5회차

**### tidyr**

**:** Data format을 reshape하는데 사용

spread() – to wide

gather() – to log

library(tidyr) ## tidyr 패키지 불러오기

library(dplyr) ## 파이프라인(%>%)을 쓰기 위해서 불러오기

**# group\_by( ) %>% summarise( )로 데이터 요약하기**

aggr = delivery %>%

group\_by(시군구,시간대,요일,업종) %>%

summarise(통화건수=sum(통화건수)) %>%

as.data.frame() ## tibble 형태가 아닌 dataframe을 위해 사용

aggr

## spread(데이터이름, 기준변수이름(열로 나열할 값), 나열할 값(데이터로 나열할 값))

**## 1.1 spread( )로 데이터를 여러 열로 나누기(long -> wide)**

aggr %>%

spread(업종, 통화건수) ## spread 안에 들어가는 요소가 가능한 지 여부도 확인중요

aggr\_wide = aggr %>% spread(업종, 통화건수)

aggr\_wide

**##1.2 replace\_na( )로 결측값 처리하기**

## replace\_na(list(변수1=값, 변수2=값, ...))

# 보통 spread()를 사용하면 replace\_na()를 어떻게 사용할 것인지는 당연한 부분

# 변수에 값들을 한 번에 다 넣어주기 때문에 list로 묶어서 사용한다.

aggr\_wide %>% replace\_na(list(족발보쌈=999, 중국음식=0, 치킨=0)) ## NA 하나 남음

aggr\_wide %>% replace\_na(list(족발보쌈=999, 중국음식=0, 치킨=0, 피자=999)) ## NA 없음

aggr\_wide2 = aggr\_wide %>% replace\_na(list(족발보쌈=0, 중국음식=0, 치킨=0, 피자=0, 피자=999))

**##1.3 drop\_na( )로 결측값을 포함한 관측치 버리기**

## drop\_na(결측값을 찾을 변수1, 변수2, ...)

**# 한 행 때문에 행을 날리는 것은 그렇게 좋은 것이 아니다. 데이터 하나하나가 돈이기 때문이다.**

# 오른쪽 데이터 확인하는 공간에서aggr\_wide2의 행이 3947인 것과 drop\_na가 적용된 것이 2617인 것이 확인되면 정상적으로 drop\_na가 작동되었음을 알 수 있다.

# 현업에서는 데이터가 현저하게 적을 때 결측값이 포함된 값이 1% 미만일때 drop\_na 사용

그 정도까지의 bias는 주지 않는다.

**##1.4 gather( )로 여러 열을 한 열+구분변수로 만들기(wide->long)**

## gather(데이터이름, 새기준변수이름, 새변수이름, 모을 변수들)

#어떤 펼쳐져있는 것을 어떤 기준으로 모으는 행위(처리)

# 아래 결과는 같지만 표현 방식이 다를 뿐임.

# 시군구:요일은 () 로 꼭 묶어줘야 한다.

# aggr\_long(12845)과 aggr(12788)의 행이 차이는 aggr이 이전에 replace\_na() 결측값 제거 진행

aggr\_wide2 %>% gather(Category, Count, 족발보쌈:치킨)

aggr\_wide2 %>% gather(Category, Count, 족발보쌈, 중국음식, 피자, 치킨)

aggr\_long = aggr\_wide2 %>% gather(Category, Count, -(시군구:요일))

**##1.5 complete( )로 빠져있는 조합 채우기 + fill= 옵션으로 빈값 채우기**

**## 3947 != 4200 = 25(시군구)\*24(시간대)\*7(요일) / 총 경우의 수는 4200, 하지만 aggr\_wide2는 3947개 임으로 4200개로 맞춰주고 데이터가 비는 부분은 complete()로 처리한다.**

nrow(aggr\_wide2)

aggr\_wide2 %>% complete(시군구, 시간대, 요일)

aggr\_wide2 %>% complete(시군구, 시간대, 요일, fill=list(족발보쌈=0, 중국음식=0, 치킨=0, 피자=0))

* csv 파일보다 확실히 xlsx 파일의 속도가 load하는데 있어서 느리다.

그 이유는 xlsx 같은 경우에는 load 시 구분자가 컴마, 쉼표, 탭이기 때문에 모든 경우의 수를

검토하고 있는 것이다.

**## 2 (실습) 서울시 지하철 이용데이터**

**# 데이터 불러오기**

library(openxlsx)

subway\_2017 = read.xlsx('subway\_1701\_1709.xlsx')

subway\_2017

**# 데이터의 구조 확인**

str(subway\_2017)

**# 첫 10개 관측치만 확인하기**

head(subway\_2017, n=10)

**# 변수이름 확인 ->이름변환**

**# 컬럼명들이 나오고 확인이 가능하다.**

names(subway\_2017)

names(subway\_2017)[6:25]

substr(names(subway\_2017)[6:25], 1, 2) ## 텍스트를 추출할꺼임

## 첫번째 시작기준 2개 추출

* 변수에 ~(05~06)이 있으면 텍스트 묶여서 변환되는 경우가 있음으로 자주 안쓰임.

## 앞에 'H'를 붙임 paste0() 함수 사용

**######################## 연습문제 #############################**

**# (실습) gather( ) 함수를 활용하여 H05부터 H24까지 변수를 모아**

**# '시간대'와 '승객수'으로 구분하는 데이터 subway2 만들기**

subway2 = gather(subway\_2017, 시간대, 승객수, H05:H24)

**## 위에서 만든 subway2 데이터와 dplyr 패키지를 활용하여**

**# 역명/시간대별 전체 승객수 합계 계산 (승객수 합계의 내림차순으로 정렬)**

subway2 %>%

group\_by(역명, 시간대) %>%

summarise(SUM = sum(승객수)) %>%

arrange(desc(SUM))

**# 위의 결과를 spread( ) 함수를 활용해서 표 형태로 변환**

subway2 %>%

group\_by(역명, 시간대) %>%

summarise(SUM = sum(승객수)) %>%

spread(시간대, SUM)

**## 역명/시간대/구분별 전체 승객수 합계 계산**

subway2 %>%

group\_by(역명, 시간대, 구분) %>%

summarise(SUM = sum(승객수)) %>%

arrange(desc(SUM))

**## 2월 한달간 역명/시간대/구분별 전체 승객수 합계 계산**

subway2 %>%

filter(월==2) %>%

group\_by(역명, 시간대, 구분) %>%

summarise(SUM = sum(승객수)) %>%

arrange(desc(SUM))

**###### 문자열 다루기 ######**

**## 기본 패키지 설치하기**

install.packages("stringr")

library(stringr)

**# 큰따움표와 작은 따움표**

작은 따움표는 큰 따움표 안에 쓰는게 기본이다. 반대로 하면 R이 특수문자로 인식해서 오류남

string2 <- "If I want to include a 'quote' inside a string, I use single quotes"

**언제 사용되는가?**

R와 SQL을 연동하는 경우가 생기는데, 그때 텍스트로 쿼리로 불러올 수 있음. SQL의 WHERE 절의 구문을 불러올 때 사용하는데 큰 따움표로 묶고 안에는 작은 따움표로 사용한다.

괄호를 마무리하지 않으면 영원히 끝나지 않는 문장이 됨으로 “” 마무리는 꼭 해줄 것

**#### 패턴찾기 ####**

**Detect : 발견하다 (의미)**

**#str\_detect(데이터, 패턴)**

x <- c("apple", "banana", "pear")

str\_detect(x, "e")

* X 변수에 e라는 텍스트가 있어? 라는 의미 / 결과값은 T,F로 도출한다.

sum(str\_detect(x, "e"))

* E의 개수를 찾을 때는 sum을 써서 사용하기도 한다. (T,F)

**#str\_count(데이터, 패턴)**

str\_count(x,"e")

* X변수에 e가 몇 개 있어? 1,0,1

str\_count('aeeeeee','e')

* 'aeeeeee'에 e가 몇 개 있어?
* 결과값은 6

**#str\_which(데이터, 패턴)**

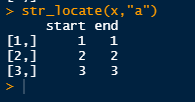
str\_which(x,"e")

* E가 X변수가 어디에 있어?
* 결과값은 1,3

**#str\_locate(데이터, 패턴)**

str\_locate(x,"a")

* x변수에서 a한 개의 시작과 끝이 어떻게 돼?



str\_locate("aaa","aa")

* aaa에서 aa의 시작과 끝이 어떻게 돼?



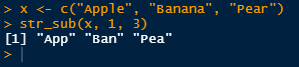
* **str\_detect & str\_which를 현업에서 많이 사용함.**

**### 부분집합 찾기**

**#str\_sub(데이터, 시작, 끝)**

x <- c("Apple", "Banana", "Pear")

str\_sub(x, 1, 3)



str\_sub(x, -3, -1) #뒤에서부터



* string 패키지 안에 있는 건 str\_sub(x,1,3)를 사용하고 내장함수는 substr(x,1,3)을 사용

#str\_subset(데이터, 패턴) /

* str\_sub 기재한 패턴만 뽑아온다.
* str\_subset는 기재한 패턴을 기준으로 그 단어를 다 뽑아온다.

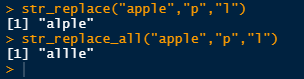
str\_subset(x,"r")



**### 문자열 변형하기**

# str\_replace(데이터,찾는변수,바꿀변수) : 처음 찾는 변수만 바뀐다.

# str\_replace\_all(데이터,찾는변수,바꿀변수) : 찾는 변수 모두가 바뀐다.



**# Mutate STrings**

str\_to\_lower("STRING") #소문자로

str\_to\_upper("string") #대문자로

str\_to\_title("string") #첫 글자만

**as.factor(x)**

# 주어진 객체 x를 팩터로 변환

**as.numeric(x)**

# 주어진 객체 x를 숫자를 저장한 벡터로 변환

**as.character(x)**

# 주어진 객체 x를 문자열을 저장한 벡터로 변환

**as.matrix(x)**

# 주어진 객체 x를 행렬로 변환

**as.data.frame(x)**

# 주어진 객체 x를 데이터 프레임으로 변환

x <- c("a", "b", "c")

as.factor(x)

* 결과값이 벡터 형식으로 labels 포함하여 나옴.

as.character(as.factor(x))

* 결과값이 문자형 형식으로 나옴

x <- matrix(1:9, ncol=3)

as.data.frame(x)

* 결과값이 dataframe 형식으로 나옴

**### 연습해보기**

**# 문제 1**

**# words를 모두 대문자로 바꾼 상태에서 'AA'를 포함한 단어의 개수는 총 몇개이며 어떤단어들이 있는가?**

sum(str\_detect(str\_to\_upper(words), "AB")) #8개

str\_to\_upper(words)[str\_which(str\_to\_upper(words),"AB")] #단어 확인 ## 다시 꼭 풀어보기

**# 문제 2**

**# words에서 "b"를 "a"로 모두 바꾸고 "aa"를 포함하는 단어 개수는?**

sum(str\_detect(str\_replace\_all(words,"b","a"), "aa")) #19개

**# 문제 3**

**# words에서 "e"의 수는 전체 합과 평균은 몇인가?**

sum(str\_count(words,"e"))

mean(str\_count(words,"e"))

**###### 날짜 데이터 ######**

**## Lubricate는 윤활유를 바르다라는 뜻으로 시간 데이터를 깔끔하게 처리할 수 있도록 도와주는패키지입니다.**

as.Date는 R에서 기본적으로 제공하는 날짜 함수이지만 “2020-11-12” 이런 형태의 고정 텍스트 데이터만 날짜 형태로 인식하는 문제가 발생 (융통성 없는 놈임)

as.Date(“2020-11-12”) => O

as.Date(“20201112”) => X

as.Date(20201112) => X

**lubridate**

Easy and fast parsing of date-times (쉬운 날짜생성)

- ymd\_hms, ymd(), dmy,hms(), …..

Ymd(20201210) #> [1] “2020-12”15”

Mdy(“4/1/17”) #> [1] “2017-04-01” ## 순서 바꾸기

**# 기본적으로 일반 텍스트 데이터를 날짜 데이터로 바꿉**

**# lubridate에 들어 있는 ymd() 함수는 어떤 모양이든 이를 날짜로 인식!**

ymd('20200110')

mdy('January 10th 2020')

dmy('10-jan-2020')

ymd('820327')

ymd(820327)

**# 날짜 데이터 뽑아내기**

date\_test <- ymd(191020)

year(date\_test)

month(date\_test)

day(date\_test)

week(date\_test) **#연도 기준 몇주차인지**

wday(date\_test) **## 몇일인지? (1~7기준)**

wday(date\_test,label = T) **## 몇일인지? (1~7기준 / labels 포함)**

**# 날짜로 각종 계산하기**

date\_test + days(100) #100일 후

date\_test + months(100) #100개월 후

date\_test + years(100) #100일 후

today()

today()-date\_test

**# 날짜 + 시간 데이터**

ymd\_hm(20-10-20 14:30)

ymd\_hm("20-10-20 14:30") **## 시간 같은 경우에는 " "으로 묶어줘야한다. 그렇지 않으면 시**

**간을 인식하지 못한다.**

hour(date\_test2)

minute(date\_test2)

second(date\_test2)

**### 연습해보기 날짜 연습해보기**

**# 문제1**

**# (실습) gather( ) 함수를 활용하여 H05부터 H24까지 변수를 모아**

**# '시간대'와 '승객수'으로 구분하는 데이터 subway2 만들기**

**# subway2 데이터의 날짜에 시간을 추가하기 ex) "2017-01-02 06:00:00"**

head(subway\_2017)

substr(names(subway\_2017)[6:25],1,2)

paste0("H", substr(names(subway\_2017)[6:25],1,2))

names(subway\_2017)[6:25] <- paste0("H", substr(names(subway\_2017)[6:25],1,2))

<정답>

subway2 <- gather(subway\_2017, 시간, 고객, H05:H24)

ymd\_h(paste0(subway\_2017$날짜, substr(subway2$시간,2,3)))

3이면 05() “0,5,빈공간”인데 이게 왜 2가 아니라 3인건지 여부, 시간에서 05: 출력되는데 : 부분까지도 고려해줘야 하는건지?

**# 문제2**

**# subway\_2017 데이터에서 월과 일을 month, day 변수명으로 추가하시오**

**<정답>**

subway\_2017$month <-month(subway\_2017$날짜)

subway\_2017$day <-day(subway\_2017$날짜)

**<나의풀이>**

mutate(subway\_2017, day = day(subway\_2017$날짜))

mutate(subway\_2017, month = month(subway\_2017$날짜))

**# 문제3**

**# 위에서 추가한 변수들 기반으로 3월중 가장 많이 탑승한 시간은 몇시인가?**

**<정답>**

subway\_2017 %>%

filter(month == 3) %>%

select(H06:H24) %>%

apply(2,sum) %>%

data.frame()

**<나의풀이> ## 마지막 확인부분까지 체크**

new\_subway\_2017 <- subway\_2017 %>%

filter(month == '3') %>%

select(H05:H24) %>%

apply(2,sum) %>%

as.data.frame

apply(new\_subway\_2017, 2, max) ## 데이터 프레임일 경우

max(new\_subway\_2017) ## 데이터 프레임일 경우

which.max(new\_subway\_2017) ## 데이터프레임이 아닐 경우

**## 결측값(NA)**

결측치(Missing Value)

• 누락된 값, 비어있는 값

• 함수 적용 불가, 분석 결과 왜곡

• 제거 후 분석 실시

* 전처리는 모든 데이터 전에 필수불가결한 작업이다.

**is.na() : 결측값 확인 (결측값이 있어? 없어?)**

• Apply 종류의 함수를 결합하여 결측치 분포 확인

• Ex) apply(데이터, 2, function(x){sum(is.na(x))})

Case1 – 결측치를 포함하는 행 제거 (na.omit) ## omit : 누락(제외)시키다

Case2 – 결측치를 특정 값으로 변경 is.na() <- mean ## 결측값을 mean으로 대체

해줘

df <- data.frame(sex = c("M", "F", NA, "M", "F"),

score = c(5, 4, 3, 4, NA))

is.na(df) **## 결측지 있어 없어?**

table(is.na(df)) **## 결측치 수가 어떻게 돼? ## count의 느낌임.**

table(is.na(df$sex)) **## 위와 동일**

summary(df)  **## 요약, 각 변수마다 NA의 수가 확인가능함 (결과값) NA’S**

* **데이터가 너무 많은 경우에는 감당하기 힘들기 때문에 apply를 이용한다.**

**(일괄적인 열처리를 할 때 빠르게 진행할 수 있다)**

* **## 텍스트 형은 NA가 없다. str(df)를 하게 되면 sex가 factor로 되어있음. factor는 우리 분석할 데이터인데 그런데 char는 분석할 데이터라고 인식을 안하고 하나의 텍스트로 인식한다. 나중에 바꾸고 싶다고 하면 as.factor로 바꿔주면 된다.**

**## apply 이용한 is.na(x)**

apply(X = df, MARGIN = 2, FUN = function(x){sum(is.na(x))})

apply(iris[,1:4], 2, function(x){sum(is.na(x))})

**## sapply 이용한 is.na(x)**

**## 행열 개념없고 데이터 프레임, 인자형태 모두 받는다.**

sapply(X = df, FUN = function(x){sum(is.na(x))})

**# 결측치 제거**

library(dplyr)

df %>% filter(is.na(score)) **## score 열의 결측지가 있는지 여부**

df %>% filter(!is.na(score)) **## score 열의 결측지가 없는지 여부**

df %>% filter(!is.na(score) & !is.na(sex)) **## 둘 중 결 측치가 하나라도 있으면 제거**

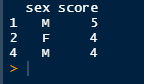
**## na.omit() /** omit: 빠뜨리다, 누락하다

# 모든 변수에 결측치 없는 데이터 추출

# 결측치를 날리는 것도 좋지만 가능하면 전처리를 하는 것이 좋다.

df\_nomiss2 <- na.omit(df)

df\_nomiss2



**## ifelse를 사용한 결측치 대체**

df$score <- ifelse(is.na(df$score), 4, df$score) **## ifelse로 결측치 대체 처리를 많이한다.**

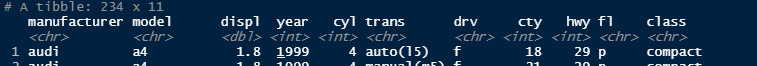
table(is.na(df$score)) **## 결측치 빈도표 생성 / table은 결측치 계의 count의 느낌**



install.packages("ggplot2")

library(ggplot2)

mpg <- as.data.frame(ggplot2::mpg) **# 패키지 안의 mpg 데이터 불러와서 df화 시키기**





mpg[c(65, 124, 131, 153, 212), "hwy"] <- NA **# 지정한 행과 열에 NA 할당하기**

mpg[c(65, 125, 131, 153)] **# 결과값 확인하기**

apply(mpg,2,function(x){sum(is.na(x))}) **## apply 이용하여 NA가 잘 들어가졌는지 확인**

mpg[is.na(mpg$hwy),"hwy"] <- mean(mpg$hwy,na.rm = T) **## mpg$hwy,na.rm = T의 평균을 앞**

**의 결과 NA 5개에 넣어줘 (다시체크)**

apply(mpg,2,function(x){sum(is.na(x))}) **## 그리고 다시 NA가 없는지 확인**

**## ZOO 패키지 (전처리를 특정값으로 하는 것이 아니라 앞의 것으로 하는 것임)**

install.packages("zoo")

library(zoo)

na.locf0(c(NA, NA, "A", NA, "B"), fromLast = FALSE) # 1

## 왼쪽 -> 오른쪽 ## 결측치가 있으면 왼쪽 것을 그대로 따라쓴다. (마지막부터(오른쪽)가 안됐으니까 왼쪽부터 시작했다고 생각하면 된다.

## 결과값 : [1] NA NA "A" "A" "B"

na.locf0(c(NA, NA, "A", NA, "B"), fromLast = TRUE) # 2

## 오른쪽 -> 왼쪽 # 결측치가 있으면 오른쪽 것을 그대로 따라쓴다.

## 결과값 [1] "A" "A" "A" "B" "B"

**####### 결측치 처리해보기!! #######**

**# 1. airquality 데이터의 결측치 개수를 구하시오 (열별로)**

**# apply, sapply, lapply, table, summary도 사용**

data(airquality) ## 내장데이터

head(airquality)

table(is.na(airquality))

apply(X = airquality, MARGIN = 2, FUN = function(X){sum(is.na(X))})

lapply(X = airquality, FUN = function(X){sum(is.na(X))})

sapply(X = airquality, FUN = function(X){sum(is.na(X))})

summary(is.na(airquality))

**# 2. 결측치가 있는 행들을 제거한 후 각 열의 평균을 구하시오**

airquality2 <- na.omit(airquality)

apply(airquality2,2,mean)

**# 3. 결측치는 변수의 중앙값으로 대체후 각 열의 평균을 출력하시오**

airquality[is.na(airquality$Ozone), "Ozone"] <- median(airquality$Ozone,na.rm=T)

airquality[is.na(airquality$Solar.R), "Solar.R"] <- median(airquality$Solar.R,na.rm=T)

apply(X=airquality, 2, mean)