

데이터 조작

데이터 조작

- 데이터 핸들링(Data Handling)
 - ✓ 수집한 원형 데이터를 데이터 분석의 목적에 맞게 데이터를 가공·처리하는 과정
 - ✓ 데이터 리모델링 관련 패키지 학습
 - ✓ 관련함수 : `dplyr`, `reshape2`

읽어보기

데이터 분석에서 가장 많은 시간을 차지하는 것은 데이터를 분석에 필요한 형태로 만드는 데이터 전처리 과정이다. 실제 데이터 분석 업무에서는 모델링이나 시각화에 적합한 형태의 데이터를 얻기 위해서는 복잡한 과정을 거치게 된다. 데이터 분석 프로젝트에 절반 이상의 시간이 데이터 전처리, 변환, 필터링 작업에 소요된다.

dplyr 패키지

- dplyr 패키지 활용
 - 데이터를 분석에 필요한 형태로 만드는 데이터 전처리 관련 함수 제공 패키지
 - dplyr 패키지의 주요 함수
 - ✓ `tbl_df()` : 데이터셋 화면창 크기 만큼만 데이터 제공
 - ✓ `filter()` : 지정한 조건식에 맞는 데이터 추출 - `subset()`
 - ✓ `select()` : 열의 추출 - `data[, c("Year", "Month")]`
 - ✓ `mutate()` : 열 추가 - `transform()`
 - ✓ `arrange()` : 정렬 - `order()`, `sort()`
 - ✓ `summarise()` : 집계 - `aggregate()`

Hadley Wickham 개발 패키지 : `ggplot2`, `plyr`, `reshape2`
`plyr`은 R로 개발, `dplyr`은 C++ 개발(빠른 처리 속도)

dplyr 패키지

- dplyr 패키지와 데이터 셋 hflight 설치

```
install.packages(c("dplyr", "hflights"))
```

```
library(dplyr)
```

```
library(hflights)
```

```
##### hflights 데이터셋 #####
```

```
# 2011년도 미국 휴스턴에서 출발하는 모든 비행기의
```

```
# 이착륙 기록이 수록된 것으로 227,496건의 이착륙 기록에
```

```
# 대해 21개 항목을 수집한 데이터
```

```
#####
```

dplyr 패키지

➤ `dim(hflights)` # 차원보기

```
# [1] 227496    21
```

➤ `tbl_df()` 함수 : 데이터셋 화면창 크기 만큼 데이터 제공

```
hflights_df <- tbl_df(hflights)
```

```
hflights_df
```

```
# Source: local data frame [227,496 x 21]
```

dplyr 패키지

1. filter(dataframe, 조건1, 조건2) 함수 이용 데이터 추출

1월 1일 데이터 추출

```
filter(hflights_df, Month == 1, DayofMonth == 1) # and(, or &)
```

Source: local data frame [552 x 21]

1월 혹은 2월 데이터 추출

```
filter(hflights_df, Month == 1 | Month == 2) # or(|)
```

Source: local data frame [36,038 x 21]

dplyr 패키지

2. arrange() 함수를 이용한 오름차순/내림차순 정렬

년도,월,도착시간 오름차순

```
arrange(hflights_df, Year, Month, ArrTime )
```

Month 기준 내림차순 정렬 - desc(변수)

```
arrange(hflights_df, desc(Month))
```

dplyr 패키지

3. select(), mutate() 함수를 이용한 열 조작

Year, Month, DayOfWeek 열을 추출

```
select(hflights_df, Year, Month, DayOfWeek)
```

Month기준 내림차순 정렬 -> Year, Month, AirTime, ArrDelay 컬럼 추출

```
select(arrange(hflights_df, desc(Month)), Year, Month, AirTime, ArrDelay)
```

Year~DayOfWeek 추출 (Year, Month, DayofMonth, DayofWeek)

```
select(hflights_df, Year:DayOfWeek)
```

Year부터 DayOfWeek를 제외한 나머지 열 추출

```
select(hflights_df, -(Year:DayOfWeek))
```


dplyr 패키지

4. mutate() 함수를 이용하여 열 추가(변형)

gain 변수 추가 -> gain_per_hour 변수 계산에 사용할 수 있음

```
mutate(hflights_df, gain = ArrDelay - DepDelay,
```

```
      gain_per_hour = gain/(AirTime/60))
```

새로 만든 열을 같은 함수 안에서 바로 사용 가능

```
.. ... ..
```

Variables not shown: ActualElapsedTime (int), AirTime (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr), Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int), CancellationCode (chr), Diverted (int), gain (int), gain_per_hour(dbl)

dplyr 패키지

새로 추가된 열을 같은 함수 안에서 select() 함수로 보기

주의 : 생성된 열은 hflights_df에 추가되지 않음 - 추가된 열 결과 보기

```
select(mutate(hflights_df, gain = ArrDelay - DepDelay,  
              gain_per_hour = gain/(AirTime/60)),  
       Year, Month, ArrDelay, DepDelay, gain, gain_per_hour)
```

#Source: local data frame [227,496 x 6]

	Year	Month	ArrDelay	DepDelay	gain	gain_per_hour
#1	2011	1	-10	0	-10	-15.000000
#2	2011	1	-9	1	-10	-13.333333

dplyr 패키지

5. summarise() 함수를 이용한 집계

n(), sum(), mean(), sd(), var(), median() 등의 함수 사용 - 기초 통계량

summarise(hflights_df, n()) # n() : dataframe의 row값 리턴 함수

평균 출발지연시간 계산

summarise(hflights_df, cnt = n(), delay=mean(DepDelay, na.rm = TRUE))

Source: local data frame [1 x 1] -> 결과는 data frame

cnt delay

#1 227496 9.444951

dplyr 패키지

6. group_by(dataframe, 기준변수) 함수를 이용한 그룹화

데이터 프레임을 대상으로 기준변수로 그룹화

예문) 항공기별로 비행편수가 20편 이상, 평균 비행 거리 2,000마일 이내의
평균 연착시간

1) 비행편수를 구하기 위해서 항공기별 그룹화

```
planes <- group_by(hflights_df, TailNum) # TailNum : 항공기 일련번호 그룹  
planes
```

2) 항공기별 필요한 변수 요약

```
planesInfo <- summarise(planes, count = n(), dist=mean(Distance, na.rm=T),  
                        delay=mean(ArrDelay, na.rm = T))
```

```
planesInfo
```

```
#Source: local data frame [3,320 x 4]
```

```
#   TailNum count    dist    delay  
#   항공기  비행편수 평균비행거리 평균연착시간  
#1      795    938.7157      NA  
#2  N0EGMQ    40    1095.2500  1.918919
```

dplyr 패키지

```
str(planesInfo) # Classes 'tbl_df'
# planesInfo 객체에 count, dist, delay 변수가 추가됨
```

3) planesInfo 객체를 대상으로 조건 지정

```
result <- filter(planesInfo, count > 20, dist < 2000)
```

result

Source: local data frame [1,526 x 4]

	TailNum	count	dist	delay
1		795	938.7157	NA
2	N0EGMQ	40	1095.2500	1.918919
3	N10156	317	801.7192	8.199357

reshape2 패키지

- reshape2 패키지 활용

- 긴 형식 <-> 넓은 형식
- reshape2 is a reboot of the reshape package.
- reboot : 기본골격만을 대상으로 패키지 개발
- reshape2 주요 함수
 - ✓ dcast() : 긴 형식 -> 넓은 형식으로 모양 변경
 - ✓ melt() : 넓은 형식 -> 긴 형식으로 모양 변경

reshape2 패키지

```
install.packages('reshape2')
```

```
library(reshape2)
```

```
# '긴 형식'(Long format)과 '넓은 형식'(wide format)으로 데이터 모양 변경
```

```
data <- read.csv("c:/workspaces/Rwork/data/data.csv")
```

```
data
```

```
# '넓은 형식'(wide format)으로 변형
```

```
wide <- dcast(data, Customer_ID ~ Date, sum)
```

```
# 형식) dcast(wide frame 변경 데이터셋, 앞변수~뒤변수, 함수)
```

```
# 해설) 데이터셋을 대상으로 앞변수와 뒤변수의 값이 같은 경우 함수 적용)
```

```
# 예) data 대상으로 Customer_ID와 Date변수가 같은 Buy변수에 합계 계산)
```

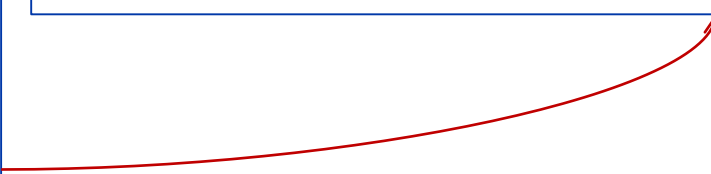
```
# Using Buy as value column: use value.var to override. - 정리 대상 변수(Buy)
```

```
wide # Buy 변수에 함수 적용 <- 정리대상 변수
```

reshape2 패키지

	Date	Customer_ID	Buy
1	20140101	1	3
2	20140101	2	4
3	20140102	1	2
4	20140102	4	6
5	20140102	5	1
6	20140103	1	5
7	20140103	2	1
8	20140103	4	8
9	20140103	5	5
10	20140104	1	5
11	20140104	2	8
12	20140104	3	5
13	20140105	5	6
14	20140106	2	6
15	20140106	3	6
16	20140107	1	9
17	20140107	5	7

	Customer_ID	20140101	20140102	20140103	20140104	20140105	20140106	20140107
1	1	3	2	5	5	0	0	9
2	2	4	0	1	8	0	6	0
3	3	0	0	0	5	0	6	0
4	4	0	6	8	0	0	0	0
5	5	0	1	5	0	6	0	7



reshape2 패키지

- melt() 함수 이용 : 넓은 형식 -> 긴 형식 변경

형식) melt(long frame 변경 데이터셋, id='열이름 변수')

해설) 데이터셋을 대상으로 id로 지정된 변수 기준으로 긴 형식 모양 변경)

```
x <- melt(wide, id='Customer_ID')
```

```
x
```

x객체의 컬럼명을 확인한 후 UserID, Date, Buy 이름으로 컬럼명 수정

```
colnames(x)
```

```
name <- c("UserID", "Date", "Buy")
```

```
colnames(x) <- name
```

```
head(x)
```