : int  7922 9101 9127 9452 9630 9834 9857
> class(years)
[1] "grouped_df" "tbl_df"      "tbl"         "data.frame"
```

# 데이터 마이닝: %>% (파이프 연산자)

- R 프로그래밍에서의 작업
  - 프로그래밍(코딩)할 때, 복잡한 작업
    - 함수의 결과를 다시 다른 함수에 넣고 그 결과를 다시 함수에 넣는 작업
      - `third(second(first(x)))`
    - 수 많은 괄호 생성으로 수정 번거롭고, 이해하기 어려움 발생
- dplyr패키지에서의 pipe operator 사용으로 해결
  - 파이프 연산자는 괄호의 괄호 등보다 직관적이고 사용이 편리
- Dplyr패키지의 함수들을 "%>%"로 일련의 연속된 작업 수행
  - 왼쪽부터 순차적으로 실행하고 그 반환 결과를 다음의 함수로 실행
- 예:
  - 일반적인 작업 절차: `third(second(first(x)))`
  - %>% 작업 절차: `first(x) %>% second %>% third`



# 데이터 마이닝: %>% (파이프 연산자)

- 실습

- apt 데이터에서 apt\_price가 평균보다 큰 데이터만 추출

```
> a <- apt %>% filter(apt_price > mean(apt_price, na.rm = TRUE))
> str(a)
'data.frame': 5944 obs. of 15 variables:
 $ id      : int  10349 10323 10316 10299 9156 10361 10324
 $ apt_price : int  108000 105000 97300 68000 67000 64500 64
 $ area_m2   : num  197 197 197 163 132 ...
 $ floor_no  : int   41 42 42 45 23 39 24 24 6 29 ...
 $ year_built : int   2010 2010 2010 2010 2015 2010 2010 2015
```

- apt 데이터에서 apt\_price가 평균보다 2배 큰 데이터만 추출한 후 apt\_price부터 area\_m2 변수들만 선택하고, 이를 다시 area\_m2의 내림차순 정렬

```
> *** 1. apt 데이터에서
> *** 2. apt_price가 평균보다 2배 큰 데이터만 추출한 후
> *** 3. apt_price부터 area_m2 변수들만 선택하고,
> *** 4. 이를 다시 area_m2의 내림차순 정렬
> b <- apt %>% # 1
+   filter(apt_price > mean(apt_price, na.rm = TRUE)*2) %>% # 2
+   select(apt_price:area_m2) %>% # 3
+   arrange(desc(area_m2)) # 4
> str(b)
'data.frame': 766 obs. of 2 variables:
 $ apt_price: int  31000 46700 108000 105000 97300 59500 33500 33000 32000 46000
 $ area_m2  : num  244 219 197 197 197 ...
> head(b)
  apt_price area_m2
1    31000 244.0710
2    46700 219.0313
3   108000 196.9760
4   105000 196.9760
5    97300 196.9760
6    59500 196.0430
```

# dplyr 패키지를 활용한 데이터 마이닝

## 데이터 요약하기: summarise()

- 집단별 요약통계: summarise()
  - group\_by() 함수에 이어 사용되어짐
- 실습1:
  - 건축년도별 아파트 거래가격의 평균과 거래건수
- 실습2:
  - 건축년도별 읍면동(urban)별 아파트 건수와 아파트 평균 가격 통계

```
> apt %>% # 데이터 선택
+   group_by(year_built) %>% # 건축년도별 자료 리스트 작성
+   summarise(ave_price = mean(apt_price), sale_freq = n()) # 요약 통계
# A tibble: 43 x 3
  year_built ave_price sale_freq
  <int>      <dbl>      <int>
1    1977    10017.         3
2    1978     8729.         7
3    1979     5105.        44
4    1980     3706.        27
5    1981     4509.        37
6    1982     8215.        20
7    1983    10282.        25
8    1984     7597.        33
9    1985     6988.        75
10   1986     5926.       138
# ... with 33 more rows
```

```
> # 건축년도별 읍면동(urban)별 아파트 건수와 아파트 평균 가격 통계
> x <- apt %>%
+   group_by(year_built, urban) %>%
+   summarise(ave_price = mean(apt_price), sale_freq = n())
> x
# A tibble: 110 x 4
# Groups:   year_built [43]
  year_built urban ave_price sale_freq
  <int>    <fct>      <dbl>      <int>
1    1977   동     10017.         3
2    1978   동     8729.         7
3    1979   동     5105.        44
4    1980   동     3706.        27
5    1981   동     4509.        37
6    1982   동     8215.        20
7    1983   동    10741.        23
8    1983   읍     5000.         2
9    1984   동     7597.        33
10   1985   동     8339.        49
# ... with 100 more rows
> table(x$urban)
```

```
동 면 읍
43 31 36
```

# dplyr 패키지를 활용한 데이터 마이닝

## tbl\_df() 소개

- The `tbl_df` class is a subclass of `data.frame`, created in order to have different default behaviour.
  - The colloquial term "tibble" refers to a data frame that has the `tbl_df` class.
  - Tibble is the central data structure for the set of packages known as the tidyverse, including `dplyr`, `ggplot2`, `tidyr`, and `readr`.
- Objects of class `tbl_df` have:
  - A class attribute of `c("tbl_df", "tbl", "data.frame")`.
  - A base type of "list", where each element of the list has the same `NROW()`.
  - A `names` attribute that is a character vector the same length as the underlying list.

# dplyr 패키지를 활용한 데이터 마이닝

## tbl\_df() 소개

- `tbl_df` ( *tibble diff*라고도 함 )
  - [데이터 프레임](#) 의 변형
  - [tibble](#) 패키지로 구현 예
    - `install.packages("tibble")`
    - `library(tibble)`
      - `mtcars_tbl <- as_data_frame(mtcars)` 또는
      - `mtcars_tbl <- tbl_df(mtcars)`
- `data.frames`와 `tbl_df`의 차이점
  - 인쇄 된 출력에는 표의 크기 요약
  - 각 열의 유형
  - 제한된 수의 행 인쇄
- dplyr 패키지의 많은 함수는 `group_by()` 와 같은 `tbl_df`로 자연스럽게 작동

```
# A tibble: 32 x 11
  mpg   cyl  disp    hp  drat    wt   qsec    vs  am  gear  carb
*   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  21.0     6 160.0   110  3.90  2.620  16.46     0     1     4     4
2  21.0     6 160.0   110  3.90  2.875  17.02     0     1     4     4
3  22.8     4 108.0    93  3.85  2.320  18.61     1     1     4     1
4  21.4     6 258.0   110  3.08  3.215  19.44     1     0     3     1
5  18.7     8 360.0   175  3.15  3.440  17.02     0     0     3     2
6  18.1     6 225.0   105  2.76  3.460  20.22     1     0     3     1
7  14.3     8 360.0   245  3.21  3.570  15.84     0     0     3     4
8  24.4     4 146.7    62  3.69  3.190  20.00     1     0     4     2
9  22.8     4 140.8    95  3.92  3.150  22.90     1     0     4     2
10 19.2     6 167.6   123  3.92  3.440  18.30     1     0     4     4
# ... with 22 more rows
```

# dplyr 패키지를 활용한 데이터 마이닝

## tbl\_df() 소개

- 실습

```
## tbl_df()
?tibble
?tbl_df
?as_data_frame

library(tibble)
mtcars

(mtcars_tbl1 <- as_data_frame(mtcars))

(mtcars_tbl2 <- tbl_df(mtcars))
```

```
> x <- select(apt, apt_price:urban) # apt_price부터 urban 열까지 추출
> x
```

	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban
1	108000	196.9760	41	2010	201905	25	oln
2	105000	196.9760	42	2010	201902	13	oln
3	97300	196.9760	42	2010	201901	13	oln
4	68000	162.6260	45	2010	201810	18	oln
5	67000	131.9410	23	2015	201901	10	oln
6	64500	152.6520	39	2010	201907	19	oln
7	64000	152.6520	24	2010	201902	20	oln
8	63000	131.9410	24	2015	201809	28	oln
9	60800	152.6520	6	2010	201811	3	oln
10	60000	108.4651	29	2018	201905	11	oln
11	60000	152.6520	19	2010	201809	1	oln
12	60000	152.6520	30	2010	201812	29	oln
13	60000	152.6520	28	2010	201904	10	oln
...	.....	.....	..	....	.....	..	oln

```
> (x_tbl <- tbl_df(x)) # tibble 데이터 셋으로 변환
# A tibble: 13,309 x 7
```

	apt_price	area_m2	floor_no	year_built	ym_sale	day_sale	urban
	<int>	<dbl>	<int>	<int>	<int>	<int>	<fct>
1	108000	197.	41	2010	201905	25	oln
2	105000	197.	42	2010	201902	13	oln
3	97300	197.	42	2010	201901	13	oln
4	68000	163.	45	2010	201810	18	oln
5	67000	132.	23	2015	201901	10	oln
6	64500	153.	39	2010	201907	19	oln
7	64000	153.	24	2010	201902	20	oln
8	63000	132.	24	2015	201809	28	oln
9	60800	153.	6	2010	201811	3	oln
10	60000	108.	29	2018	201905	11	oln
# ... with 13,299 more rows							

## 연습문제 07

- apt 객체에서 읍면동(urban)별 거래된 아파트 가격(apt\_price)의 최댓값과 전용면적(area\_m2)의 최댓값을 구하시오.

## 연습문제 08

- apt 객체에서 충주시(address\_sigungu=="충주시")에서 전용면적(area\_m2)이 100보다 크거나 같고 apt\_price가 20000보다 크게 거래된 데이터를 추출하고 싶다. 이에 대한 함수(명령문)을 적고, 총 몇 개의 거래건수가 있는 지 적으시오.

## 연습문제 09

- apt 객체에서 1979년도에 건축된 아파트의 거래건수와 거래된 아파트의 평균가격을 dplyr패키지를 활용하여 구하고자 한다. 실행 함수를 작성하고, 거래건수와 평균 가격을 적으시오.



## 연습문제 10

- apt 데이터 프레임을 tibble 데이터 셋으로 변환하여 사용하고  
자 한다. 적용할 함수를 적고, 이의 장점을 서술하시오.
- 함수명:
- 장점:

# 요약

- 데이터 불러오기와 저장하기
  - 데이터의 종류와 불러오기(저장하기)
  - 내장 데이터 불러오기
  - 작업경로(폴더) 확인 및 변경
  - 외장 데이터 불러오기와 저장하기
    - 텍스트 파일(.txt, .csv, ...)
    - 엑셀 스프레드 시트(.xls, .xlsx)
- 데이터 마이닝 기초
  - 데이터 탐구하기
  - 데이터 정렬하기
  - 데이터 병합하기
  - 데이터 변환하기
  - 데이터 추출하기
  - 기타 유용한 함수들: (...), with, attach, detach
- dplyr 패키지를 활용한 데이터 마이닝
  - dplyr 패키지 소개
  - 추출하기: select()
  - 특정 조건에 맞는 데이터 추출하기: filter()
  - 정렬하기: arrange()
  - 변수명 변환하기: rename()
  - 데이터 수정 및 변형: mutate(), transmute()
  - 집단별 데이터 분할하기: group\_by()
  - 데이터 마이닝 간결하게 수행하기: %>% (파이프 연산자)
  - 데이터 요약하기: summarise()
  - 응용 실습
  - tbl\_df() 소개

# 끝

- 질의와 토의(Question & Discussion)
  - 이번 강의 내용을 시청하고, 실행하면서 궁금한 점이나 어려운 점에 대하여 토의해봅시다.
- 다음 주 강의주제
  - 그래프와 ggplot2 패키지