M21 - Flow

LearningSpoonsR

2019-01-06

The Flow

- 해당업체는 미국과 미국 외의 오프라인 지점에서 유아용 카시트를 판매하는 업체입니다.
- 도시별 판매 집계 데이터를 이용해 insight를 얻어봅니다.
- Part 1. 데이터 수집
  - `ISLR` package에서 `Carseat` 데이터 셋을 구합니다.
  - `ISLR`은 Introduction to Statistical Learning in R이라는 머신러닝 학술서적에 포함된 함수와 데이터를 포함한 패키지입니다.
  - 기초 통계에 대한 지식이 있으면 읽을 수 있는 좋은 서적입니다.
  - 영문판은 무료로 책이 베포되어 있어서 github에도 공유하였고, 번역서도 출판되어 있습니다.
- Part 2. 전처리
  - `dplyr`은 R에서 데이터 사이언스의 근간이 되는 `data.frame`을 효율적으로 다루는 패키지입니다.
  - Rstudio의 Scientist가 만들었기에 체계적이고 간결합니다.
- Part 3. 분석 & 시각화
  - `ggplot2`은 R을 다른 언어에 비해서 가장 돋보이게 하는 체계적이고 효과적인 시각화 도구입니다.
- Part 4. 문서화
  - `rmarkdown`은 분석 결과를 다양한 포맷의 문서로 만들어주는 도구입니다.
  - 데이터 분석 환경을 제공합니다.

Part 1. 수집 - from "ISLR" package

패키지 (package, 응용프로그램, library)

- 각각의 패키지는 특정 데이터나 함수를 제공함.
- R을 처음에 설치하면 **base**라는 패키지가 인스톨 되어있음.
- **base**외의 확장 기능을 제공하는 패키지를 설치하여 사용하게 됨.
- R은 오픈 소스 기반 언어로써, 누구나 패키지를 만들고 R프로그램의 발전에 기여할 수 있음.
- R과 Python과 같은 오픈소스 언어에서는 많은 사람들이 많은 패키지를 만들고 많은 사람들이 사용하면서 발전이 일어남.
- 저희 수업에서는 R에서 데이터 사이언스의 과정을 support하는 강력한 패키지 위주로 진행합니다.

- base외의 패키지를 사용하려면 먼저 설치를 해주어야 함.
- 따옴표를 넣어서 문자열로 입력

```
install.packages("packageName") # playstore, Appstore
```

- 설치이후에는 패키지 사용 전에 아래와 같이 패키지 사용을 선언해주어야 함.
- 따옴표 없이 입력
- #declare, #load, #import

```
library(packageName) # declaring "I will use this package!"
```

파일 (files)

1. 소스 파일

- 명령어와 주석을 기록해둔 파일
- 한번에 실행이 가능한 단위로 구성
- 저장/불러오기/편집 등이 가능함 (Rstudio에서 마치 오피스 처럼 다룰 수 있음)
- `.r` : R 소스 파일
- `.Rmd` : R Markdown 소스파일
- 서식(폰트의 종류나 크기)이 없는 파일이기 때문에 메모장과 같은 텍스트 에디터에서도 저장/불러오기/편집이 가능함.

2. 데이터 파일

- 데이터가 기록된 파일
- `.csv`
  - 컴마로 구분된 파일
  - (Comma Separated Values)
- `.txt`
  - 더 일반적인 텍스트 파일.
  - `\t`(Tab)이나 `\n`(줄바꿈, linebreak) 등의 기호로 구분되어 있음
- `.xls, xlsx`
  - 엑셀파일
  - 서식 등이 포함되어 있는 경우에는 [다른 이름으로 저장 - csv로 저장]하여 csv 파일로 저장하면 용량이 줄어들어서 다루기 좋은 경우가 많음.
  - 엑셀 파일을 바로 불러오는 것도 특정 패키지(`read_excel`, `readxl`)를 이용하면 가능하고 점점 발전하고 있음.

3. R 데이터 파일

- R에서 작업중인 데이터를 저장하고 불러오는 데 유용한 파일.
- `.Rdata`: R 작업시에 현재 메모리의 상태를 저장하고 나중에 불러올 수 있어서 작업을 이어서 하는데 유용
- `.Rda`: Data Structure 개체 (object) 1개를 저장하는 파일

데이터 파일 불러오기/저장하기

0. 파일을 불러오면 R에서는 기본적으로 **data.frame**으로 받아들임

1. **csv**파일

- 데이터 프레임을 생성하는 명령이기에 **stringsAsFactors = FALSE**의 옵션의 사용을 추천!
- 만약에 데이터 파일의 첫 번째 행이 제목인 경우 이를 column name으로 만들 수 있음. (이게 default임, 즉 **header=TRUE**를 입력하지 않더라도 **header=TRUE**로 동작함)

```
dataset <- read.csv("fileName.csv", header = TRUE, stringsAsFactors = FALSE)
```

- 만약에 첫 번째 line이 제목이 아니라면 **header=FALSE**를 사용해야 겠죠?

```
dataset <- read.csv("fileName.csv", header = FALSE, stringsAsFactors = FALSE) # load
```

- **data.frame**을 파일로 저장할 때에는 아래의 명령으로…

```
write.csv(dataset, "fileName.csv") # save
```

2. txt파일

- csv파일과 매우 유사함.
- csv파일은 comma로 나누어져 있기에 별도의 입력이 필요하지 않지만, txt파일에서는 무엇으로 나누어져있는가를 입력해주어야 함. 이를 seperator(sep)라고 함.

```
datasaet <- read.table("fileName.txt", header = FALSE, sep = "\t", stringsAsFactors = FALSE
write.table(dataset, "fileName.txt") # save
```

3. xls, xlsx파일

- 불러오기는 가능, 저장은 불가능

```
library(readxl)
dataset <- read_excel("fileName.xlsx")
dataset <- read_excel("fileName.xlsx", col_names = FALSE)
           # (same as `header=FALSE`)
```

4. R 데이터 파일

- 데이터 셋 한개

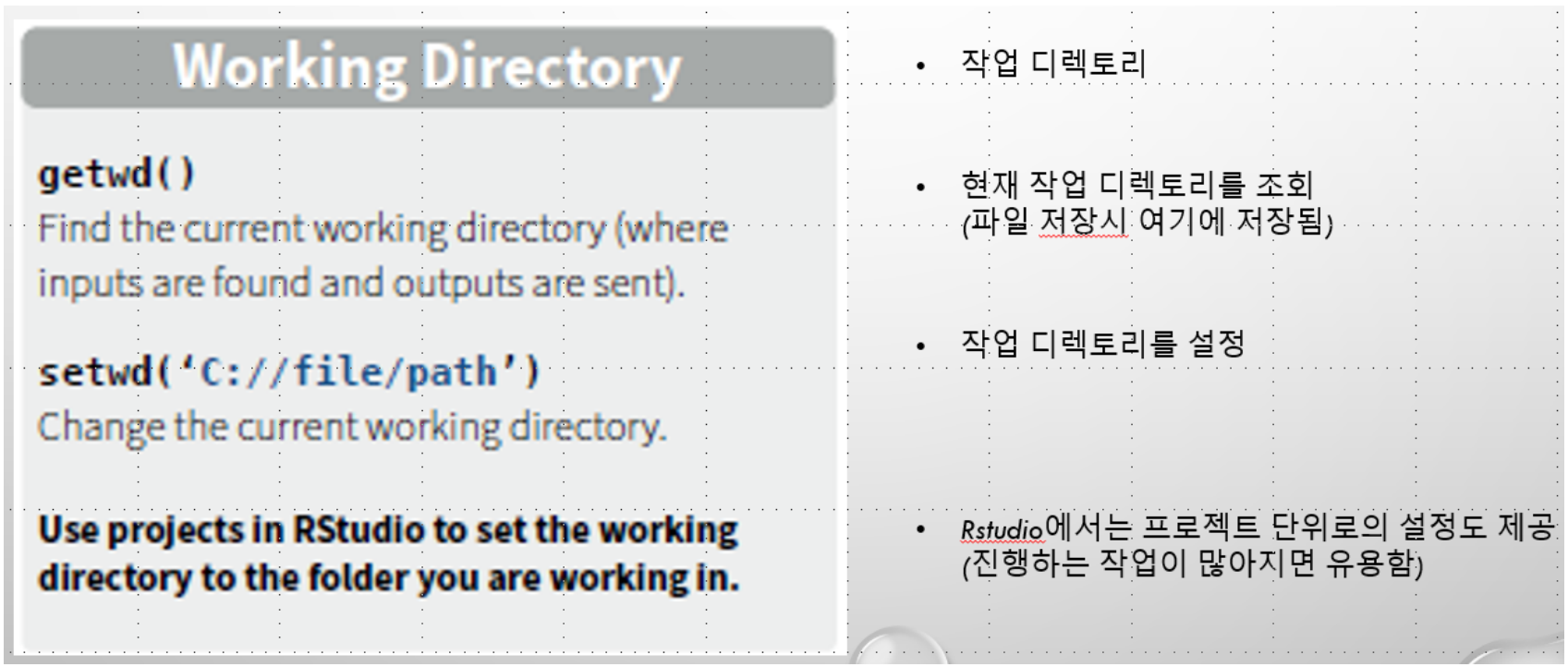
```r
save(dataset, "fileName.rda") # save
load("fileName.rda") # load
```

- 현재 메모리의 모든 객체 (변수와 함수)

```r
save("fileName.Rdata") # save
load("fileName.Rdata") # load
```

경로 (디렉토리, directory)

- **Rstudio를 어떻게 실행시켰냐에 따라서 현재 작업 디렉토리가 설정됨.**
  - Rstudio 아이콘을 더블클릭하여 실행하였을 때에는
  - ⇒ 시스템의 기본 설정대로 현재 작업 디렉토리가 정해짐.
  - 이미 존재하는 `.R`이나 `.Rmd` 파일을 더블클릭하여 Rstudio를 실행했을 때에는
  - ⇒ 해당 파일의 경로가 현재 작업 디렉토리가 됨.
- 확인 및 변경
  - 현재 작업 디렉토리를 확인하려면 `getwd()`를 실행.
  - 작업 디렉토리를 바꾸려면 `setwd("C:/LS-DS")`을 실행.
- !디렉토리의 입력은 "C:/LS-DS", "C://LS-DS", "C:\\LS-DS"의 방식이 가능함.



도시별 카시트 세일즈 데이터 불러오기 (ISLR 패키지)

```
install.package("ISLR") # installation
```

```
library(ISLR) # load package
class(Carseats) # name of data structure
```

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

```
head(Carseats) # the first 6 observations
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age
## 1  9.50       138     73           11        276   120       Bad  42
## 2 11.22       111     48           16        260    83      Good  65
## 3 10.06       113     35           10        269    80    Medium  59
## 4  7.40       117    100            4        466    97    Medium  55
## 5  4.15       141     64            3        340   128       Bad  38
## 6 10.81       124    113           13        501    72       Bad  78
##   Education Urban  US
## 1        17   Yes Yes
## 2        10   Yes Yes
## 3        12   Yes Yes
## 4        14   Yes Yes
## 5        13   Yes  No
## 6        16    No Yes
```

```r
tail(Carseats, 5) # the last 5 observations
```

```
##     Sales CompPrice Income Advertising Population Price ShelveLoc Age
## 396 12.57       138    108          17        203   128      Good  33
## 397  6.14       139     23           3         37   120    Medium  55
## 398  7.41       162     26          12        368   159    Medium  40
## 399  5.94       100     79           7        284    95       Bad  50
## 400  9.71       134     37           0         27   120      Good  49
##     Education Urban  US
## 396        14   Yes Yes
## 397        11    No Yes
## 398        18   Yes Yes
## 399        12   Yes Yes
## 400        16   Yes Yes
```

```r
View(Carseats) # pop-up windows
dim(Carseats) # dimensions
```

```
## [1] 400  11
```

```r
str(Carseats) # structural view
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

```
summary(Carseats) # summary statistics
```

```
##      Sales          CompPrice       Income        Advertising
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc        Age
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75
##  Median :272.0   Median :117.0   Medium:219   Median :54.50
##  Mean   :264.8   Mean   :115.8                Mean   :53.32
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00
##  Max.   :509.0   Max.   :191.0                Max.   :80.00
##    Education    Urban       US
##  Min.   :10.0   No :118   No :142
##  1st Qu.:12.0   Yes:282   Yes:258
##  Median :14.0
##  Mean   :13.9
##  3rd Qu.:16.0
##  Max.   :18.0
```

Part 2. 전처리 - ”dplyr”

dplyr?

1. 빠르고 직관적으로 `data.frame`을 다루는 패키지
    - 가장 빠른 언어인 C를 기반으로 만들어서 빠름
    - 데이터 처리에 있어서 가장 직관적인 SQL (Standard Query Language)과 유사하게 만들어져서 직관적임!
    - 직관적이어서 코드 가독성도 높음
    - **그러나 base 명령어도 같이 알아 두면 타인의 코드 참조와 파이썬, SQL등 다른 언어를 배울 때 도움이 됨**
2. Hadley Wickham, Ph.D.
    - Head Scientist, Rstudio
    - youtube.com에 keynote등 좋은 동영상 많아요!
    - 통계학 박사 후 R에서 다수의 사용하기 좋은 패키지 개발
    - 누나도 통계학 전공 교수
    - `ggplot2`, `dplyr`, `reshape2`, `stringr` 등의 본인이 작성한 패키지를 묶어서 `tidyverse`라는 패키지를 제작

Basic Manipulation

- tidy data.frame!

1. 개체 타입은 `data.frame`
2. 각각의 row는 관찰값을 의미
3. 각각의 column은 변수를 의미

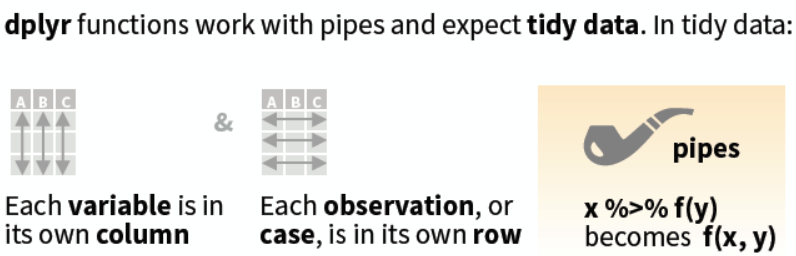


Figure 1: from `dplyr` Cheatsheet

- Basic Manipulations
  1. `rename`
     - 변수 이름 바꾸기
     - column의 이름을 바꿈
  2. `filter`
     - 관찰값 추출
     - row를 선택함
  3. `select`
     - 변수 추출
     - column을 선택함
  4. `arange`
     - 관찰값 정렬
     - row를 재정렬함
  5. `mutate`
     - 변수 생성
     - column을 만듬
  6. `group_by` + `summarise`
     - Categorical 변수로 집계

1. rename (이름 바꾸기)

```
# rename Sales to Revenue
Carseats <- rename(Carseats, Revenue = Sales) # dplyr
names(Carseats)[names(Carseats)=="Sales"] <- "Revenue" # base
```

2. filter (관찰값 추출, Row 추출)

```
# takes only if Income > 100
temp <- filter(Carseats, Income > 100) # dplyr
temp <- Carseats %>% filter(Income > 100) # dplyr, pipe
temp <- Carseats[Carseats$Income > 100,] # base
# takes only if Age between 30 and 40
temp <- filter(Carseats, Age >= 30 & Age < 40) # dplyr
temp <- Carseats %>% filter(Age >= 30 & Age < 40) # dplyr, pipe
temp <- Carseats[((Carseats$Age >= 30) & (Carseats$Age < 40)),] # base
```

- &는 “and”
- |는 “or”
- pipe는 가독성을 높여줍니다.
  - $y = tanh(\sqrt{(x_1 + x_2)^2 + x_3}) - x_4$
  - `y <- x_1 %>% add(x_2) %>% square() %>% add(x_3) %>% sqrt() %>% tanh() %>% subtract(x_4)`

3. select (변수 추출, Column 선택)

```r
# Choose the variablse Income and Population
temp <- select(Carseats, Income, Population) # dplyr
temp <- Carseats %>% select(Income, Population) # dplyr
temp <- Carseats[,c("Income", "Population")] # base
```

4. arrange (정렬)

```r
# Ascending (1-2-3)
Carseats <- arrange(Carseats, Price) # dplyr
Carseats <- Carseats %>% arrange(Price) # dplyr
Carseats <- Carseats[order(Carseats$Price),] # base
# Descending (3-2-1)
Carseats <- arrange(Carseats, desc(Price)) # dplyr
Carseats <- Carseats %>% arrange(desc(Price)) # dplyr
Carseats <- Carseats[order(Carseats$Price, decreasing = TRUE),] # base
```

5. mutate (새로운 변수)

```r
# base
Carseats$Revenue <- Carseats$Sales

# dplyr
Carseats <- mutate(Carseats,
                   AdvPerCapita = Advertising/Population,
                   RevPerCapita = Revenue/Population)

# dplyr (with pipe)
Carseats <- Carseats %>%
  mutate(AdvPerCapita = Advertising/Population,
         RevPerCapita = Revenue/Population)

# base
Carseats$AdvPerCapita <- Carseats$Advertising/Carseats$Population
Carseats$RevPerCapita <- Carseats$Revenue/Carseats$Population

# mutate with ifelse
Carseats <- mutate(Carseats, AgeClass = ifelse(Age>=60, "Silver", "non-Silver"))
Carseats <- Carseats %>% mutate(AgeClass = ifelse(Age>=60, "Silver", "non-Silver"))
Carseats$AgeClass <- ifelse(Carseats$Age >= 60, "Silver", "non-Silver")
```

Carseats

- 해당 Carseat 회사의 주요 구매층은 아이를 낳아서 키울 나이인 30대 연령층입니다.
- 소득이 높으면서 도시의 평균 연령이 30대인 도시에 충분한 광고비를 지출하고 있나요?

```r
# Successive treatments by pipe
focusCity <- Carseats %>%
  filter(Income > 100) %>%
  filter(Age >= 30 & Age < 40) %>%
  mutate(AdvPerCapita = Advertising/Population) %>%
  select(Sales, Income, Age, Population, Education, AdvPerCapita) %>%
  arrange(Sales) # ascending
print(focusCity)
```

```
##    Sales Income Age Population Education AdvPerCapita
## 1   5.04    114  34        298        16   0.00000000
## 2   5.32    116  39        170        16   0.00000000
## 3   6.80    117  38        337        10   0.01483680
## 4   7.49    119  35        178        13   0.03370787
## 5   7.67    117  36        400        10   0.02000000
## 6   8.55    111  36        480        16   0.04791667
## 7   8.97    107  33        144        13   0.00000000
## 8   9.03    102  35        123        16   0.10569106
## 9   9.39    118  32        445        15   0.03146067
## 10  9.58    104  37        353        17   0.06515581
## 11 10.36    105  34        428        12   0.04205607
## 12 10.59    120  30        262        10   0.05725191
## 13 12.57    108  33        203        14   0.08374384
```

- 도시의 평균 나이가 20대, 30대, 40대 이상인 경우에 Revenue에 차이가 있을까요?

```
Carseats %>%
  mutate(AgeClass =
           ifelse(Age < 30, "Twenties",
                  ifelse(Age < 40, "Thirties", "FourtyAbove"))) %>%
  group_by(AgeClass) %>%
  summarise(avgRevenue = mean(Sales))
```

```
## # A tibble: 3 x 2
##   AgeClass    avgRevenue
##   <chr>            <dbl>
## 1 FourtyAbove       7.30
## 2 Thirties          8.26
## 3 Twenties          7.76
```

- Process
  1. AgeClass라는 factor변수 생성 (mutate)
  2. AgeClass로 묶어서(group_by)
  3. mean(Sales)를 집계(summarise)
  4. 새로운 data.frame이 만들어 짐

Discussion

- 많은 데이터 분석은 “연속 변수”를 “이산 변수”로 변환하여,
- “이산 변수”의 group별 차이를 파악하는 접근을 사용합니다.
- ex) 연령, 소득, 기온, 성별,…
- Later…

1. **`Age`**를 factor로 잡아 **`Sales`**의 Boxplot.
2. **`Age`**를 factor로 잡아 scatterplot for **`Population`** & **`Sales`**.
3. **변수 3개**를 사용하는 습관
4. **공간**과 **시간**에 대한 동질성과 이질성을 생각하는 습관

- **`#generatlizationIsLearning`**
- **`#universalTruth #specificTime&Place`**

Discussion

- 평균 나이에 따른 평균 Revenue의 차이는 무엇을 말해주나요?
- 평균 나이가 포함하지 못하고 있는 정보는 어떤 것이 있나요?
- 어떤 데이터가 있으면 더 좋을까요?
- 그 데이터가 있으면 어떻게 하실 건가요?

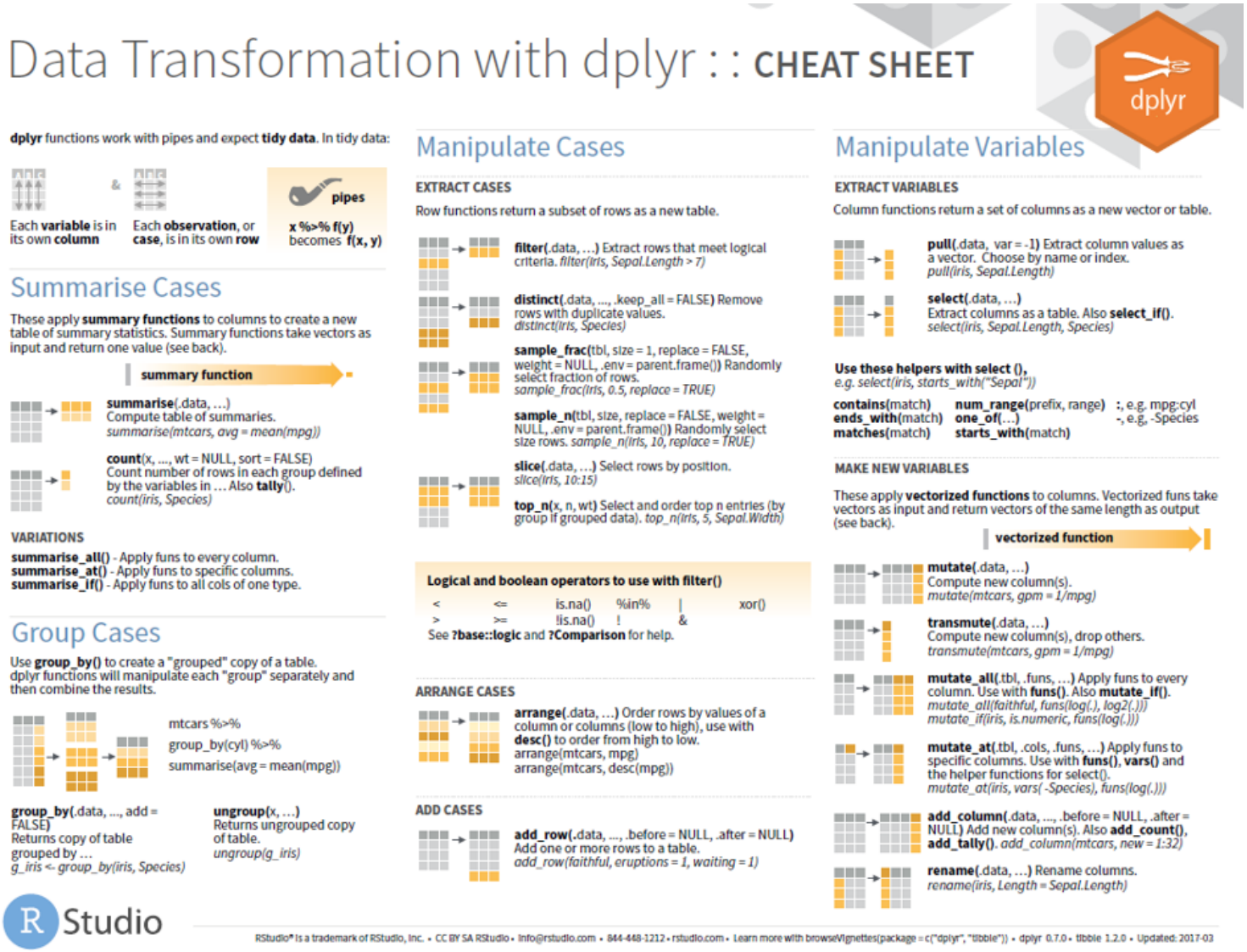
평균 값이 아닌 "분포"가 더 중요한 정보!

Discussion

dplyr	base
Everyday Language	*Classic Programming language*
사용성	타 언어와 범용적
SQL	*Pandas package in Python*

- SQL (Standard Query Language)
  1. 대용량 데이터 베이스와 통신하는 언어
  2. R에서도 `sqldf`라는 패키지를 사용해서 SQL명령어로 작업할 수 있음
  3. R을 할 줄 아는 사람이 SQL을 배우는데 걸리는 시간 < 1 day
  4. 데이터를 보는 눈을 키워 줍니다.
- 데이터를 전처리하는 작업의 소요시간은 전체 프로젝트에서 80% 이상입니다.

Cheatsheet for `dplyr` (1)



Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE)
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …)
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …) Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL) Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)
ends_with(match)
matches(match)
num_range(prefix, range)
one_of(…)
starts_with(match)
:, e.g. mpg:cyl
-, e.g. -Species

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

mutate(.data, …)
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

transmute(.data, …)
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)

mutate_all(.tbl, .funs, …) Apply funs to every column. Use with **funs()**. Also **mutate_if()**.
mutate_all(faithful, funs(log(.), log2(.)))
mutate_if(iris, is.numeric, funs(log(.)))

mutate_at(.tbl, .cols, .funs, …) Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().
mutate_at(iris, vars(-Species), funs(log(.)))

add_column(.data, ..., .before = NULL, .after = NULL) Add new column(s). Also **add_count()**, **add_tally()**. *add_column(mtcars, new = 1:32)*

rename(.data, …) Rename columns.
rename(iris, Length = Sepal.Length)

RStudio

Cheatsheet for `dplyr` (2)

Vector Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS

dplyr::**lag()** - Offset elements by 1
dplyr::**lead()** - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::**cumall()** - Cumulative all()
dplyr::**cumany()** - Cumulative any()
cummax() - Cumulative max()
dplyr::**cummean()** - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

RANKINGS

dplyr::**cume_dist()** - Proportion of all values <=
dplyr::**dense_rank()** - rank with ties = min, no gaps
dplyr::**min_rank()** - rank with ties = min
dplyr::**ntile()** - bins into n bins
dplyr::**percent_rank()** - min_rank scaled to [0,1]
dplyr::**row_number()** - rank with ties = "first"

MATH

+, - , *, /, ^, %/%, %% - arithmetic ops
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - logical comparisons
dplyr::**between()** - x >= left & x <= right
dplyr::**near()** - safe == for floating point numbers

MISC

dplyr::**case_when()** - multi-case if_else()
dplyr::**coalesce()** - first non-NA values by element across a set of vectors
dplyr::**if_else()** - element-wise if() + else()
dplyr::**na_if()** - replace specific values with NA
pmax() - element-wise max()
pmin() - element-wise min()
dplyr::**recode()** - Vectorized switch()
dplyr::**recode_factor()** - Vectorized switch() for factors

Summary Functions

TO USE WITH SUMMARISE ()

summarise() applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

summary function

COUNTS

dplyr::**n()** - number of values/rows
dplyr::**n_distinct()** - # of uniques
sum(!is.na()) - # of non-NA's

LOCATION

mean() - mean, also **mean(!is.na())**
median() - median

LOGICALS

mean() - Proportion of TRUE's
sum() - # of TRUE's

POSITION/ORDER

dplyr::**first()** - first value
dplyr::**last()** - last value
dplyr::**nth()** - value in nth location of vector

RANK

quantile() - nth quantile
min() - minimum value
max() - maximum value

SPREAD

IQR() - Inter-Quartile Range
mad() - median absolute deviation
sd() - standard deviation
var() - variance

Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

rownames_to_column()
Move row names into col.
a <- rownames_to_column(iris, var = "C")

column_to_rownames()
Move col in row names.
column_to_rownames(a, var = "C")

Also **has_rownames()**, **remove_rownames()**

Combine Tables

COMBINE VARIABLES

Use **bind_cols()** to paste tables beside each other as they are.

bind_cols(...) Returns tables placed side by side as a single table.
BE SURE THAT ROWS ALIGN.

Use a "**Mutating Join**" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

left_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix=c(".x",".y"),...)
Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy=FALSE, suffix=c(".x",".y"),...)
Join data. Retain all values, all rows.

Use **by = c("col1", "col2")** to specify the column(s) to match on.
left_join(x, y, by = "A")

Use a named vector, **by = c("col1" = "col2")**, to match on columns with different names in each data set.
left_join(x, y, by = c("C" = "D"))

Use **suffix** to specify suffix to give to duplicate column names.
left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))

COMBINE CASES

Use **bind_rows()** to paste tables below each other as they are.

bind_rows(..., .id = NULL)
Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured)

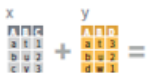
intersect(x, y, ...)
Rows that appear in both x and y.

setdiff(x, y, ...)
Rows that appear in x but not y.

union(x, y, ...)
Rows that appear in x or y. (Duplicates removed). union_all() retains duplicates.

Use **setequal()** to test whether two data sets contain the exact same rows (in any order).

EXTRACT ROWS



Use a "**Filtering Join**" to filter one table against the rows of another.

semi_join(x, y, by = NULL, ...)
Return rows of x that have a match in y.
USEFUL TO SEE WHAT WILL BE JOINED.

anti_join(x, y, by = NULL, ...)
Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.

Part 3. 분석 & 시각화 - ‘ggplot2’

”The simple graph has brought more information to the data analyst’s mind than any other device.”
(간단한 그래프는 다른 어떤 장치보다 데이터 분석가의 마음에 더 많은 정보를 제공합니다.)
- John Tukey

mpg

- `ggplot2` 패키지에 내장된 데이터셋
- 1999년 부터 2008년 까지의 38개 차종 연비 데이터
- 데이터셋 `diamond`, `iris`와 함께 R에서 가장 많이 예제로 쓰임

변수 이름	변수 설명
manufacturer	
model	차종 *Model name*
displ	엔진 크기 *engine displacement, in litres*
year	연식 *year of manufacture*
cyl	기통 *number of cylinders*
trans	트랜스미션 *type of transmission*
drv	전륜/후륜/사륜 *f = front-wheel drive, r = rear wheel drive, 4 = 4wd*
cty	도심마일리지 *city miles per gallon*
hwy	고속도로마일리지 *highway miles per gallon*
fl	연료 종류 *fuel type*
class	*(세단, SUV...) "type" of car*

```r
library(ggplot2)
? mpg # getting help
```

```
## starting httpd help server ... done
```

```r
mpg
```

```
## # A tibble: 234 x 11
##    manufacturer model displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4      1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4      1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4      2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4      2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4      2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4      2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4      3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 q~   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 q~   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 q~   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

```r
class(mpg)
```

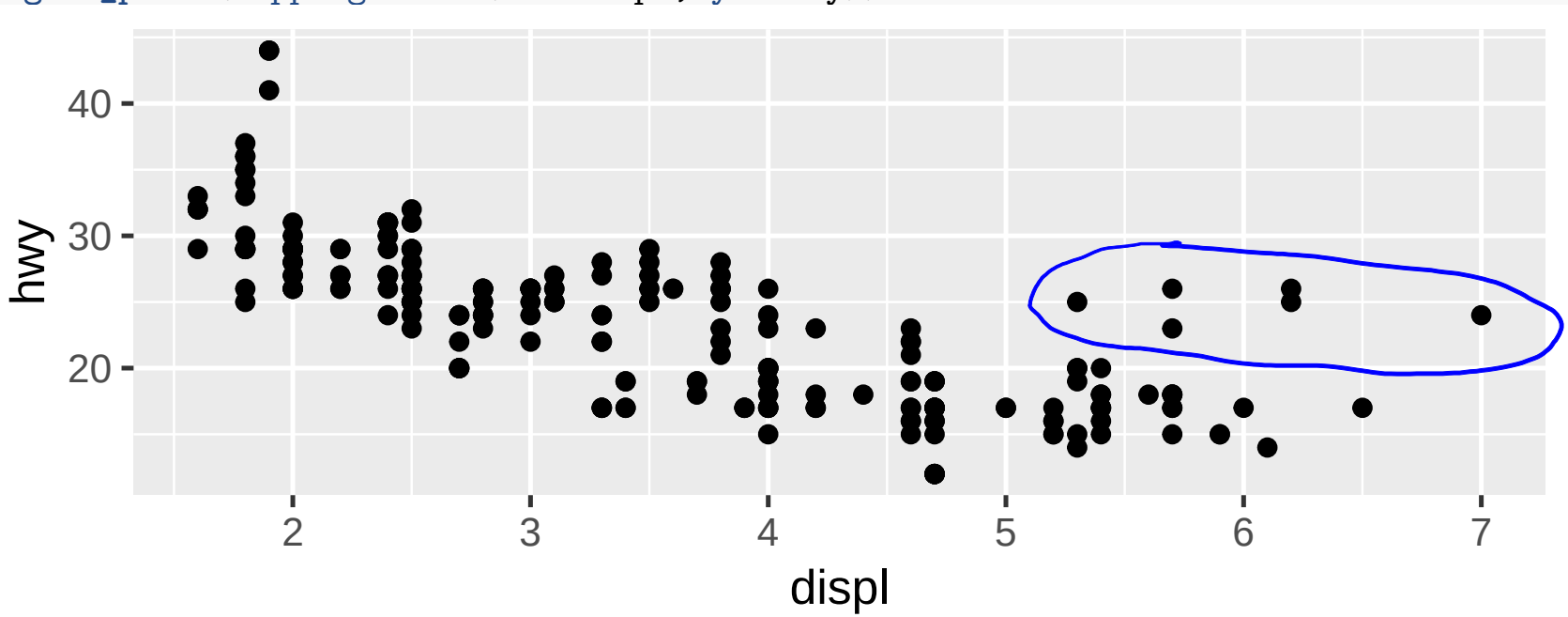
```
## [1] "tbl_df"     "tbl"        "data.frame"
```

- `mpg`는 여러개의 class의 특징을 가지고 있습니다.
- `tbl_df`는 `data.frame`의 업그레이드 버전입니다.
  - `data.frame`의 모든 기능 사용
  - 몇 가지 기능 추가

엔진이 크면 연비가 안 좋을까요?

- 엔진 크기를 x축, 고속도로 마일리지를 y축으로 하는 산점도 (scatterplot)을 그려봅시다.

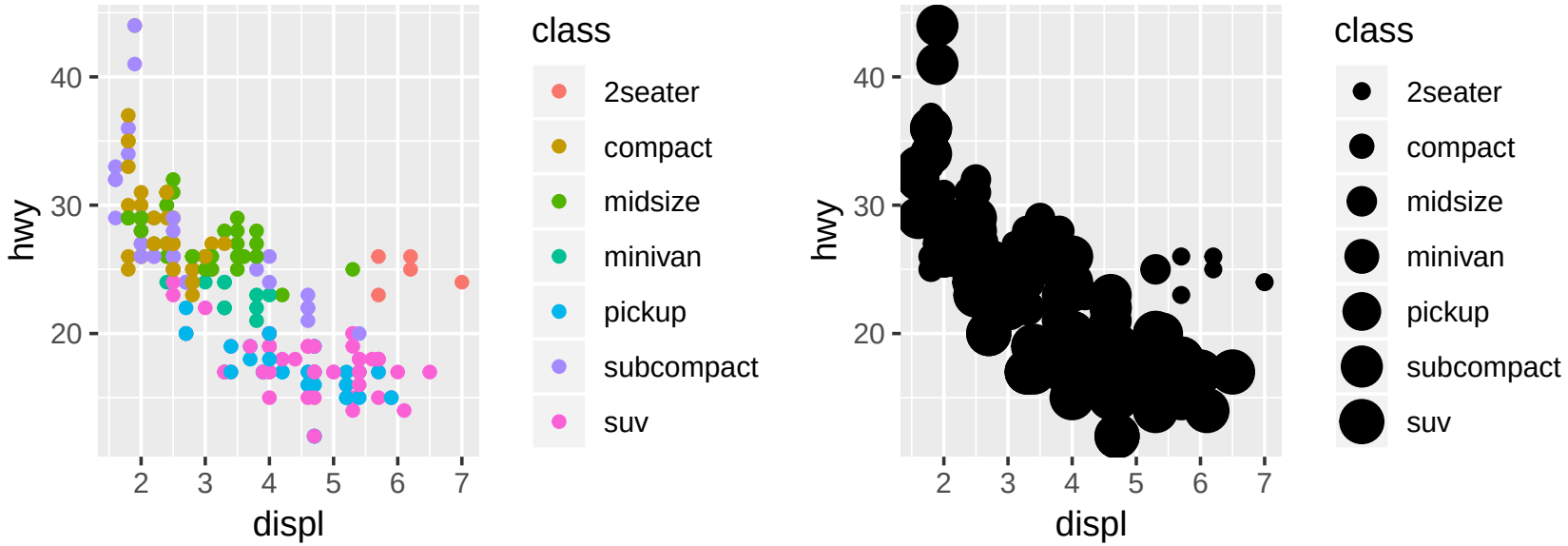
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```




```
# basic ggplot command
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPING>))
```

- 엔진이 크면 연비가 안 좋나요?
- 추가적으로 궁금한게 있나요?

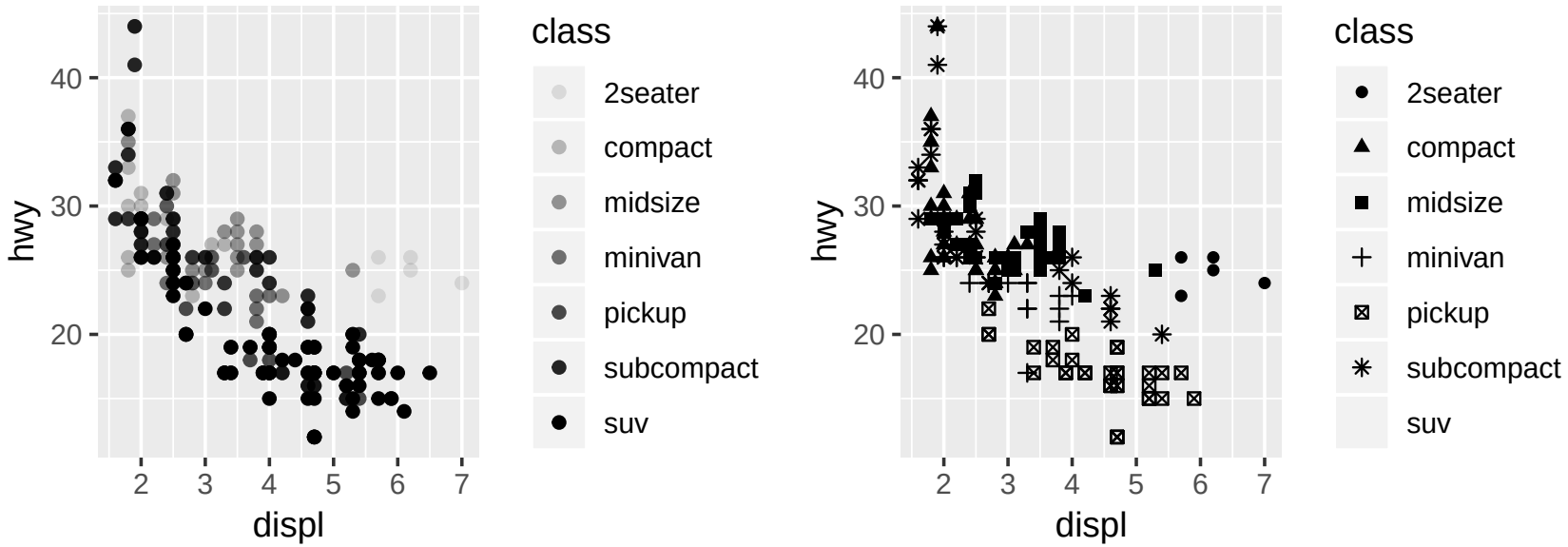
```r
suppressMessages(library(gridExtra)) # grid.arrange()
a <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
b <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
grid.arrange(a, b, nrow=1, ncol=2)
```



- `color=class`: `class` 변수의 값에 따라 color가 다르게
- `size=class`: `class`변수의 값에 따라 size가 다르게
- `a`와 `b`는 각각 graph 객체
- `gridExtra`는 grid.arrange 함수를 사용하게 해 줌
- `suppressMessages()`는 "로딩되었습니다."와 같은 메시지를 보기 싫을때 선택적으로 사용
- `grid.arrange(a, b, nrow=1, ncol=2)`: `a`와 `b`를 1개행, 2개열에 배치

```
c <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))
d <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
grid.arrange(c, d, nrow=1, ncol=2)
```

```
## Warning: Using alpha for a discrete variable is not advised.
## Warning: The shape palette can deal with a maximum of 6 discrete values
## because more than 6 becomes difficult to discriminate; you have 7.
## Consider specifying shapes manually if you must have them.
```



- `alpha=class`: `class` 변수의 값에 따라 alpha(진하기)가 다르게
- `shape=class`: `class`변수의 값에 따라 shape이 다르게
- `a`, `b`, `c`, `d` 중에 어느 그림이 가장 효과적인가요?
- 일반화 시키실 수 있나요? 즉, `size`, `alpha`, `color`, `shape`의 사용 기준은?

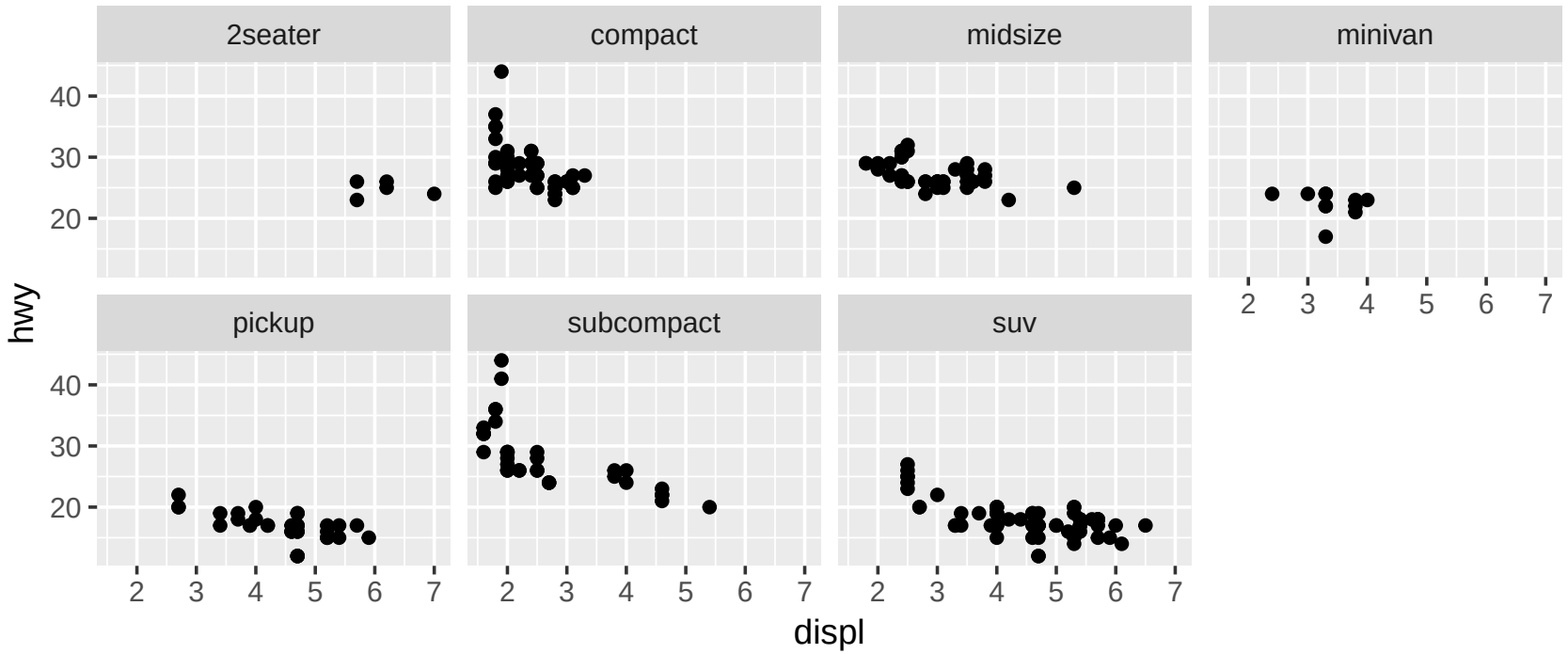
Aesthetics (`aes`) - size, alpha, color, shape

변수 타입	ggplot aes		특징
Numeric (quantitative, 정량적)	**`size`** **`alpha`**	수치	더할 수 있음 크고 작음
Factor (qualitative, 정량적)	**`color`** (<10) **`shape`** (<7)	분류	더할 수 없음 집합 연산

자주 쓰이는 ggplot의 추가 기능들

Facets (1 variable) - facet_wrap

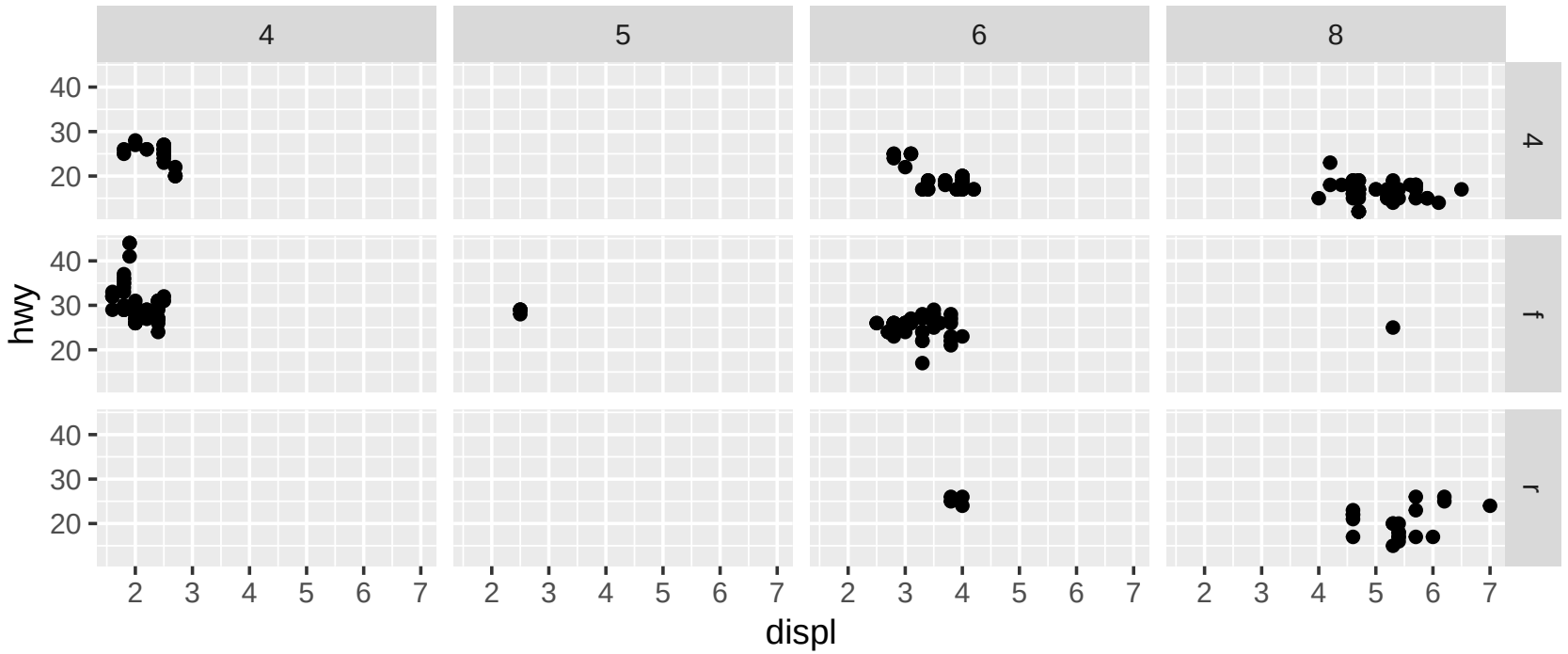
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```



- `facet_wrap(~ class, nrow = 2)`
  - `class`를 x축으로 하여
  - 2행으로 배열

Facets (2 variables) - facet_grid

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



- `facet_grid(drv ~ cyl)`
  - `cyl`을 x축으로
  - `drv`를 y축으로
  - grid 모양으로 배열

Discussion

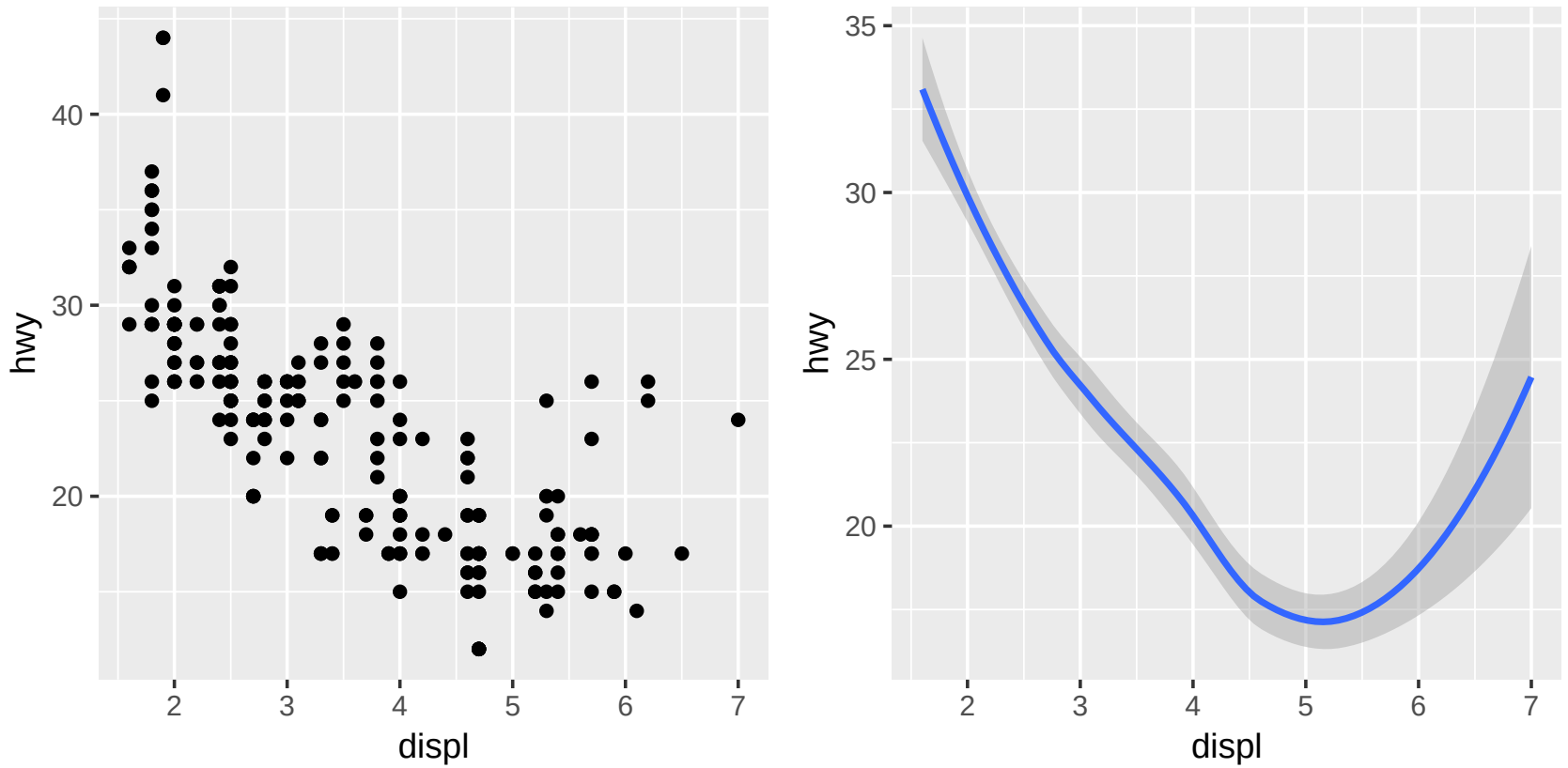
- When or not to facet?

1. Data가 충분히 많을 때 -> do facet
2. 각각의 Distribution을 보고 싶을 때 -> do facet
3. 근접 비교를 하고 싶을 때 -> don't facet
4. 결과물의 reader가 content에 대한 사전 지식 수준이 높을때 -> may facet
5. factor 변수의 **갯수**를 고려해야 함 -> **magic number**를 염두

(5~7개 max)

Point vs Smooth

```
e <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
f <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
grid.arrange(e, f, nrow = 1, ncol = 2)
```

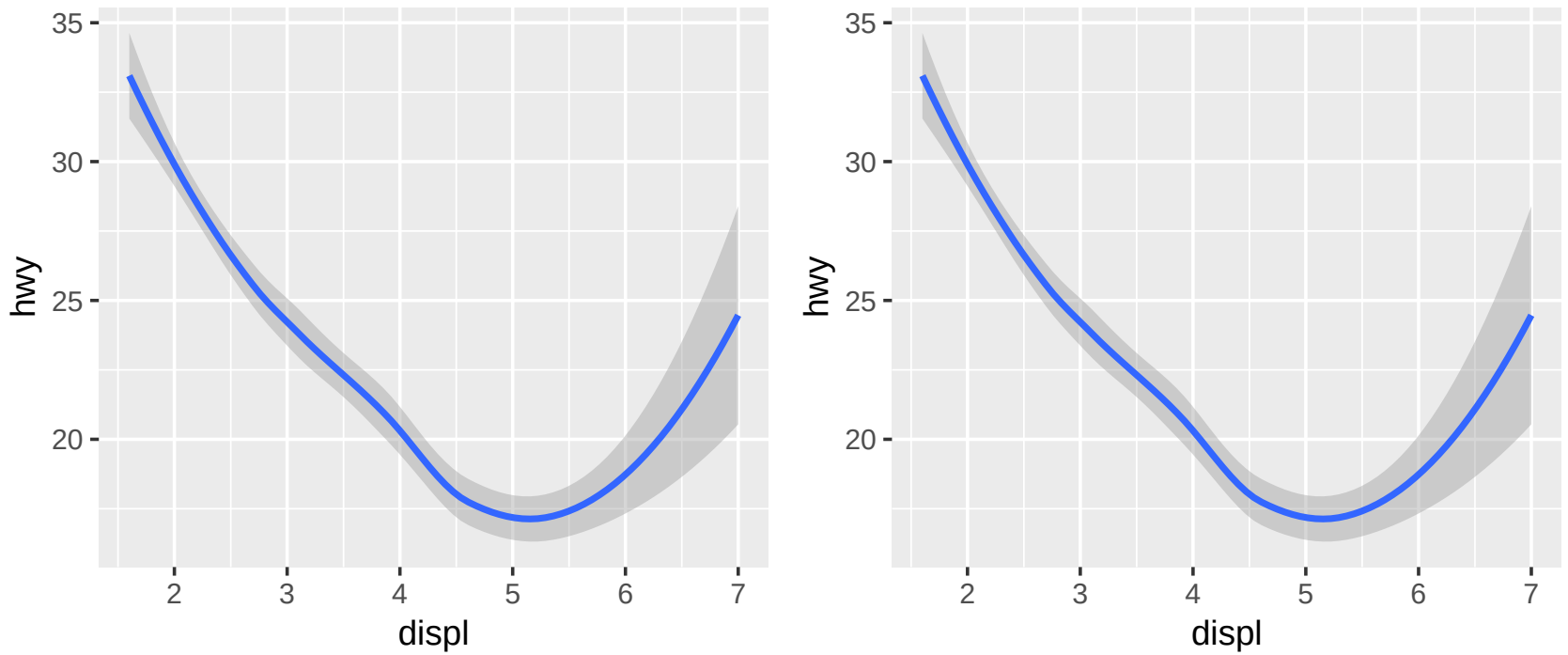


- geom_point: 점으로 표시
- geom_smooth: curve로 표시

AES mapping의 중첩

```
g <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
h <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth()
grid.arrange(g, h, nrow = 1, ncol = 2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

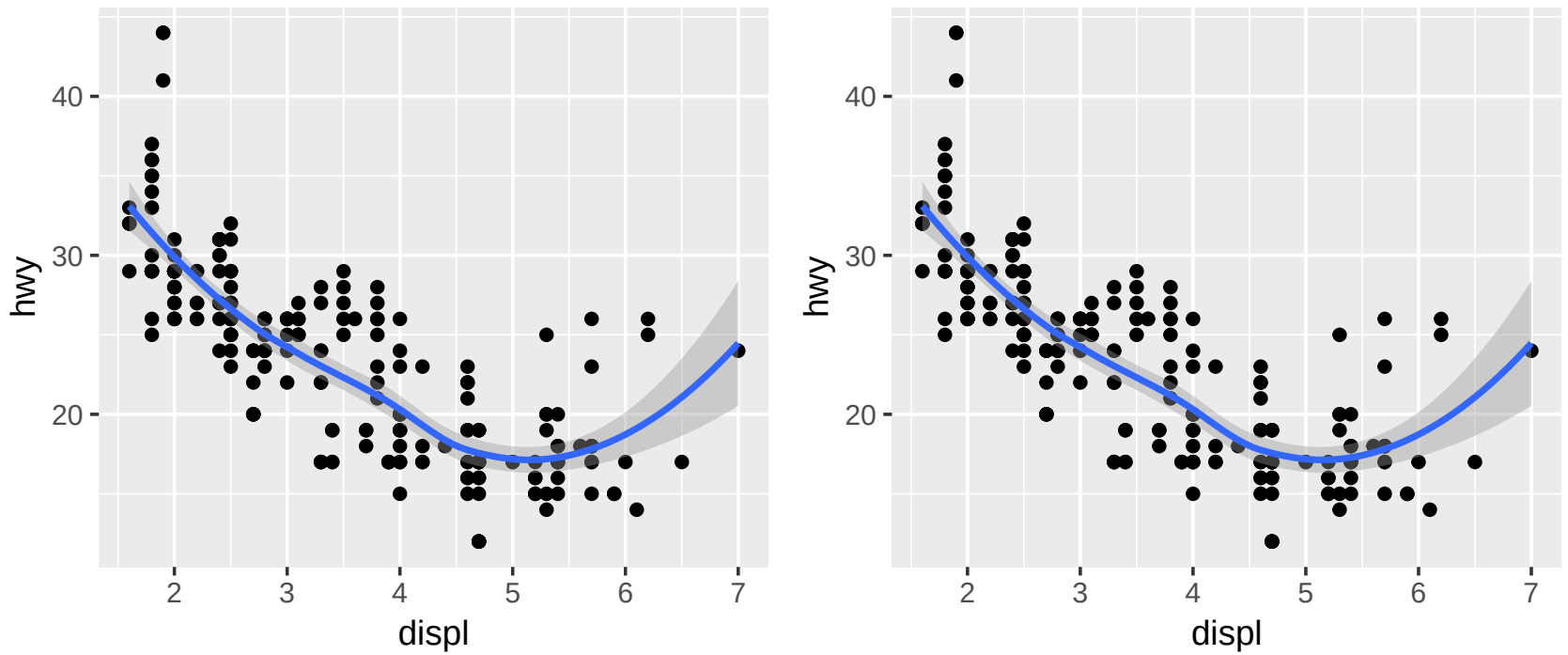


- g와 h는 동일하지만, 명령어의 variation을 주었음
- 즉, h의 geom_smooth()에서는 ggplot함수의 mapping 값을 가져옴

Point + Smooth

```
i <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
j <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
grid.arrange(i, j, nrow = 1, ncol = 2)
```

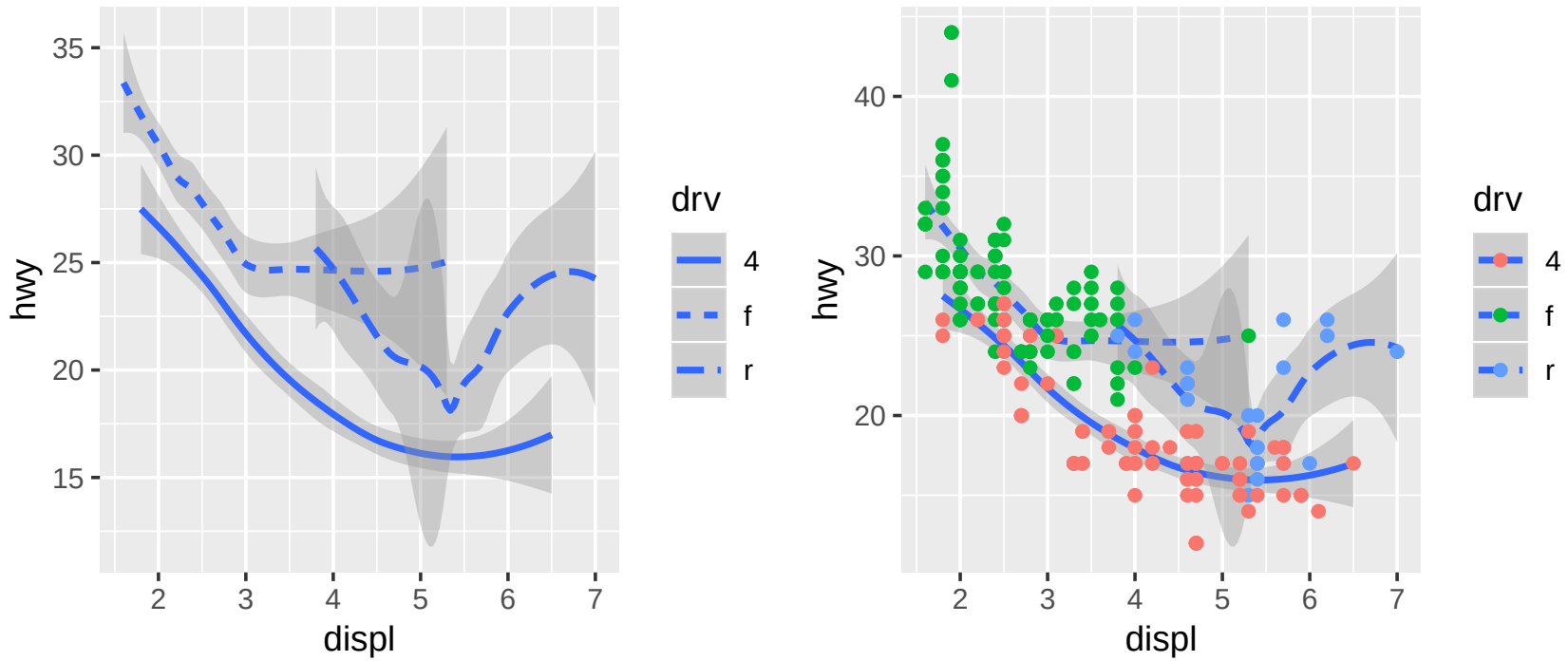
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- j의 _point()와 _smooth()가 앞의 mapping을 상속받음
- 코드가 간결해짐

```r
k <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(aes(linetype = drv))
l <- ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_smooth(aes(linetype = drv)) +
  geom_point(aes(color = drv))
grid.arrange(k, l, nrow = 1, ncol = 2)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



- l에서 mapping을 상속받으면서 _smooth()와 _point()가 더욱 구체화 됨

Summary - ggplot

- **Data**, **Aesthetic**, **Geometric Object**를 명시하여 그림을 rendering.
- **gridExtra** 패키지의 **grid.arrange** 함수를 이용하여 grid 형태로 위치시킬 수 있음
- 변수의 성격을 이해하여 **size, alpha, color, shape**와 **facet**을 적절히 사용

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPING>))
```

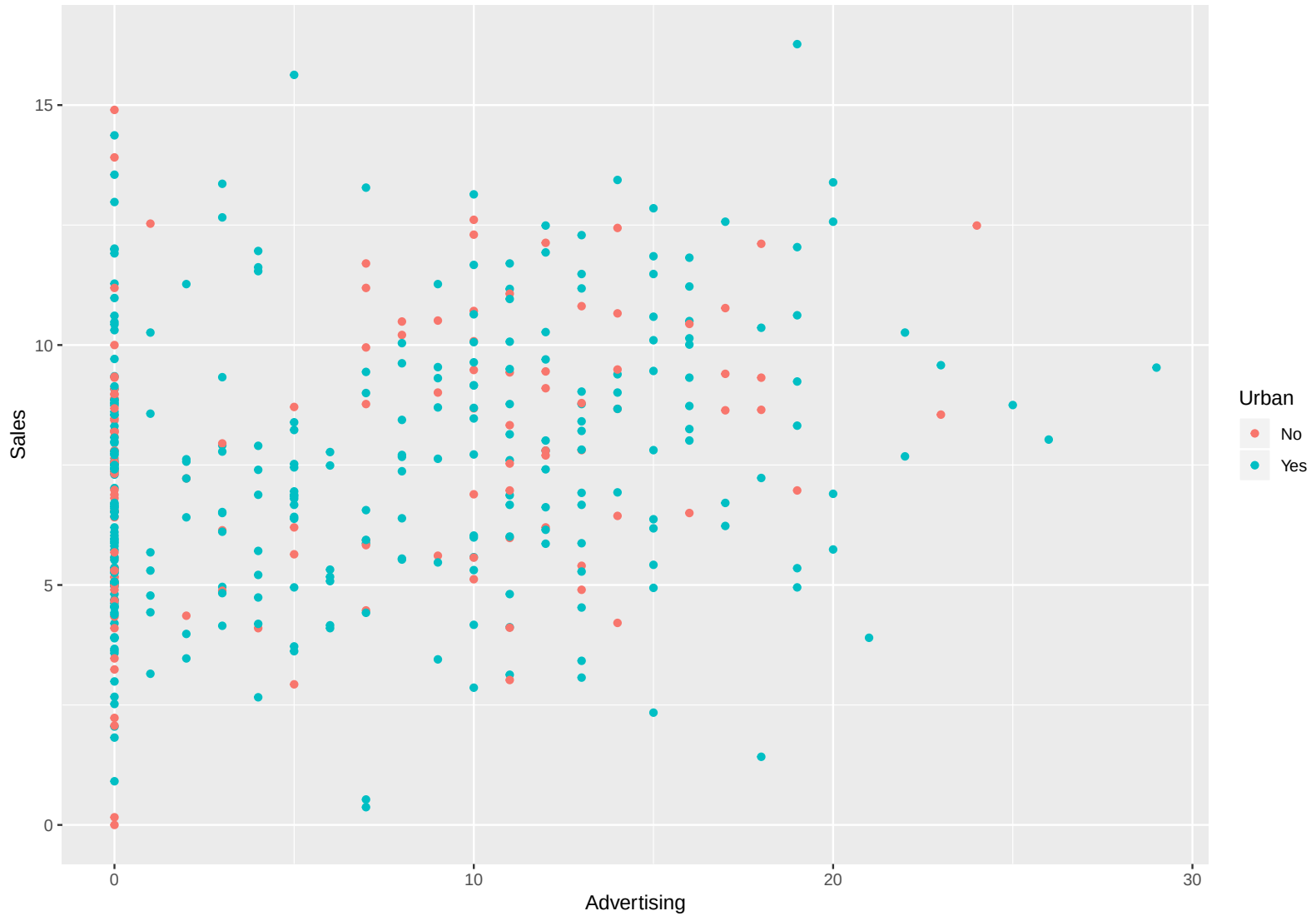
1. 그림의 목적을 정하고
2. X (explanatory variable - 설명 변수)의 개수와 각 변수의 중요도와 순서
3. Y (dependent variable)를 명시
4. ggplot 객체를 rendering

Carseat

```
str(Carseats)
```

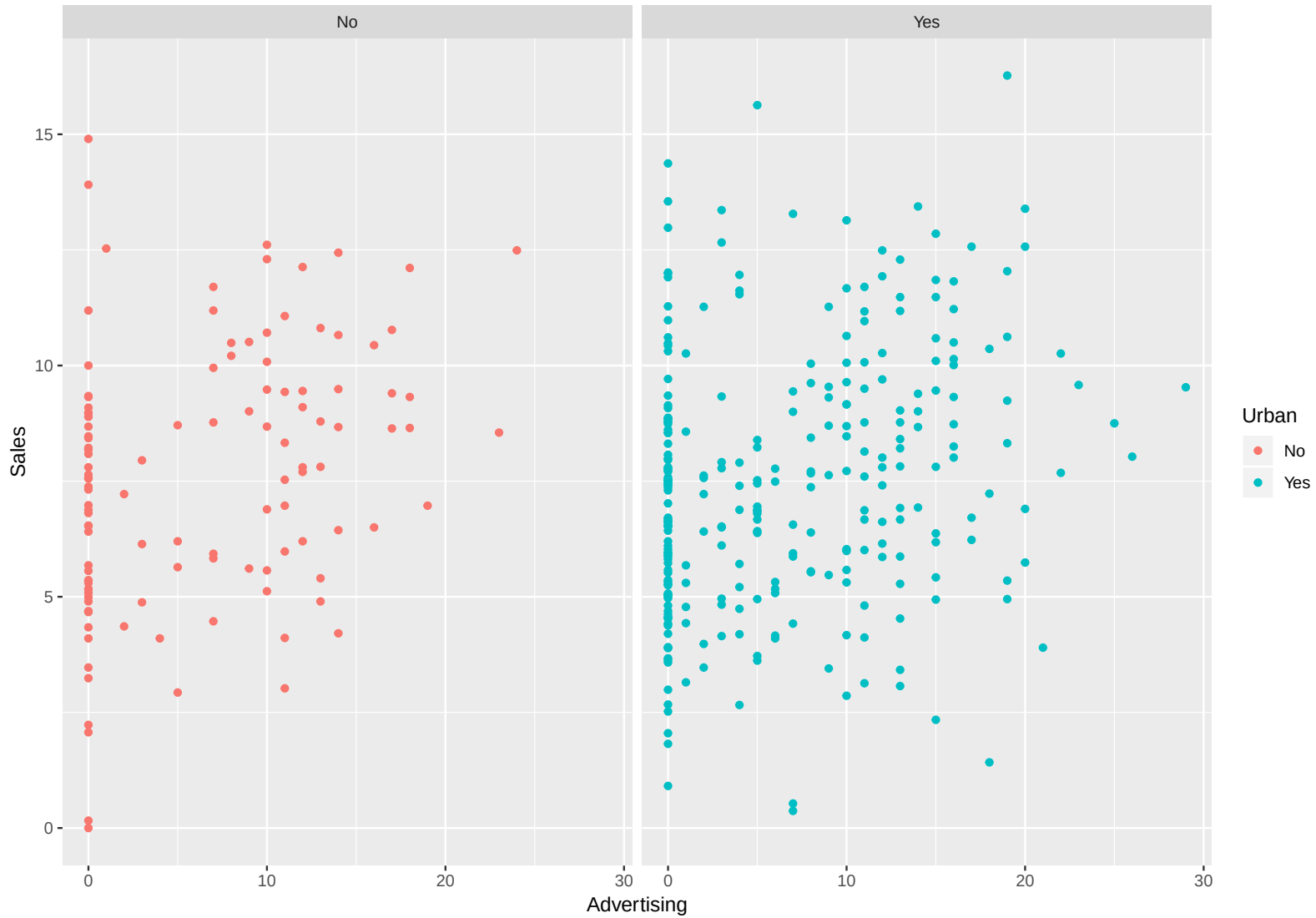
```
## 'data.frame':    400 obs. of  15 variables:
##  $ Sales       : num  0.37 0 6.67 6.39 5.01 9.53 3.58 5.4 6.53 2.93 ...
##  $ CompPrice   : num  147 139 156 131 159 175 142 149 154 143 ...
##  $ Income      : num  58 24 42 21 69 65 109 73 30 21 ...
##  $ Advertising : num  7 0 13 8 0 29 0 13 0 5 ...
##  $ Population  : num  100 358 170 220 438 419 111 381 122 81 ...
##  $ Price       : num  191 185 173 171 166 166 164 163 162 160 ...
##  $ ShelveLoc   : Factor w/ 3 levels "Bad","Good","Medium": 1 3 2 2 3 3 2 1 3 3 ...
##  $ Age         : num  27 79 74 29 46 53 72 26 57 67 ...
##  $ Education   : num  15 15 14 14 17 12 12 11 17 12 ...
##  $ Urban       : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 1 1 1 ...
##  $ US          : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 2 1 2 1 2 ...
##  $ Revenue     : num  0.37 0 6.67 6.39 5.01 9.53 3.58 5.4 6.53 2.93 ...
##  $ AdvPerCapita: num  0.07 0 0.0765 0.0364 0 ...
##  $ RevPerCapita: num  0.0037 0 0.0392 0.029 0.0114 ...
##  $ AgeClass    : chr  "non-Silver" "Silver" "Silver" "non-Silver" ...
```

```r
a <- ggplot(data = Carseats, aes(x = Advertising, y = Sales)) +
  geom_point(aes(color = Urban))
print(a)
```



Amazingly Additive

```
# Add features to existing ggplot object!
a <- a + facet_wrap(~ Urban)
print(a)
```



- 2 slides above…

```
a <- ggplot(data = Carseats, aes(x = Advertising, y = Sales)) +
  geom_point(aes(color = Urban))
```

- 1 slide above…

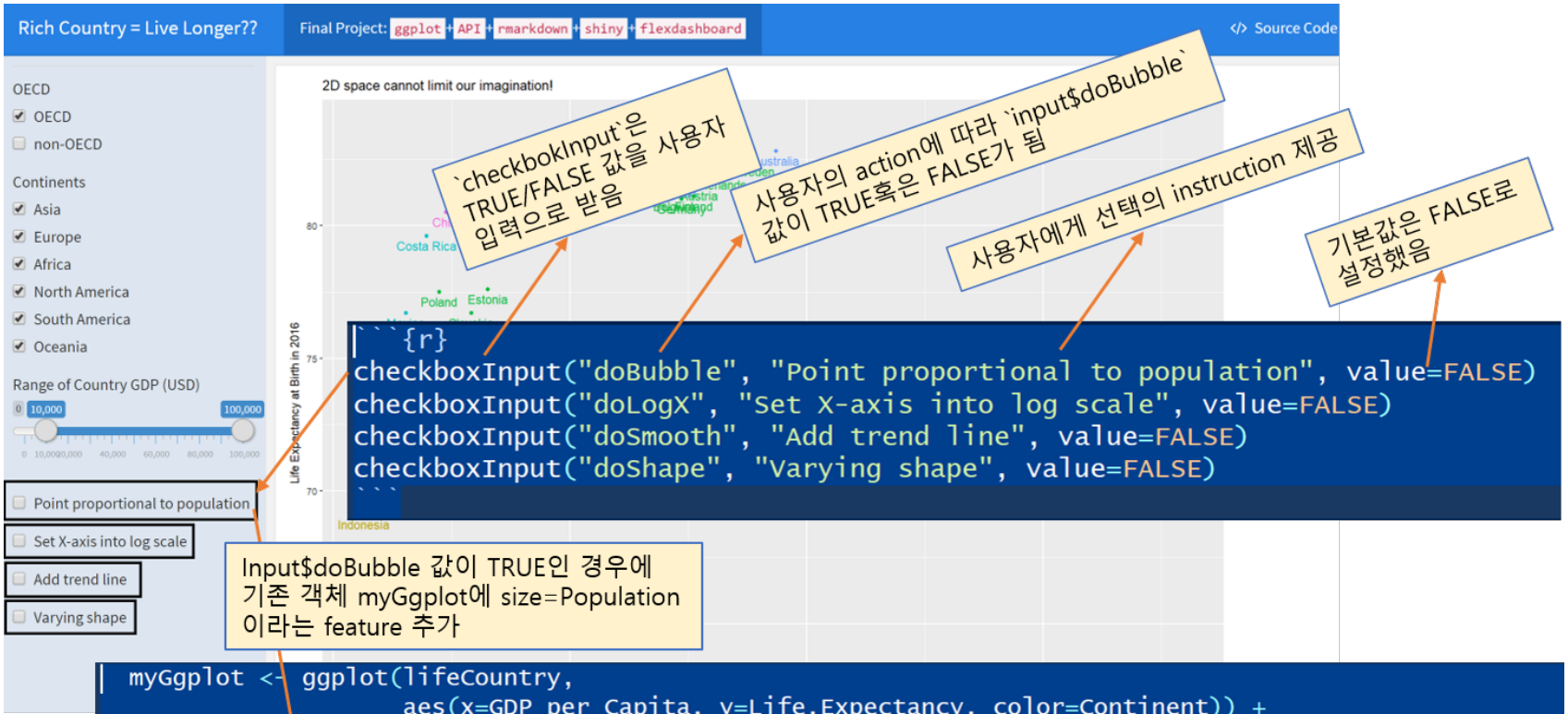
```
a <- a + facet_wrap(~ Urban)
```

- 두 가지 기능을 합하고 사용자가 선택할 수 있게 한다.
- May allow user to choose as well!
- Why markdown?

```
doAgeFacet <- TRUE
a <- ggplot(data = Carseats, aes(x = Advertising, y = Sales)) +
  geom_point(aes(color = Urban))
if (doAgeFacet) {
  a <- a + facet_wrap(~ Urban)
}
print(a)
```

```
doAgeFacet <- FALSE
a <- ggplot(data = Carseats, aes(x = Advertising, y = Sales)) +
  geom_point(aes(color = Urban))
if (doAgeFacet) {
  a <- a + facet_wrap(~ Urban)
}
print(a)
```

Example: M41-longevity



Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +                    Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + geom_map(aes(map_id = state), map = map) + **expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))
```

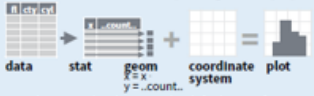
l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).



Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) **x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) **sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

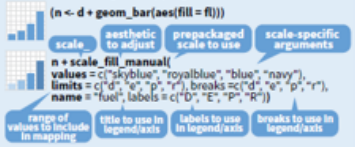
e + stat_summary(fun.data = "mean_cl_boot")

h + stat_summary_bin(fun.y = "mean", geom = "bar")

e + stat_unique()

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



GENERAL PURPOSE SCALES

Use with most aesthetics

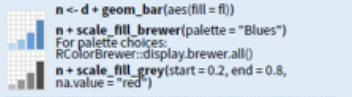
scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()) - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks") - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

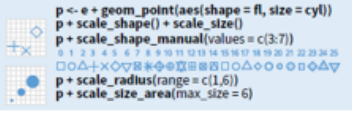


n <- d + geom_bar(aes(fill = fl))
n + scale_fill_brewer(palette = "Blues")
For palette choices:
RColorBrewer::display.brewer.all()
n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))
o + scale_fill_distiller(palette = "Blues")
o + scale_fill_gradient(low="red", high="yellow")
o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)
o + scale_fill_gradientn(colours=topo.colors(6))
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

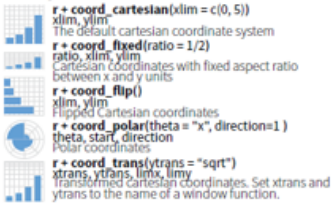
SHAPE AND SIZE SCALES



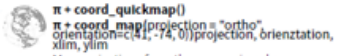
p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)

Coordinate Systems

r <- d + geom_bar()



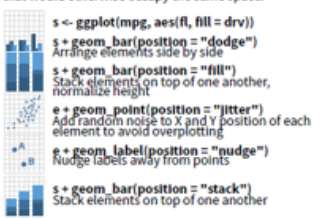
r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system
r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units
r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates
r + coord_polar(theta = "x", direction=1)
theta, start, direction
Polar coordinates
r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed Cartesian coordinates. Set xtrans and ytrans to the name of a window function.



π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0))projection, orienztation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

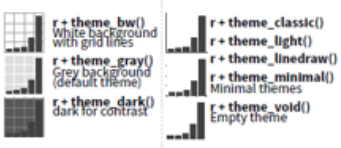
Position adjustments determine how to arrange geoms that would otherwise occupy the same space.



s <- ggplot(mpg, aes(fl, fill = drv))
s + geom_bar(position = "dodge")
Arrange elements side by side
s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height
e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting
e + geom_label(position = "nudge")
Nudge labels away from points
s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment can be recast as a function with manual **width** and **height** arguments
s + geom_bar(position = position_dodge(width = 1))

Themes



r + theme_bw()
White background with grid lines
r + theme_gray()
Grey background (default theme)
r + theme_dark()
dark for contrast
r + theme_classic()
r + theme_light()
r + theme_linedraw()
r + theme_minimal()
Minimal themes
r + theme_void()
Empty theme

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) **+ geom_point()**

t + facet_grid(. ~ fl)
facet into columns based on fl
t + facet_grid(year ~ .)
facet into rows based on year
t + facet_grid(year ~ fl)
facet into both rows and columns
t + facet_wrap(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(drv ~ fl, **scales = "free"**)
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, **labeller = label_both**)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(fl ~ ., **labeller = label_bquote**(alpha ^ .(fl)))

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

t + facet_grid(. ~ fl, **labeller = label_parsed**)

c	d	e	p	r

Labels

t + labs(x = "New x axis label", y = "New y axis label", **title** ="Add a title above the plot", **subtitle** = "Add a subtitle below title", **caption** = "Add a caption below plot", <AES> = "New <AES> legend title")

Use scale functions to update legend labels

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place | manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"
n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)
t + coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))
With clipping (removes unseen data points)
t + xlim(0, 100) **+ ylim**(10, 20)
t + scale_x_continuous(limits = c(0, 100)) + **scale_y_continuous**(limits = c(0, 100))

RStudio

More References

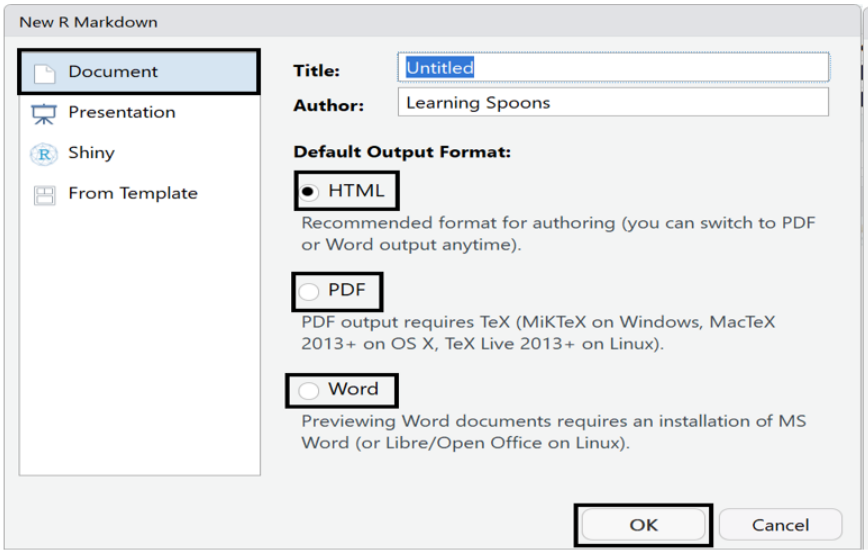
1. `M23-More on ggplot2` - 다음주!
2. `M24-ggplot2-50examples` - ggplot의 갤러리 (50개 예제)
3. `M34-Happy B-day, ggplot`
   - ggplot의 10주년을 기념하는 article
   - 크롤링과 구글 번역 API를 이용하여 자동 번역 및 문서화
   - https://qz.com/1007328/
     all-hail-ggplot2-the-code-powering-all-those-excellent-charts-is-10-years-old/
4. `M6X References/books/ggplot2.pdf`
   - 무료 베포된 ggplot관련 영문 서적
   - 번역본도 출간되어 있음

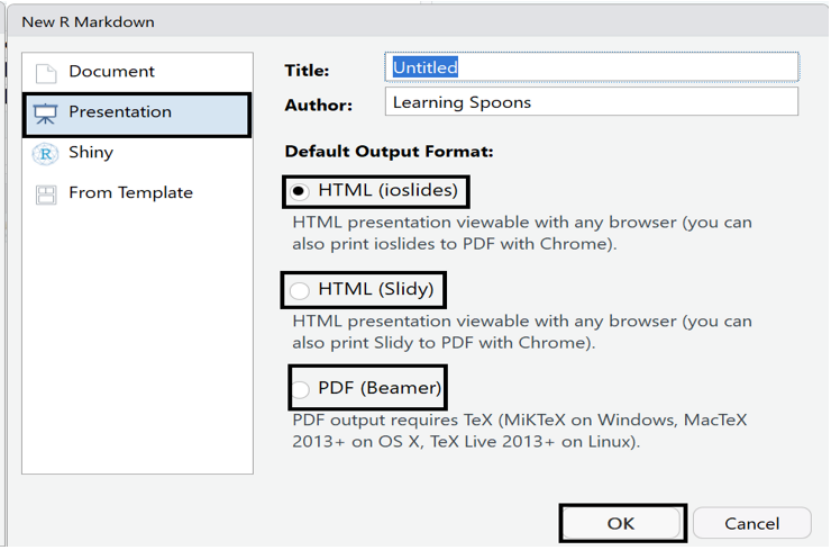
Part 4. 문서화 - ”rmarkdown”

New Generation of Computer Programming
Literate Programming

- Motivation
  1. Workflow 관점
     - old: 계산 -> 표, 그림 정리 -> 워드프로세서에 정리 -> 다시 계산 -> 표, 그림 정리 -> 워드프로세서 -> 반복 반복
     - new: **`rmarkdown`**
  2. Display 관점
     - old: 에디터, 콘솔, 아웃풋 윈도우, 엑셀, 워드프로세서…
     - new: **`rmarkdown`** under Rstudio

파일 -> 새파일 -> R Markdown



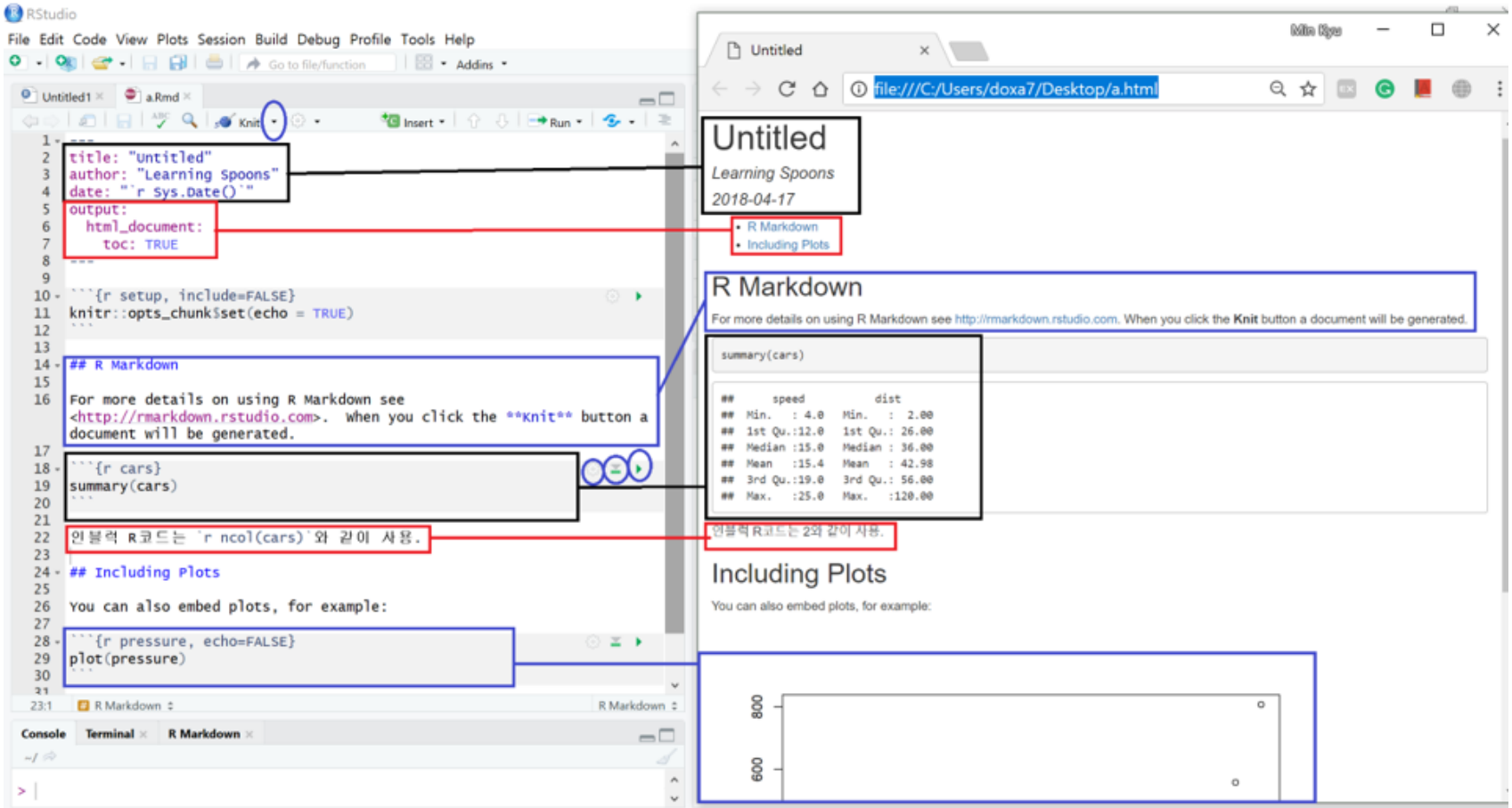


- `.html`
  - interactive feature 가능
- `.pdf`
  - professional 문서
  - 조판을 위해 texlive가 설치되어 있어야 함 (slide 76)
  - 한글 사용은 별도의 템플릿 사용 (`M25-.pdf`)
- `.docx`
  - MS office 프로그램 없이도 제작 가능

- ioslide, slidy
  - interactive feature 가능
  - `.html` 형식의 슬라이드
  - slidy 템플릿 (`M27-slidy`)
- beamer (`.pdf`)
  - Scientific presentation
  - 조판을 위해 texlive가 설치되어 있어야 함 (slide 76)
  - 한글 사용은 별도의 템플릿 사용 (`M26-beamer`)
  - **성실한** 선생님들의 강의노트

.html

- 시작: File -> New File -> R Markdown -> Document/Presentation + 포맷 선택
- 렌더링: Knit -> Knit to XXX



.pdf (템플릿: M25-.pdf)

RStudio Source Editor

rmd-pdf-template.Rmd

```
1  ---
2  title: "rmd pdf template "
3  author: "Learning Spoons"
4  date: "`r Sys.Date()`"
5  output:
6    pdf_document:
7      latex_engine: xelatex
8      highlight: haddock
9      keep_tex: true
10     # pandoc_args: [
11     #   "-V", "classoption=twocolumn"
12     # ]
13   smaller: true
14 mainfont: NanumGothic
15 classoption: a4paper
16 ---
17
18 ```{r setup, include=FALSE}
19 knitr::opts_chunk$set(echo = TRUE)
20 ```
21
22 ## R Markdown 한글
23
24 Code의 9,10,11 번째 라인의 pound sign(#)을 제거하고 위의 줄과 indent를
25 <http://rmarkdown.rstudio.com>.
26 When you click the **Knit** button a document will be generated.
27
28 ```{r cars}
29 summary(cars)
30 ```
31
32 ## Including Plots
33
34 ```{r pressure, echo=FALSE, fig.height = 3}
35 plot(pressure)
36 ```
37
38 ## Line Numbering
39
40 ```{#numCode .R .numberLines}
41 x <- 1:10
```

13:16 rmd pdf template

rmd-pdf-template.pdf

페이지: 1 / 1 찾기:

rmd pdf template

Learning Spoons

2018-06-30

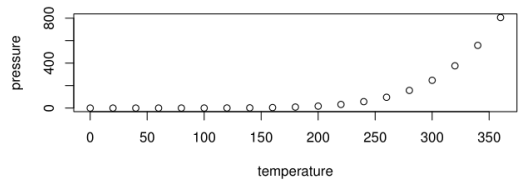
R Markdown 한글

Code의 9,10,11 번째 라인의 pound sign(#)을 제거하고 위의 줄과 indent를 맞추면 2 컬럼의 문서가 render 됩니다.
http://rmarkdown.rstudio.com.
When you click the **Knit** button a document will be generated.

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots



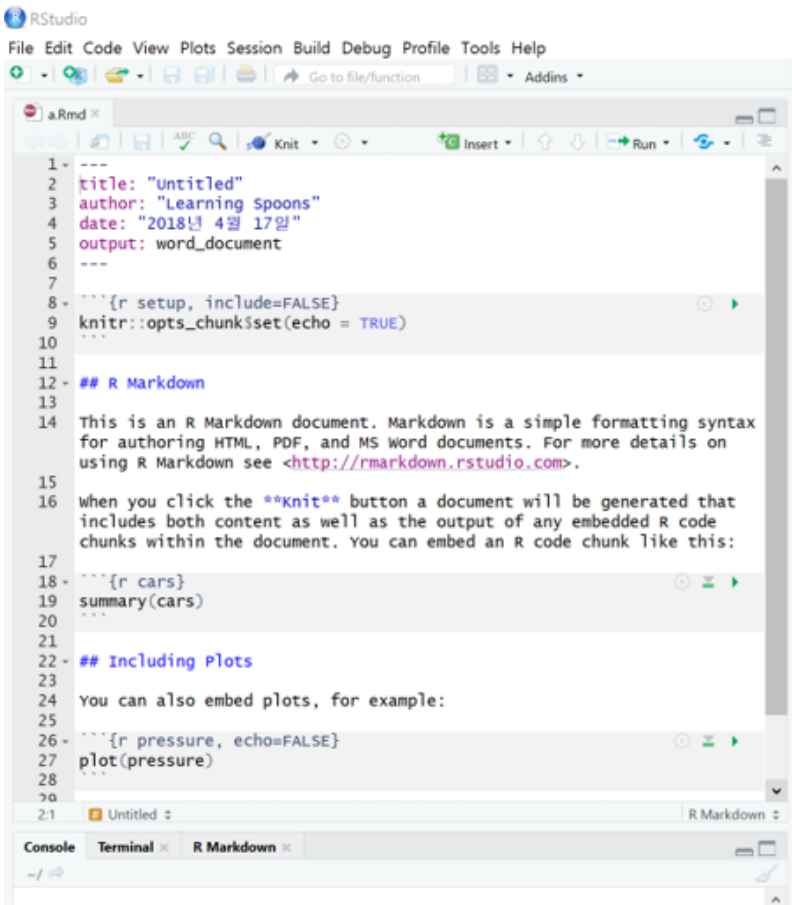
Line Numbering

```
1 x <- 1:10
2 y <- x^2
3 plot(x,y)
4 "Hello"
```

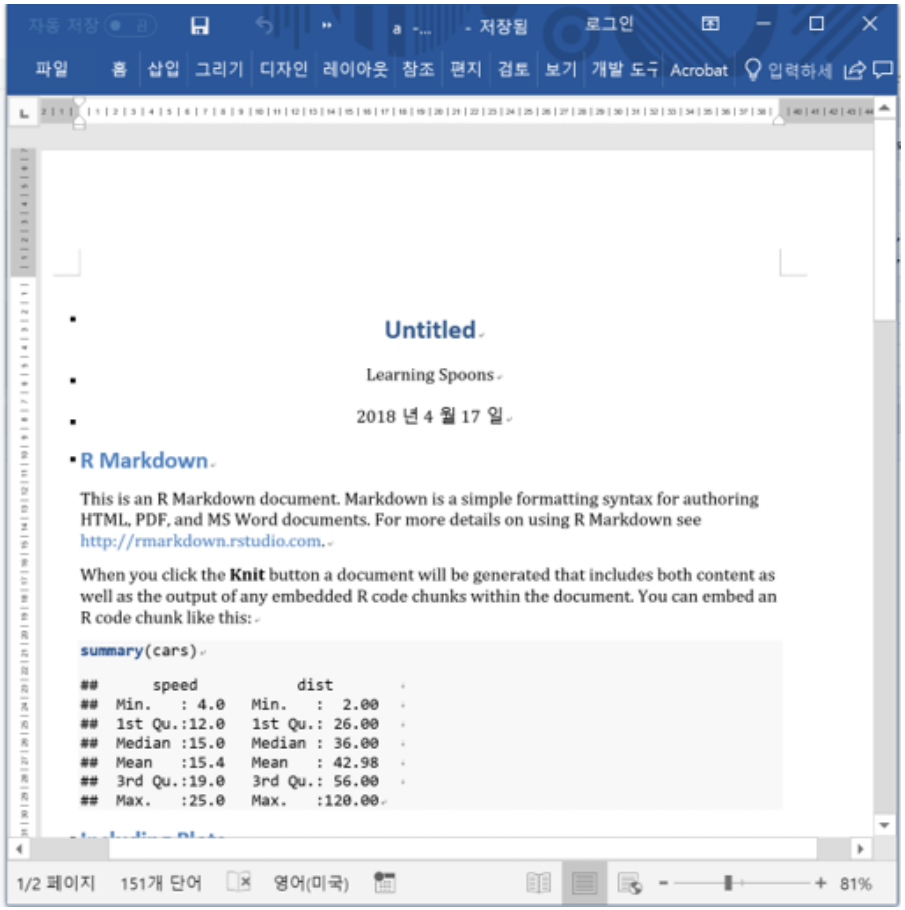
```
##  [1]  1  2  3  4  5  6  7  8  9 10
## [1] "Hello"
```

1

docx


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
a.Rmd
Knit
Insert
Run
---
title: "Untitled"
author: "Learning Spoons"
date: "2018년 4월 17일"
output: word_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
```{r cars}
summary(cars)
```
Including Plots
You can also embed plots, for example:
```{r pressure, echo=FALSE}
plot(pressure)
```
2:1
Untitled
R Markdown
Console
Terminal
R Markdown


저장됨
로그인
파일 홈 삽입 그리기 디자인 레이아웃 참조 편지 검토 보기 개발 도구 Acrobat 입력하세
Untitled
Learning Spoons
2018 년 4 월 17 일
R Markdown
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.
When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
summary(cars)
speed dist
Min. : 4.0 Min. : 2.00
1st Qu.:12.0 1st Qu.: 26.00
Median :15.0 Median : 36.00
Mean :15.4 Mean : 42.98
3rd Qu.:19.0 3rd Qu.: 56.00
Max. :25.0 Max. :120.00
1/2 페이지 151개 단어 영어(미국) 81%

ioslides

```
---
title: "Untitled"
author: "Learning Spoons"
date: "2018년 4월 17일"
output: ioslides_presentation
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```

R Markdown

This is an R Markdown presentation.
<http://rmarkdown.rstudio.com>.

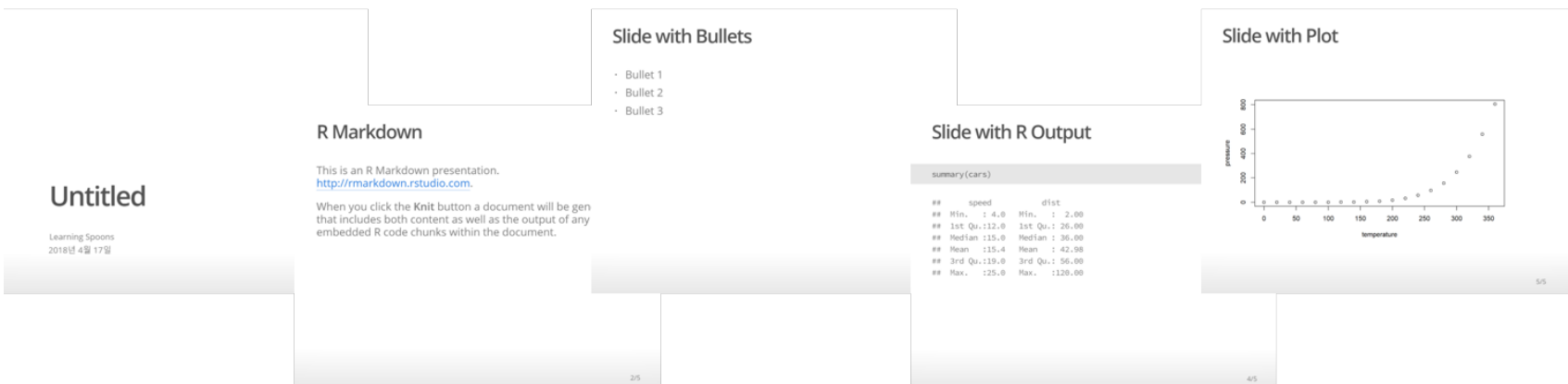
When you click the **Knit** button a document will be generated
that includes both content as well as the output of any embedded
R code chunks within the document.

Slide with Bullets

- Bullet 1
- Bullet 2
- Bullet 3

Slide with R Output

```{r cars, echo = TRUE}
summary(cars)
```

Slide with Plot
```



slidy (템플릿: M27-slidy)

```
---
title: "Untitled"
author: "Learning Spoons"
date: "`r Sys.Date()`"
output:
  slidy_presentation: default
  ioslides_presentation: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
```

R Markdown

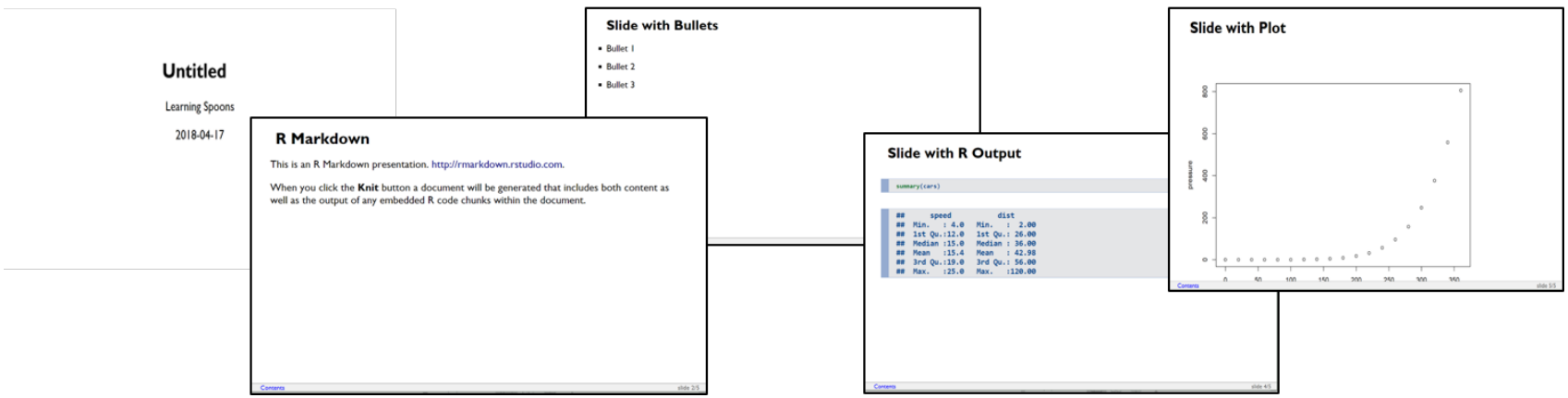
This is an R Markdown presentation.
<http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated
that includes both content as well as the output of any embedded
R code chunks within the document.

Slide with Bullets

- Bullet 1
- Bullet 2
- Bullet 3

Slide with R Output

```{r cars, echo = TRUE}
summary(cars)
```
```



beamer (템플릿: M26-beamer)

rmd-beamer-template.Rmd

```
---
title: "rmd - beamer - template "
author: "LearningSpoonsR "
date: "`r Sys.Date()`"
fontsize: 9pt
output:
  beamer_presentation:
    theme: "Singapore"
    # For code line number, choose among
    # ["Antibes", "Montpellier", "Singapore", "Szeged"]
    colortheme: "beaver"
    # For Singapore - ["beaver": print-friendly, "beetle": grey]
    latex_engine: xelatex
    # keep_tex: true
    # template: myTemplate.tex
    includes:
      in_header: myRmdBeamerStyle/latex-topmatter.tex
classoption: t
mainfont: NanumGothic
---

```{r setup, include=FALSE}
library(rmarkdown)
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(background = '718CBA')
```

beamer@RMarkdown

- 이 템플릿은 한글 및 twocolumn layout으로 beamer 문서를 제작할 수 있는 템플릿입니다. \br
- RMarkdown (`M2X-rmd`를 참조) \br
- pdf 조판을 위한 texlive 엔진 (`M2X-rmd`를 참조) \br
- slide형태의 pdf를 만드는 beamer 패키지 (R 패키지가 아니라 tex 패키지) \br
- `tex`설치등의 작업이 되있어야 이 템플릿을 사용할 수 있습니다. \br
- 하위 폴더인 `myRmdBeamerStyle`이 있고, 폴더안에는 `latex-topmatter.tex`와 `markdown_cust

Design

1. 헤더 부분의 아래 명령어를 검색해서 바꾸시면 다른 색상
  + `theme: "Singapore"`
  + `colortheme: "beaver"`
  + `knitr::opts_chunk$set(background = '718CBA')`

2단 구성

\lc

1. Left Column
- 이 템플릿은 2단 구성이 가능합니다. \br
- `M5X-tidyr`의 강의노트가 beamer로 만든 좋은 예제입니다

\rc

2. Right Column

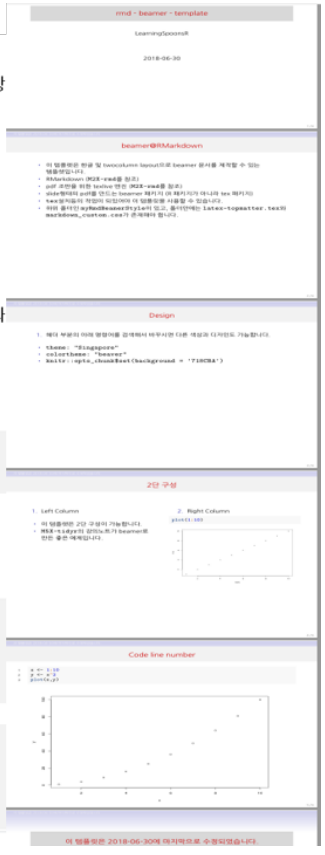
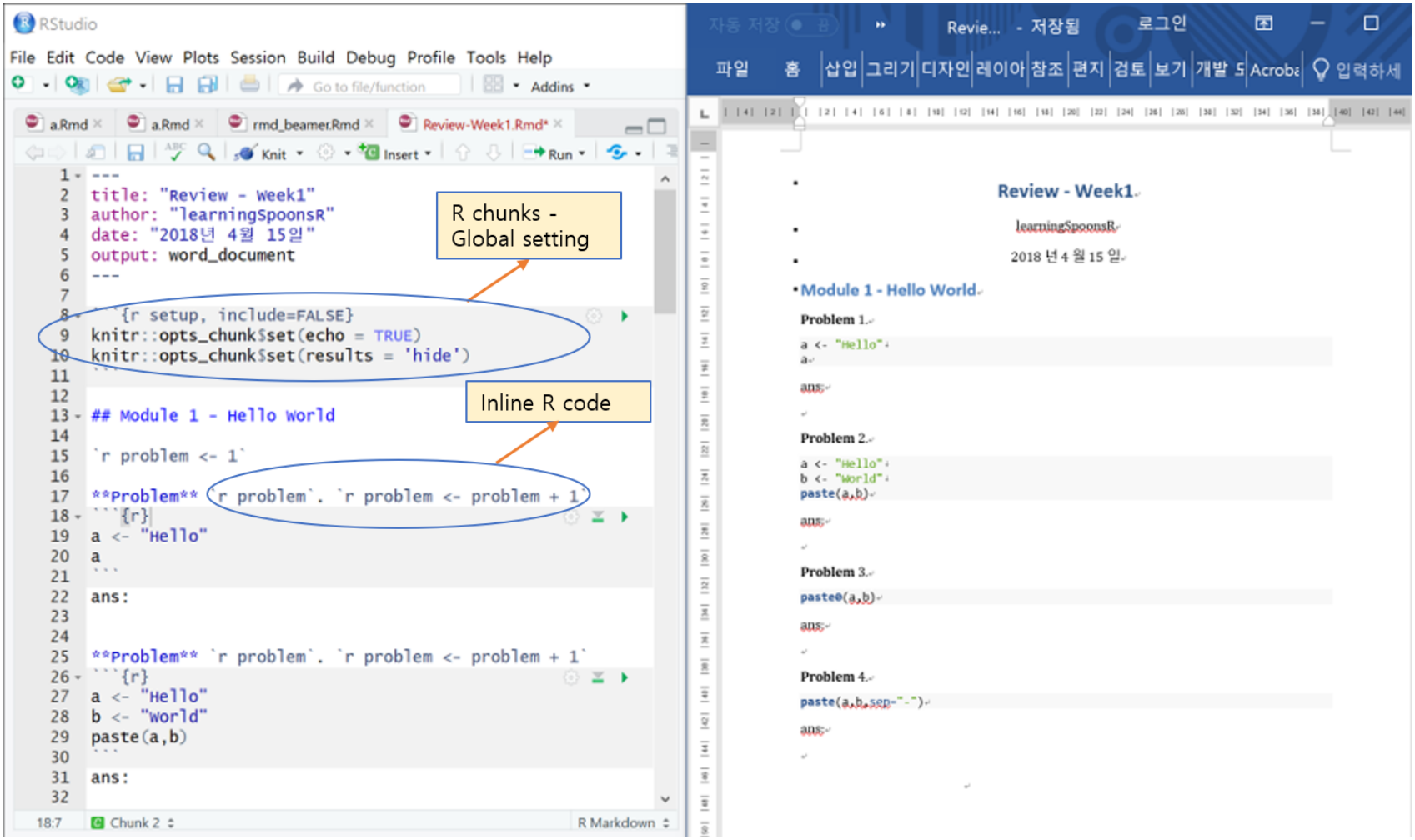
```{r}
plot(1:10)
```

\ec

Code line number

```{#numCode .R .numberLines}
x <- 1:10
y <- x^2
plot(x,y)
```

```{r results='asis', echo=FALSE}
x <- 1:10
y <- x^2
plot(x,y)
```

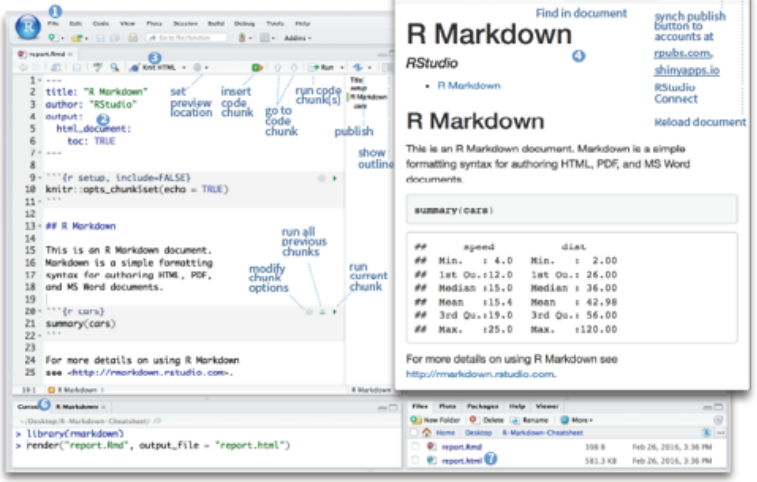
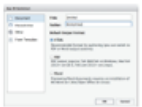
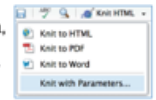

```

.Rmd 파일의 구성

1. Header
    - `yaml`, `pandoc`
    - title, author, date 입력
    - 파일 포맷 컨트롤
2. R chunks
    - R Language 입력
    - 오른쪽에 3개의 단추 (setting, run all chunks above, run this chunk)
    - `knitr::opts_chunks$set()`으로 global setting 가능
3. MD chunks
    - Human Language 입력
    - Inline R 사용 가능

Example - 1주차 숙제



Problem에 1을 더해 자동으로 문제번호 컨트롤

```
**Problem** `r problem`. `r problem <- problem + 1`

```{r}
grep("ui", font)
```

```{r, echo=FALSE}
ans0 <- "ANS: The function grep somehow returns 0 or 1, which is equivalent to TRUE or FALSE!"
ans0
```

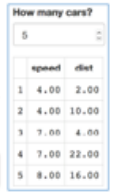
ans:
```

Problem 3.

```
grep("ui", font)

## [1] 1

## [1] "ANS: The function grep somehow returns 0 or 1, which is equivalent to
 TRUE or FALSE!"
```

ans:

R Markdown : : CHEAT SHEET

What is R Markdown?

.Rmd files - An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

Reproducible Research - At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.

Dynamic Documents - You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.



Workflow



1. **Open a new .Rmd file** at File ▶ New File ▶ R Markdown. Use the wizard that opens to pre-populate the file with a template
2. **Write document** by editing template
3. **Knit document to create report**; use knit button or **render()** to knit
4. **Preview Output** in IDE window
5. **Publish** (optional) to web server
6. **Examine build log** in R Markdown console
7. **Use output file** that is saved along side .Rmd

render

Use rmarkdown::**render()** to render/knit at cmd line. Important args:

input - file to render	**output_options** - List of render options (as in YAML)	**output_file**	**params** - list of params to use	**envir** - environment to evaluate code chunks in	**encoding** - of input file
output_format		**output_dir**			

Embed code with knitr syntax

INLINE CODE
Insert with `` `r <code>` ``. Results appear as text without code.
Built with `` `r getRversion()` `` ➡ Built with 3.2.3

CODE CHUNKS
One or more lines surrounded with ```` ```{r} ```` and ```` ``` ````. Place chunk options within curly braces, after r. Insert with

```
```{r echo=TRUE}
getRversion()
```
```

GLOBAL OPTIONS
Set with **knitr::opts_chunk$set()**, e.g.

```
```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

IMPORTANT CHUNK OPTIONS

cache - cache results for future knits (default = FALSE)
cache.path - directory to save cached results in (default = "cache/")
child - file(s) to knit and then include (default = NULL)
collapse - collapse all output into single block (default = FALSE)
comment - prefix for each line of results (default = '##')
dependson - chunk dependencies for caching (default = NULL)
echo - Display code in output document (default = TRUE)
engine - code language used in chunk (default = 'R')
error - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)
eval - Run code in chunk (default = TRUE)
fig.align - 'left', 'right', or 'center' (default = 'default')
fig.cap - figure caption as character string (default = NULL)
fig.height, fig.width - Dimensions of plots in inches
highlight - highlight source code (default = TRUE)
include - Include chunk in doc after running (default = TRUE)
message - display code messages in document (default = TRUE)
results (default = 'markup')
'asis' - passthrough results
'hide' - do not display results
'hold' - put all results below all code
tidy - tidy code for display (default = FALSE)
warning - display code warnings in document (default = TRUE)

Options not listed above: K.options, aniopts, autodep, background, cache.comments, cache.lazy, cache.rebuild, cache.vars, dev, dev.args, dpi, engine.opts, engine.path, fig.asp, fig.env, fig.ext, fig.keep, fig.lp, fig.path, fig.pos, fig.process, fig.retina, fig.scap, fig.show, fig.showtext, fig.subcap, interval, out.extra, out.height, out.width, prompt, purl, ref.label, render, size, split, tidy.opts

.rmd Structure

YAML Header
Optional section of render (e.g. pandoc) options written as key:value pairs (YAML).
At start of file
Between lines of ---

Text
Narration formatted with markdown, mixed with:

Code Chunks
Chunks of embedded code. Each chunk:
Begins with ```` ```{r} ````
ends with ```` ``` ````
R Markdown will run the code and append the results to the doc.
It will use the location of the .Rmd file as the **working directory**

Parameters

Parameterize your documents to reuse with different inputs (e.g., data, values, etc.)

1. Add parameters - Create and set parameters in the header as sub-values of params

2. Call parameters - Call parameter values in code as params$<name>

3. Set parameters - Set values wth Knit with parameters or the params argument of render():

render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01")))

```
---
params:
  n: 100
  d: !r Sys.Date()
---
```

Today's data is `r params$d`



Interactive Documents

Turn your report into an interactive Shiny document in 4 steps

1. Add runtime: shiny to the YAML header.
2. Call Shiny input functions to embed input objects.
3. Call Shiny render functions to embed reactive output.
4. Render with rmarkdown::run or click Run Document in RStudio IDE

```
---
output: html_document
runtime: shiny
---

```{r, echo = FALSE}
numericInput("n",
  "How many cars?", 5)

renderTable({
  head(cars, input$n)
})
```
```



Embed a complete app into your document with **shiny::shinyAppDir()**

*NOTE: Your report will rendered as a Shiny app, which means you must choose an html output format, like **html_document**, and serve it with an active R Session.*

RStudio

Pandoc's Markdown

Write with syntax on the left to create effect on right (after render)

```
Plain text
End a line with two spaces
to start a new paragraph.
*italics* and **bold**
`verbatim code`
sub/superscript^2^~2~
~~strikethrough~~
escaped: \* \_ \\
endash: --, emdash: ---
equation: $A = \pi*r^{2}$
equation block:

$$E = mc^{2}$$

> block quote

# Header1 {#anchor}

## Header 2 {#css_id}

### Header 3 {.css_class}

#### Header 4

##### Header 5

###### Header 6

<!--Text comment-->

\textbf{Tex ignored in HTML}
<em>HTML ignored in pdf</em>

<http://www.rstudio.com>
[link](www.rstudio.com)
Jump to [Header 1](#anchor)
image:

![Caption](smallorb.png)

* unordered list
  + sub-item 1
  + sub-item 2
    - sub-sub-item 1

* item 2

  Continued (indent 4 spaces)

1. ordered list
2. item 2
   i) sub-item 1
      A. sub-sub-item 1

(@) A list whose numbering

continues after

(@) an interruption

Term 1

: Definition 1

| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

- slide bullet 1
- slide bullet 2

(>- to have bullets appear on click)

horizontal rule/slide break:

---

A footnote [^1]

[^1]: Here is the footnote.
```

Plain text
End a line with two spaces
to start a new paragraph.
italics and **bold**
`verbatim code`
sub/superscript^2~2~
~~strikethrough~~
escaped: * _ \
endash: –, emdash: —
equation: $A = \pi * r^2$
equation block:

$$E = mc^2$$

> block quote

Header1

Header 2

Header 3

Header 4

Header 5

Header 6

HTML ignored in pdfs

http://www.rstudio.com
link
Jump to Header 1
image:

Caption

- unordered list
  - sub-item 1
  - sub-item 2
    - sub-sub-item 1
- item 2

  Continued (indent 4 spaces)

1. ordered list
2. item 2
   i. sub-item 1
      A. sub-sub-item 1

1. A list whose numbering

continues after

2. an interruption

Term 1

Definition 1

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

- slide bullet 1
- slide bullet 2

(>- to have bullets appear on click)

horizontal rule/slide break:

A footnote [1]

1. Here is the footnote.[↩2]

Set render options with YAML

When you render, R Markdown

1. runs the R code, embeds results and text into .md file with knitr
2. then converts the .md file into the finished format with pandoc

Set a document's default output format in the YAML header:

```
---
output: html_document
---
# Body
```

output value	creates
html_document	html
pdf_document	pdf (requires Tex)
word_document	Microsoft Word (.docx)
odt_document	OpenDocument Text
rtf_document	Rich Text Format
md_document	Markdown
github_document	Github compatible markdown
ioslides_presentation	ioslides HTML slides
slidy_presentation	slidy HTML slides
beamer_presentation	Beamer pdf slides (requires Tex)

Customize output with sub-options (listed to the right):

Indent 2 spaces / Indent 4 spaces

```
---
output: html_document:
  code_folding: hide
  toc_float: TRUE
---
# Body
```

html tabsets

Use tablet css class to place sub-headers into tabs

```
# Tabset {.tabset .tabset-fade .tabset-pills}
## Tab 1
text 1
## Tab 2
text 2
### End tabset
```

Tabset

Tab 1 | Tab 2

text 1
End tabset

rmarkdown

sub-option	description	html	pdf	word	odt	rtf	md	github	ioslides	slidy	beamer
citation_package	The LaTeX package to process citations, natbib, biblatex or none		X				X				X
code_folding	Let readers to toggle the display of R code, "none", "hide", or "show"	X									
colortheme	Beamer color theme to use										X
css	CSS file to use to style document	X							X	X	
dev	Graphics device to use for figure output (e.g. "png")	X	X				X	X	X	X	X
duration	Add a countdown timer (in minutes) to footer of slides									X	
fig_caption	Should figures be rendered with captions?	X	X	X	X				X	X	X
fig_height, fig_width	Default figure height and width (in inches) for document	X	X	X	X	X	X	X	X	X	X
highlight	Syntax highlighting: "tango", "pygments", "kate","zenburn", "textmate"	X	X	X						X	X
includes	File of content to place in document (in_header, before_body, after_body)	X	X		X		X	X	X	X	X
incremental	Should bullets appear one at a time (on presenter mouse clicks)?								X	X	X
keep_md	Save a copy of .md file that contains knitr output	X		X	X	X			X	X	
keep_tex	Save a copy of .tex file that contains knitr output		X								X
latex_engine	Engine to render latex, "pdflatex", "xelatex", or "lualatex"		X								X
lib_dir	Directory of dependency files to use (Bootstrap, MathJax, etc.)	X							X	X	
mathjax	Set to local or a URL to use a local/URL version of MathJax to render equations	X							X	X	
md_extensions	Markdown extensions to add to default definition or R Markdown	X	X	X	X	X	X	X	X	X	X
number_sections	Add section numbering to headers	X	X								
pandoc_args	Additional arguments to pass to Pandoc	X	X	X	X	X	X	X	X	X	X
preserve_yaml	Preserve YAML front matter in final document?						X				
reference_docx	docx file whose styles should be copied when producing docx output			X							
self_contained	Embed dependencies into the doc	X							X	X	
slide_level	The lowest heading level that defines individual slides										X
smaller	Use the smaller font size in the presentation?								X		
smart	Convert straight quotes to curly, dashes to em-dashes, ... to ellipses, etc.	X							X	X	
template	Pandoc template to use when rendering file quarterly_report.html).	X	X		X					X	X
theme	Bootswatch or Beamer theme to use for page	X									X
toc	Add a table of contents at start of document	X	X	X		X	X	X			X
toc_depth	The lowest level of headings to add to table of contents	X	X	X		X	X	X			
toc_float	Float the table of contents to the left of the main content	X									

Create a Reusable Template

1. **Create a new package** with a inst/rmarkdown/templates directory
2. In the directory, **Place a folder** that contains:
   template.yaml (see below)
   skeleton.Rmd (contents of the template)
   any supporting files
3. **Install the package**
4. **Access template** in wizard at File ▶ New File ▶ R Markdown

template.yaml

```
---
name: My Template
---
```


Table Suggestions

Several functions format R data into tables


```
data <- faithful[1:4, ]
```

```{r results = 'asis'}
knitr::kable(data, caption = "Table with kable")
```

```{r results = "asis"}
print(xtable::xtable(data, caption = "Table with xtable"),
      type = "html", html.table.attributes = "border=0")
```

```{r results = "asis"}
stargazer::stargazer(data, type = "html", title = "Table with stargazer")
```

Learn more in the stargazer, xtable, and knitr packages.

Citations and Bibliographies

Create citations with .bib, .bibtex, .copac, .enl, .json, .medline, .mods, .ris, .wos, and .xml files

1. **Set bibliography file** and CSL 1.0 Style file (optional) in the YAML header
2. **Use citation keys in text**

```
---
bibliography: refs.bib
csl: style.csl
---
```

Smith cited [@smith04].
Smith cited without author [-@smith04].
@smith04 cited in line.

3. **Render.** Bibliography will be added to end of document

Smith cited (Joe Smith 2004).
Smith cited without author (2004).
Joe Smith (2004) cited in line.

R 마크다운 컨닝쪽지

추가 학습 정보 rmarkdown.rstudio.com

rmarkdown 0.2.50 최종갱신일: 8/14

RStudio

1. 작업흐름 R 마크다운은 R로 재현가능하고, 동적인 보고서를 작성하는 서식이다. R 마크다운을 사용해서 R 코드와 실행결과를 발표자료, pdf, html, 워드 문서 등에 삽입할 수 있다. 보고서를 작성하려면:

i. 파일열기 - .Rmd 확장자를 갖는 파일을 연다.

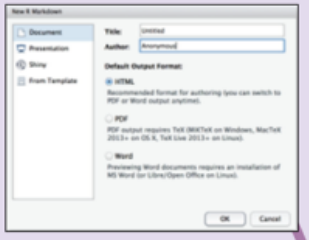
ii. 작성하기 - 본문을 작성하기 쉬운 R 마크다운 구문을 사용해서 작성한다.

iii. 내장하기 - 리포트에 포함될 출력결과를 생성하는 R 코드를 내장한다.

iv. 렌더링(Render) - R 코드를 출력형식으로 처환하고 보고서를 발표자료, pdf, html, MS 워드 파일 형식으로 변환한다.

2. 파일 열기 .Rmd 확장자를 갖는 텍스트 파일로 저장해서 시작하거나, Studio Rmd 템플릿을 열어 시작한다.

- 메뉴막대에서, 다음순으로 클릭한다. **File ▶ New File ▶ R Markdown...**
- 윈도우가 열리면, .Rmd 파일로 작성하려는 출력유형을 선택한다.
- 라디오 버튼으로 출력형식을 선택한다 (나중에 출력형식은 변경할 수 있다)
- OK 버튼을 클릭한다.



3. 마크다운 다음으로, 일반 텍스트로 보고서를 작성한다. 마크다운 구문을 사용해서 최종 보고서에 적용할 텍스트 서식을 기술한다.

입력 구문

```
일반 텍스트
새로운 단락을 시작하려면 줄 마지막을 공백 2개로 끝낸다.
*기울인 글씨* and _기울인 글씨_
**굵은 글씨** and __굵은 글씨__
윗첨자^2^
~~취소선~~
[링크](www.rstudio.com)

# 제목 1

## 제목 2

### 제목 3

#### 제목 4

##### 제목 5

###### 제목 6

N자 크기 대시 부호: --
m자 크기 대시 부호: ---
생략: ...
즉시 처리하는 수식: $A = \pi*r^{2}$
이미지: ![](path/to/smallorb.png)

수평선 (혹은 슬라이드 멈춤):

***

> 인용 블록

* 순서없는 목록
* 항목 2
  + 하위 항목 1
  + 하위 항목 2

1. 순서있는 목록
2. 항목 2
  + 하위 항목 1
  + 하위 항목 2

표 제목 | 두번째 제목
---------- | ----------
표 칸 | 칸 2
칸 3 | 칸 4
```

출력 결과

일반 텍스트
새로운 단락을 시작하려면 줄 마지막을 공백 2개로 끝낸다.
기울인 글씨 and *기울인 글씨*
굵은 글씨 and **굵은 글씨**
윗첨자^2
~~취소선~~
링크

제목 1

제목 2

제목 3

제목 4

제목 5

제목 6

N자 크기 대시 부호: –
m자 크기 대시 부호: —
생략: …
즉시 처리하는 수식: $A = \pi * r^2$
이미지:

수평선 (혹은 슬라이드 멈춤):

> 인용 블록

* 순서없는 목록
* 항목 2
  + 하위 항목 1
  + 하위 항목 2

1. 순서있는 목록
2. 항목 2
  + 하위 항목 1
  + 하위 항목 2

표 제목	두번째 제목
표 칸	칸 2
칸 3	칸 4

4. 출력형식 선정 R 마크다운 파일에서 생성할 문서 유형을 기술하는 YAML 헤더정보를 작성한다.

YAML

YAML 헤더는 키(key) 집합: 파일 시작지점에 나오는 키-값 쌍. 헤더 시작과 끝은 3개 대쉬를 갖는 라인 (- - -)

```
---
title: "xwMOOC 보고서"
author: "무명씨"
output: html_document
---

보고서의 시작지점으로, YAML 헤더에 메타데이터 정보가 저장되어 있다.
```

RStudio 템플릿이 자동으로 YAML 헤더 정보를 작성해 준다.

출력값이 .Rmd 파일에서 어떤 형식 파일을 생성할 것인지 결정한다(단계 6)

- **output: html_document** ..... html 파일 (웹페이지)
- **output: pdf_document** ..... pdf 문서
- **output: word_document** ..... MS 워드문서 .docx
- **output: beamer_presentation** ..... Beamer 발표자료 (pdf)
- **output: ioslides_presentation** ..... 맥 발표자료 (html)

5. 코드내장하기 knitr 구문을 사용해서 R 코드를 보고서에 내장한다. R이 코드를 실행하고, 보고서를 렌더링할 때 결과를 포함시킨다.

인라인 코드

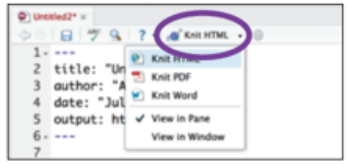
r 코드를 백틱(`)으로 감싼다.
R이 인라인 코드를 실행된 결과로 대체한다.

```
2 더하기 2는 `r 2+2`와 같다.
```

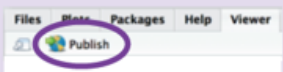
Two plus two equals 4.

코드 덩어리

R코드 덩어리를 ```{r} 으로 시작하고, ``` 으로 마무리한다.

```
실행결과는 다음과 같다
```{r}
dim(iris)
```
```

Here's some code

```
dim(iris)
```

```
## [1] 150   5
```

화면 출력 선택옵션

knitr 선택옵션을 사용해서 코드 덩어리 출력 스타일을 적용한다.
코드 상단 괄호 내부에 선택옵션을 지정한다.

```
Here's some code
```{r eval=FALSE}
dim(iris)
```
```

Here's some code

```
Here's some code
```{r echo=FALSE}
dim(iris)
```
```

Here's some code

```
## [1] 150   5
```

선택옵션	기본설정	효과
eval	TRUE	코드를 평가하고 실행결과를 포함한다.
echo	TRUE	실행결과와 함께 코드를 출력한다.
warning	TRUE	경고메시지를 출력한다.
error	FALSE	오류메시지를 출력한다.
message	TRUE	메시지를 출력한다.
tidy	FALSE	깔끔한 방식으로 코드 형태를 변형한다.
results	"markup"	"markup", "asis", "hold", "hide"
cache	FALSE	결과값을 캐쉬해서 향후 실행시 건너뛰게 설정한다.
comment	"##"	주석문자로 출력결과에 서두를 붙인다.
fig.width	7	덩어리로 생성되는 그래프에 대한 폭을 인치로 지정한다.
fig.height	7	덩어리로 생성되는 그래프에 대한 높이을 인치로 지정한다.

보다 자세한 사항은 웹사이트를 참조: yihui.name/knitr/

6. 렌더링 최종보고서를 생성하는데 .Rmd 파일을 사용하여 청사진을 제작한다.

두가지 방식으로 보고서를 렌더링한다.

1. **rmarkdown::render("<파일 경로>")** 명령어를 실행한다.
2. RStudio 스크립트 작성창 상단에 **knit HTML** 버튼을 클릭한다.



렌더링 명령을 실행시키면, R은 다음을 수행한다

- 내장된 코드 덩어리를 각각 실행시키고, 실행결과를 보고서에 삽입한다.
- 출력 파일형식에 맞춰 신규 보고서를 생성한다.
- 미리보기로 뷰어창에 출력파일을 연다.
- 작업디렉토리에 출력파일을 저장한다.

7. 인터랙티브 문서 작성한 보고서를 3단계를 거쳐 인터랙티브 Shiny 문서변환.

1 YAML 헤더에 **runtime: shiny** 을 추가한다.

```
---
title: "Line graph"
output: html_document
runtime: shiny
---
```

2 코드 덩어리에, 위젯을 내장하는 Shiny **input** 함수를 추가한다. Shiny **render** 함수를 추가해서 반응형 출력결과를 내장한다.

```
---
title: "Line graph"
output: html_document
runtime: shiny
---

Choose a time series:
```{r echo = FALSE}
selectInput("data", "",
  c("co2", "lh"))
```

See a plot:
```{r echo = FALSE}
renderPlot({
  d <- get(input$data)
  plot(d)
})
```
```

3 **rmarkdown::run** 명령어로 렌더링하거나 RStudio **Run Document** 버튼을 클릭한다.

- 주목: 보고서는 Shiny 앱이 된다. 따라서, (인터랙티브 보고서를 위해) html_document 혹은 (인터랙티브 발표자료) ioslides_presentation 출력형식을 선택한다.

8. Publish 온라인으로 접속하는 사용자와 보고서를 공유한다.

Rpubs.com

RStudio 무료 R 마크다운 게시 사이트를 통해 정적 문서를 공유한다.
www.rpubs.com

ShinyApps.io

Studio 서버에 인터랙티브 문서를 올려 호스팅한다. 무료와 유료 선택옵션이 있다.
www.shinyapps.io

RStudio 미리보기 창에 "Publish" 버튼을 클하여, rpubs.com 사이트에 버튼 한번 클릭으로 바로 올린다.



9. 추가 학습

문서와 예제 - rmarkdown.rstudio.com
추가 기사 - shiny.rstudio.com/articles
- blog.rstudio.com
- @rstudio

RStudio

References

1. Cheatsheets
    - 영문 (2페이지)
        - `M6X References/cheatsheets/08. rmarkdown-2.0.pdf`
    - 한글 (2페이지)
        - `M6X References/cheatsheets/09. rmarkdown-cheatsheet-kr`
    - 영문 (5페이지)
        - https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
2. rmarkdown 공식 페이지
    - http://rmarkdown.rstudio.com
3. LaTex
    - pdf 조판 언어
    - https://www.nyu.edu/projects/beber/files/Chang_LaTeX_sheet.pdf

Carseats (M22-Carseats 파일 참조)

Carseat_pdfRmd

```
---
title: 'Carseat 판매량 '
author: "Learning Spoons"
date: "`r Sys.Date()`"
output:
  pdf_document:
    latex_engine: xelatex
    keep_tex: true
    # pandoc_args: [
    #   "-v", "classoption=twocolumn"
    # ]
  smaller: true
mainfont: NanumGothic
classoption: a4paper
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Carseat 소개

```{r}
library(ISLR)
library(dplyr)
library(ggplot2)
str(Carseats)
```

Focus City

```{r}
focusCity <- Carseats %>%
  filter(Income > 100) %>%
  filter(Age >= 30 & Age < 40) %>%
  mutate(AdvPerCapita = Advertising/Population) %>%
  select(Sales, Income, Age, Population, Education, AdvPerCapita) %>%
  arrange(Sales)
print(focusCity)
```

14:21 Carseat 판매량

```

## Income vs Sales

```{r}
doFacetWrap <- FALSE
a <- ggplot(data = Carseats, aes(x = Income, y = Sales)) +
  geom_point(aes(shape = Urban, color = US))
if (doFacetWrap) {
  a <- a + facet_wrap(~ floor(Age/10))
}
print(a)
```

Your comment!

Income vs Sales

```{r}
doFacetWrap <- TRUE
a <- ggplot(data = Carseats, aes(x = Income, y = Sales)) +
  geom_point(aes(shape = Urban, color = US))
if (doFacetWrap) {
  a <- a + facet_wrap(~ floor(Age/10))
}
print(a)
```

Carseat 판매량

Learning Spoons

2018-04-21

Carseat 소개

```
library(ISLR)
library(dplyr)
```

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

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

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

```
library(ggplot2)
str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

Focus City

```
focusCity <- Carseats %>%
  filter(Income > 100) %>%
  filter(Age >= 30 & Age < 40) %>%
  mutate(AdvPerCapita = Advertising/Population) %>%
  select(Sales, Income, Age, Population, Education, AdvPerCapita) %>%
  arrange(Sales)
print(focusCity)
```

```
##    Sales Income Age Population Education AdvPerCapita
## 1   5.04    114  34        298        16   0.00000000
## 2   5.32    116  39        170        16   0.00000000
## 3   6.80    117  38        337        10   0.01483680
## 4   7.49    119  35        178        13   0.03370787
## 5   7.67    117  36        400        10   0.02000000
## 6   8.55    111  36        480        16   0.04791667
## 7   8.97    107  33        144        13   0.00000000
## 8   9.39    118  32        445        15   0.03146067
## 9   9.58    104  37        353        17   0.06515581
## 10 10.36    105  34        428        12   0.04205607
## 11 12.59    120  30        262        10   0.06725191
## 12 12.67    108  33        203        14   0.08374384
```

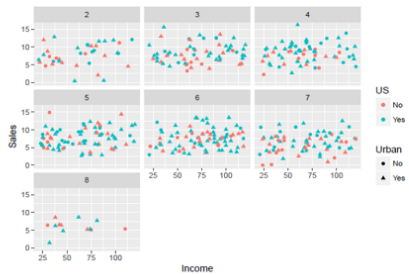
Income vs Sales

```
doFacetWrap <- FALSE
a <- ggplot(data = Carseats, aes(x = Income, y = Sales)) +
  geom_point(aes(shape = Urban, color = US))
if (doFacetWrap) {
  a <- a + facet_wrap(~ floor(Age/10))
}
print(a)
```

Your comment!

Income vs Sales

```
doFacetWrap <- TRUE
a <- ggplot(data = Carseats, aes(x = Income, y = Sales)) +
  geom_point(aes(shape = Urban, color = US))
if (doFacetWrap) {
  a <- a + facet_wrap(~ floor(Age/10))
}
print(a)
```



3

Templates (`M25, M26, M27`)

- `M25-.pdf`: 2 columns, line numbering, page numbering
- `M27-beamer`: 2 columns, Singapore theme, line numbering, page numbering
- `M26-slidy`: 각종 컬러 삽입
- **하위폴더(rmd-pdf-template_files, myRmdBeamerStyle)등을 데리고 다녀야 됨**

Discussion - rmarkdown

- Markdown이라는 Markup Language로 변환하여 사용성 높음
- 변환 과정
    1. Rmd -> md -> docx, html
    2. Rmd -> md -> tex -> pdf
- Advantages
    - 약간의 수정으로 여러가지 형식의 문서를 만들 수 있음
    - 자연스럽게 문법에 맞는 Color Coding
    - R output에 가장 비슷한 quality의 output을 얻을 수 있음.
    - 분석 -> 정리의 반복적인 작업을 하나로 줄여줌.

Discussion - Literature Programming

- Features
  1. 프로그래밍이 아닌 글쓰기가 초점
  2. 데이터 분석하고 insight를 공유하는 업무에 연관성이 높음!
  3. 글을 쓴다는 것과 프로그래밍을 하는 것은 자기 자신을 표현하는 가장 높은 수준의 지적인 활동
  4. 이것을 한꺼번에 할 수 있게 해주는 도구
  5. Express Yourself!!!
  6. ex) Python Notebook
- Advantages
  1. 코드와 문서가 하나의 파일이라서 관리가 편함
  2. 코드에서 주석을 조금만 달아도 됨
  3. 코드만 있는 것에 비해서 의사소통이 용이함
  4. 데이터나 분석의 결과가 달라지는 것이 문서에 즉각적으로 반영됨
  5. 반복적으로 작성하는 문서 작업과 엑셀 작업을 안해도 됨

"You don't know what you will can get away with until you try" - Colin Powell

pdf 파일 제작에 필요한 tex 엔진 설치

1. Texlive 2017 설치 (< 1Gb)

- http://www.ktug.org/xe/index.php?mid=install

2. 한글 폰트 설치

- google “nanumgothic download”
- google “nanummyeongjo download”
- google “nanumgothiccoding download”

3. 참고: google “latex beamer에서 한글 쓰기!”

실습과제 1

1. `C:/LS-DS/classProject`라는 폴더를 만드세요.
2. Rstudio를 열어서 rmarkdown의 html형식에 해당하는 .Rmd파일을 만드세요.
3. `classProject1.Rmd`로 위 폴더에 저장하세요.
4. Rstudio를 닫고 `classProject1.Rmd`를 더블클릭하여 실행하면 working directory가 자동으로 위의 폴더가 됩니다.
5. github `M41-longevity`에서 `lifeCountry.csv`파일을 다운받아서 위 폴더에 넣으세요.
6. 이제 분석을 위한 모든 준비가 끝났습니다.
7. Infile/Preprocessing을 해야합니다. `classProject1.Rmd`에서 `lifeCountry.csv`를 불러와서 `data.frame`으로 저장합니다.
8. 각 나라의 GDP와 기대수명에 대해서 산점도를 그리고 대륙에 따라서 점의 색깔이 달라지게 해보세요.
9. 자유롭게 분석을 시작해보세요.
10. html로 제작해 이메일을 보내주세요. `learningSpoonsR@gmail.com`