lec 1108

박효선 1585063

2018년 12월 3일

################  
# 11 월 8 일  
################  
#물어볼 것 table3 tidy data 인지? - 각 값은 각자의 cell 가지고 있기 때문임  
  
  
  
##################  
# Other strategies  
# 모든 자료를 character 로 읽은 후 col\_type 지정  
# col(): Create column specification 강제로 자료 형태 지정  
# .default: 디폴트로 자료 형태 지정 Any named columns not explicitly overridden in ... will be read with this column type.  
challenge2 <- read\_csv(readr\_example("challenge.csv"),  
 col\_types=cols(.default=col\_character()))   
challenge2

## # A tibble: 2,000 x 2  
## x y   
## <chr> <chr>  
## 1 404 <NA>   
## 2 4172 <NA>   
## 3 3004 <NA>   
## 4 787 <NA>   
## 5 37 <NA>   
## 6 2332 <NA>   
## 7 2489 <NA>   
## 8 1449 <NA>   
## 9 3665 <NA>   
## 10 3863 <NA>   
## # ... with 1,990 more rows

tail(challenge2)

## # A tibble: 6 x 2  
## x y   
## <chr> <chr>   
## 1 0.805274312151596 2019-11-21  
## 2 0.1635163405444473 2018-03-29  
## 3 0.47193897631950676 2014-08-04  
## 4 0.7183186465408653 2015-08-16  
## 5 0.26987858884967864 2020-02-04  
## 6 0.608237189007923 2019-01-06

# type\_convert() : type 에 맞게 column type 바꿔준다  
# you can read the columns in as character, clean it up with (e.g.) regular expressions and then let readr take another stab at parsing it.  
challenge3 <- type\_convert(challenge2)

## Parsed with column specification:  
## cols(  
## x = col\_double(),  
## y = col\_date(format = "")  
## )

challenge3

## # A tibble: 2,000 x 2  
## x y   
## <dbl> <date>   
## 1 404 NA   
## 2 4172 NA   
## 3 3004 NA   
## 4 787 NA   
## 5 37 NA   
## 6 2332 NA   
## 7 2489 NA   
## 8 1449 NA   
## 9 3665 NA   
## 10 3863 NA   
## # ... with 1,990 more rows

tail(challenge3)

## # A tibble: 6 x 2  
## x y   
## <dbl> <date>   
## 1 0.805 2019-11-21  
## 2 0.164 2018-03-29  
## 3 0.472 2014-08-04  
## 4 0.718 2015-08-16  
## 5 0.270 2020-02-04  
## 6 0.608 2019-01-06

# n\_max 옵션을 이용하여 자료의 읽부를 읽어 문제를 해결한 후에 전체 자료를 읽는 것이 좋다  
# 심각한 parising 문제가 있는 경우에는 read\_lines(), read\_file()을 아용하여  
# character 로 자료를 읽은 후 format 등을 이용하여 parsing  
  
#####################  
# Writing to a file  
# write\_csv() 와 write\_tsv()  
# – string 은 UTF-8 로 encoding  
# – date, date-time 은 ISO8601 format 으로 저장.  
# write\_excel\_csv(): excel 파일로 저장  
# – 파일의 시작 부분에 UTF-8 로 encoding 을 하고 있다는 것을 excel 에 알려주는 특별 문자(“byte order mark”)를 써줌  
getwd()

## [1] "C:/Users/HS/Documents/GitHub/Statistical-Graphics-with-Big-Data/Study for tidy"

write\_csv(challenge3, "ch.csv")  
# csv 로 저장할 때에는 type 에 대한 정보는 잃어버림  
read\_csv("ch.csv")

## Parsed with column specification:  
## cols(  
## x = col\_integer(),  
## y = col\_character()  
## )

## Warning in rbind(names(probs), probs\_f): number of columns of result is not  
## a multiple of vector length (arg 1)

## Warning: 1000 parsing failures.  
## row # A tibble: 5 x 5 col row col expected actual file expected <int> <chr> <chr> <chr> <chr> actual 1 1001 x no trailing characters .23837975086644292 'ch.csv' file 2 1002 x no trailing characters .41167997173033655 'ch.csv' row 3 1003 x no trailing characters .7460716762579978 'ch.csv' col 4 1004 x no trailing characters .723450553836301 'ch.csv' expected 5 1005 x no trailing characters .614524137461558 'ch.csv'  
## ... ................. ... ................................................................ ........ ................................................................ ...... ................................................................ .... ................................................................ ... ................................................................ ... ................................................................ ........ ................................................................  
## See problems(...) for more details.

## # A tibble: 2,000 x 2  
## x y   
## <int> <chr>  
## 1 404 <NA>   
## 2 4172 <NA>   
## 3 3004 <NA>   
## 4 787 <NA>   
## 5 37 <NA>   
## 6 2332 <NA>   
## 7 2489 <NA>   
## 8 1449 <NA>   
## 9 3665 <NA>   
## 10 3863 <NA>   
## # ... with 1,990 more rows

# write\_rds()와 read\_rds()를 이용.  
# – R 의 기본함수인 saveRDS()와 readRDS()에 해당  
# – binary format 으로 저장  
write\_rds(challenge3, "ch.rds") # 정보손실 x  
A <- read\_rds("ch.rds")  
A

## # A tibble: 2,000 x 2  
## x y   
## <dbl> <date>   
## 1 404 NA   
## 2 4172 NA   
## 3 3004 NA   
## 4 787 NA   
## 5 37 NA   
## 6 2332 NA   
## 7 2489 NA   
## 8 1449 NA   
## 9 3665 NA   
## 10 3863 NA   
## # ... with 1,990 more rows

#####################  
# 드디어 나왔다 Tidy data  
# dataset 을 tidy 하게 만들기 위한 3 가지의 기본 규칙  
# - 각 변수는 각자의 column 을 가지고 있어야 한다.  
# - 각 observation 은 각자의 row 를 가지고 있어야 한다.  
# - 각 값은 각자의 cell 을 가지고 있어야 한다.  
  
# Tidy 자료   
# 1. 각 변수는 각자의 column을 가지고 있어야 한다  
# 2. 각 observation은 각각의 row를 가지고 있어야 한다  
# 3. 각 값은 각자의 cell 을 가지고 있어야 한다  
  
# Tidy 자료의 장점  
# 1. 자료를 저장하는데에 일관된 방법을 제공하고 이후 작업을 위한 도구들을 이에 맞춰 개발하므로 배우기 쉽다  
# 2. 변수를 column 으로 놓는 것은 특히 유용하다   
# R의 내장된 함수들은 vector를 기본으로 하고 있으므로 tidy data 를 vector 로 변환하는 것이 유용  
  
  
table1 # tidy data O

## # A tibble: 6 x 4  
## country year cases population  
## <chr> <int> <int> <int>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

table2 # tidy data 아님 - 하나의 observation 이 두 2 rows에 있음. 각 나라, 연도별로 type, count 한 행을 두행으로 표현

## # A tibble: 12 x 4  
## country year type count  
## <chr> <int> <chr> <int>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 1999 population 19987071  
## 3 Afghanistan 2000 cases 2666  
## 4 Afghanistan 2000 population 20595360  
## 5 Brazil 1999 cases 37737  
## 6 Brazil 1999 population 172006362  
## 7 Brazil 2000 cases 80488  
## 8 Brazil 2000 population 174504898  
## 9 China 1999 cases 212258  
## 10 China 1999 population 1272915272  
## 11 China 2000 cases 213766  
## 12 China 2000 population 1280428583

table3 # tidy data 아님 - 하나의 column 에 두 2 variables의 값을 가지고 있음. 한줄에 나타나있긴 하지만 한칸에 두개

## # A tibble: 6 x 3  
## country year rate   
## \* <chr> <int> <chr>   
## 1 Afghanistan 1999 745/19987071   
## 2 Afghanistan 2000 2666/20595360   
## 3 Brazil 1999 37737/172006362   
## 4 Brazil 2000 80488/174504898   
## 5 China 1999 212258/1272915272  
## 6 China 2000 213766/1280428583

table4a # tidy data 아님 - year 이라는 variables가 여러2columns에 걸쳐져 있음

## # A tibble: 3 x 3  
## country `1999` `2000`  
## \* <chr> <int> <int>  
## 1 Afghanistan 745 2666  
## 2 Brazil 37737 80488  
## 3 China 212258 213766

table4b # tidy data 아님 - t4a, t4b 두개의 table 로 저장되어 있음

## # A tibble: 3 x 3  
## country `1999` `2000`  
## \* <chr> <int> <int>  
## 1 Afghanistan 19987071 20595360  
## 2 Brazil 172006362 174504898  
## 3 China 1272915272 1280428583

# table1 이 tidy data 인 것이다!  
# tidy 데이터는 자료를 다루기 쉽다. 별다른 전처리 없이 rate 계산 가능  
table1 %>% mutate(rate = cases/population\*10000)

## Warning: package 'bindrcpp' was built under R version 3.5.1

## # A tibble: 6 x 5  
## country year cases population rate  
## <chr> <int> <int> <int> <dbl>  
## 1 Afghanistan 1999 745 19987071 0.373  
## 2 Afghanistan 2000 2666 20595360 1.29   
## 3 Brazil 1999 37737 172006362 2.19   
## 4 Brazil 2000 80488 174504898 4.61   
## 5 China 1999 212258 1272915272 1.67   
## 6 China 2000 213766 1280428583 1.67

table1 %>% count(year)

## # A tibble: 2 x 2  
## year n  
## <int> <int>  
## 1 1999 3  
## 2 2000 3

# Compute cases per year  
table1 %>% count(year, wt = cases)

## # A tibble: 2 x 2  
## year n  
## <int> <int>  
## 1 1999 250740  
## 2 2000 296920

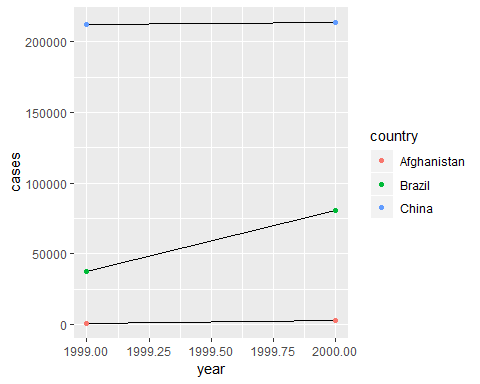
745+37737+212258

## [1] 250740

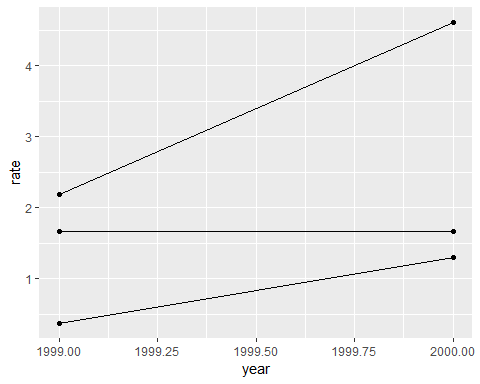
2666+80488+213766

## [1] 296920

table1 %>% ggplot(aes(year, cases)) +  
 geom\_line(aes(group = country)) +   
 geom\_point(aes(color = country))



# Visualise changes over time  
table1 %>% mutate(rate = cases/population\*10000) %>%   
 ggplot(aes(year, rate, group = country)) +   
 geom\_point() + geom\_line()



###############################  
## Spreading and gathering  
# 대부분의 자료는 모아지기 편리한 형태로 구성되어 있으므로  
# 분석하기 편한 tidy 형태로 만드는 작업이 필요  
# 1. 하나의 변수들을 multiple column 에 표현해야될 수 있다 -> tidy : gather  
# 2. 하나의 관측을 여러 줄에 표현해야 할 수 있다 -> tidy : spread  
  
  
####################  
# gathering  
# Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed.   
# You use gather() when you notice that you have columns that are not variables.  
  
# year 변수가 2 columns 에 걸쳐져 입력되어 있음  
# tidy 가 되기 위해서 한 column 에 한 variable  
  
# tidy 자료로 만들기 위해서 column 을 새로운 변수의 pair 로 만들어야함. 이를  
# gathering 이라고 함.  
# 이를 위하여 3 가지의 인자를 지정하는 것이 필요.  
# 1. 값을 나타내는 column 의 이름; table4a 에서는 1999 와 2000  
# 2. column 의 이름에 나타난 값을 위한 변수 이름 (key); ‘year’  
# 3. column 에서 가지고 있는 값을 저장하기 위한 변수 이름 (value);‘cases’  
  
gather(table4a, `1999`, `2000`, key = "YEAR", value = "CASES") # 변수명이 year 라는 새 변수명의 값으로 들어가야해

## # A tibble: 6 x 3  
## country YEAR CASES  
## <chr> <chr> <int>  
## 1 Afghanistan 1999 745  
## 2 Brazil 1999 37737  
## 3 China 1999 212258  
## 4 Afghanistan 2000 2666  
## 5 Brazil 2000 80488  
## 6 China 2000 213766

A <- gather(table4a, `1999`, `2000`, key = "YEAR", value = "CASES")  
B <- gather(table4a, `1999`, `2000`, key = "YEAR", value = "POPULATION")  
A

## # A tibble: 6 x 3  
## country YEAR CASES  
## <chr> <chr> <int>  
## 1 Afghanistan 1999 745  
## 2 Brazil 1999 37737  
## 3 China 1999 212258  
## 4 Afghanistan 2000 2666  
## 5 Brazil 2000 80488  
## 6 China 2000 213766

B

## # A tibble: 6 x 3  
## country YEAR POPULATION  
## <chr> <chr> <int>  
## 1 Afghanistan 1999 745  
## 2 Brazil 1999 37737  
## 3 China 1999 212258  
## 4 Afghanistan 2000 2666  
## 5 Brazil 2000 80488  
## 6 China 2000 213766

left\_join(A, B)

## Joining, by = c("country", "YEAR")

## # A tibble: 6 x 4  
## country YEAR CASES POPULATION  
## <chr> <chr> <int> <int>  
## 1 Afghanistan 1999 745 745  
## 2 Brazil 1999 37737 37737  
## 3 China 1999 212258 212258  
## 4 Afghanistan 2000 2666 2666  
## 5 Brazil 2000 80488 80488  
## 6 China 2000 213766 213766

####################  
# spreading  
# spreading 은 gathering 의 반대로 observation 이 여러줄에 나타날 때 이용  
# Spread a key-value pair across multiple columns. obs가 여러 줄에 나타났을 때!   
  
spread

## function (data, key, value, fill = NA, convert = FALSE, drop = TRUE,   
## sep = NULL)   
## {  
## UseMethod("spread")  
## }  
## <bytecode: 0x000000001ce3f700>  
## <environment: namespace:tidyr>

# 한가지 관측이 3 rows 에 입력되어 있음.   
# tidy 가 되기 위해서 한 row 에 한 variable  
# 위의 자료를 tidy 자료로 바꾸기 위해서는 두가지의 인자가 필요.  
# 1. 변수 이름이 저장되어 있는 key column: 여기서는 type  
# 2. 변수의 값이 저장되어 있는 value column: 여기서는 value  
  
table2

## # A tibble: 12 x 4  
## country year type count  
## <chr> <int> <chr> <int>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 1999 population 19987071  
## 3 Afghanistan 2000 cases 2666  
## 4 Afghanistan 2000 population 20595360  
## 5 Brazil 1999 cases 37737  
## 6 Brazil 1999 population 172006362  
## 7 Brazil 2000 cases 80488  
## 8 Brazil 2000 population 174504898  
## 9 China 1999 cases 212258  
## 10 China 1999 population 1272915272  
## 11 China 2000 cases 213766  
## 12 China 2000 population 1280428583

spread(table2, key = type, value = count) # key = "새로운 변수명", value = 값

## # A tibble: 6 x 4  
## country year cases population  
## <chr> <int> <int> <int>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

###########################  
# Separateing and Uniting  
# table3 의 “하나의 column 에 두변수의 값”을 가지고 있는 문제를 separate()과 unite()을 이용하여 해결  
# Separate - character  
# 하나의 column 에 두 변수의 값을 가지고 있는 문제를 seperate() 와 unite() 로 구분  
separate

## function (data, col, into, sep = "[^[:alnum:]]+", remove = TRUE,   
## convert = FALSE, extra = "warn", fill = "warn", ...)   
## {  
## UseMethod("separate")  
## }  
## <bytecode: 0x000000001ce29620>  
## <environment: namespace:tidyr>

table3

## # A tibble: 6 x 3  
## country year rate   
## \* <chr> <int> <chr>   
## 1 Afghanistan 1999 745/19987071   
## 2 Afghanistan 2000 2666/20595360   
## 3 Brazil 1999 37737/172006362   
## 4 Brazil 2000 80488/174504898   
## 5 China 1999 212258/1272915272  
## 6 China 2000 213766/1280428583

separate(table3, rate, into = c("A", "B")) # rate 라는 변수를 A, B로 나누세요

## # A tibble: 6 x 4  
## country year A B   
## \* <chr> <int> <chr> <chr>   
## 1 Afghanistan 1999 745 19987071   
## 2 Afghanistan 2000 2666 20595360   
## 3 Brazil 1999 37737 172006362   
## 4 Brazil 2000 80488 174504898   
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

separate(table3, rate, into = c("cases", "population")) # 쭉 보다가 숫자가 아닌 / 문자가 들어가서 거기에서 나누어줌

## # A tibble: 6 x 4  
## country year cases population  
## \* <chr> <int> <chr> <chr>   
## 1 Afghanistan 1999 745 19987071   
## 2 Afghanistan 2000 2666 20595360   
## 3 Brazil 1999 37737 172006362   
## 4 Brazil 2000 80488 174504898   
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

table1

## # A tibble: 6 x 4  
## country year cases population  
## <chr> <int> <int> <int>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

separate(table3, rate, into = c("cases", "population"), sep = "/") # 무족권..,.,.,.char 형태로 남아있다.

## # A tibble: 6 x 4  
## country year cases population  
## \* <chr> <int> <chr> <chr>   
## 1 Afghanistan 1999 745 19987071   
## 2 Afghanistan 2000 2666 20595360   
## 3 Brazil 1999 37737 172006362   
## 4 Brazil 2000 80488 174504898   
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

separate(table3, rate, into = c("cases", "population"), sep = "/", convert = TRUE) # 이후에 계산으로 쓰고 싶으면 convert = TRUE

## # A tibble: 6 x 4  
## country year cases population  
## \* <chr> <int> <int> <int>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

separate(table3, rate, into = c("cases", "population"), sep = 2) # 나누는 글의갯수 정해주기, 숫자 만큼 나눠주기

## # A tibble: 6 x 4  
## country year cases population   
## \* <chr> <int> <chr> <chr>   
## 1 Afghanistan 1999 74 5/19987071   
## 2 Afghanistan 2000 26 66/20595360   
## 3 Brazil 1999 37 737/172006362   
## 4 Brazil 2000 80 488/174504898   
## 5 China 1999 21 2258/1272915272  
## 6 China 2000 21 3766/1280428583

separate(table3, rate, into = c("cases", "population"), sep = 2, convert = TRUE)

## # A tibble: 6 x 4  
## country year cases population   
## \* <chr> <int> <int> <chr>   
## 1 Afghanistan 1999 74 5/19987071   
## 2 Afghanistan 2000 26 66/20595360   
## 3 Brazil 1999 37 737/172006362   
## 4 Brazil 2000 80 488/174504898   
## 5 China 1999 21 2258/1272915272  
## 6 China 2000 21 3766/1280428583

table5 <- separate(table3, year, into = c("century", "year"), sep = 2) # convert = TRUE  
table5

## # A tibble: 6 x 4  
## country century year rate   
## \* <chr> <chr> <chr> <chr>   
## 1 Afghanistan 19 99 745/19987071   
## 2 Afghanistan 20 00 2666/20595360   
## 3 Brazil 19 99 37737/172006362   
## 4 Brazil 20 00 80488/174504898   
## 5 China 19 99 212258/1272915272  
## 6 China 20 00 213766/1280428583

####################  
# uniting  
# unite(): 여러개의 column 을 하나의 column 으로 합쳐줌.  
  
# unite()에서는 두번째 인자부터 나열된 변수들의 값을 “\_“를 이용하여 연결한 후 첫번째  
# 인자에 지정된 값의 변수에 저장한다.  
table5 %>% unite(new, century, year)

## # A tibble: 6 x 3  
## country new rate   
## <chr> <chr> <chr>   
## 1 Afghanistan 19\_99 745/19987071   
## 2 Afghanistan 20\_00 2666/20595360   
## 3 Brazil 19\_99 37737/172006362   
## 4 Brazil 20\_00 80488/174504898   
## 5 China 19\_99 212258/1272915272  
## 6 China 20\_00 213766/1280428583

# sep 옵션을 이용하여 연결문자 지정  
table5 %>% unite(new, century, year, sep = "") # sep = 연결 문자 지정. 여전히 char로 읽는다

## # A tibble: 6 x 3  
## country new rate   
## <chr> <chr> <chr>   
## 1 Afghanistan 1999 745/19987071   
## 2 Afghanistan 2000 2666/20595360   
## 3 Brazil 1999 37737/172006362   
## 4 Brazil 2000 80488/174504898   
## 5 China 1999 212258/1272915272  
## 6 China 2000 213766/1280428583

# table5 %>% unite(new, century, year, sep = "", convert = TRUE) # convert 옵션 없어  
# Error: `TRUE` must evaluate to column positions or names, not a logical vector  
# In addition: Warning message:  
# 'glue::collapse' is deprecated.  
# Use 'glue\_collapse' instead.  
# See help("Deprecated") and help("glue-deprecated").   
  
  
  
###################  
# Missing values  
# Missing 의 종류  
# - explicitly missing: 자료는 있으나 파악되지 못하여 NA 로 표시  
# – implicitly missing: 자료가 존재하지 않는 경우  
stocks <- tibble(year = c(2015,2015,2015,2015,2016,2016,2016),  
 qtr = c(1,2,3,4,2,3,4),  
 return = c(1.88,0.59,0.35,NA,0.92,0.17,2.66))  
stocks

## # A tibble: 7 x 3  
## year qtr return  
## <dbl> <dbl> <dbl>  
## 1 2015 1 1.88  
## 2 2015 2 0.59  
## 3 2015 3 0.35  
## 4 2015 4 NA   
## 5 2016 2 0.92  
## 6 2016 3 0.17  
## 7 2016 4 2.66

# 2015년 qtr4 자료 = explictly missing   
# 2016년 qtr1 자료 = implicit missing  
# spread 사용하여 implicit missing 을 explictly missing 으로 만들어준다  
stocks %>% spread(year, return)

## # A tibble: 4 x 3  
## qtr `2015` `2016`  
## <dbl> <dbl> <dbl>  
## 1 1 1.88 NA   
## 2 2 0.59 0.92  
## 3 3 0.35 0.17  
## 4 4 NA 2.66

# spread 와 gather 동시에 사용해서 missing 모두 드러낸다  
stocks %>% spread(year, return) %>%   
 gather(year, return, `2015`, `2016`)

## # A tibble: 8 x 3  
## qtr year return  
## <dbl> <chr> <dbl>  
## 1 1 2015 1.88  
## 2 2 2015 0.59  
## 3 3 2015 0.35  
## 4 4 2015 NA   
## 5 1 2016 NA   
## 6 2 2016 0.92  
## 7 3 2016 0.17  
## 8 4 2016 2.66

# explicit missing 을 제거하고 싶은 경우에는 na.rm=TRUE 옵션을 이용  
stocks %>% spread(year, return) %>%   
 gather(year, return, `2015`, `2016`, na.rm = TRUE)

## # A tibble: 6 x 3  
## qtr year return  
## \* <dbl> <chr> <dbl>  
## 1 1 2015 1.88  
## 2 2 2015 0.59  
## 3 3 2015 0.35  
## 4 2 2016 0.92  
## 5 3 2016 0.17  
## 6 4 2016 2.66

# gather()과 spread()는 대칭적인 함수인가  
# (gather 와 spread 를 거치면 같은 자료가 되는가)? 안된다!!!!!  
# 처음의 stocks 에서 변수들은 모두 double, 그러나 spread, gather 을 거친 후에는  
# spread 에서 변수 이름으로 쓰인 year 가 character 로 바뀜.  
  
####################  
# complete  
# complete() 함수를 이용하여 implicit missing 을 exlicit missing 으로 변환.  
# complete(): 모든 unique combination 을 찾아서 만들어줌. missing 은 NA 로 채우기.  
stocks %>% complete(year, qtr)

## # A tibble: 8 x 3  
## year qtr return  
## <dbl> <dbl> <dbl>  
## 1 2015 1 1.88  
## 2 2015 2 0.59  
## 3 2015 3 0.35  
## 4 2015 4 NA   
## 5 2016 1 NA   
## 6 2016 2 0.92  
## 7 2016 3 0.17  
## 8 2016 4 2.66

treatment<-tribble( # row - by -row 는 tribble  
 ~person, ~treatment,~response,  
 "DW",1,7,  
 NA,2,10,  
 NA,3,9,  
 "KB",1,4)  
  
treatment

## # A tibble: 4 x 3  
## person treatment response  
## <chr> <dbl> <dbl>  
## 1 DW 1 7  
## 2 <NA> 2 10  
## 3 <NA> 3 9  
## 4 KB 1 4

####################  
# fill() : missing을 LOCF(Last Observation Carried Forward)규칙에 따라 채워준  
fill(treatment, person) # NA 값 채워짐

## # A tibble: 4 x 3  
## person treatment response  
## <chr> <dbl> <dbl>  
## 1 DW 1 7  
## 2 DW 2 10  
## 3 DW 3 9  
## 4 KB 1 4