Lecture 03 - R: 데이터 가공하기

개요

- tidy data 이해하기
- dplyr 패키지로 데이터 가공하기

tidy data

- 문서 링크: tidy data
- 분석하기 좋은 데이터는 어떤 형태인가?

Codd's 3rd normal form

1. Each **variable** forms a **column**.
2. Each **observation** forms a **row**.
3. Each type of **observational unit** forms a **table**.

- 가장 중요한 것은 행과 열의 단위를 이해하는 것.
    - 도움이 되는 질문: '데이터를 처음 생성할 때 어떻게 기록을 시작했을까?'
    - 기존의 변수에 대해서 새로운 관찰을 하면, 열이 아니라 행이 늘어난다.
- 같은 내용을 담고 있어도 데이터 셋을 구성하는 방법은 매우 다양하다.
- 되도록 다양한 분석을 하는데 편한 형태로 정리해두는 것이 유리하다.

사례1: Column headers are values, not variable names.

messy data1

- 문제 상황: 열에 관찰 대상에 대한 변수(특성)가 아니라 관찰 값 자체가 들어 있다.

religion	<$10k	$10-20k	$20-30k	$30-40k	$40-50k	$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

tidy data1

- 해결 방안: 열에 있는 관찰 값을 대표하는 변수(income)을 새로 만들어서 관찰 값을 행으로 나열한다.

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

사례2: Column headers are values, not variable names.

messy data2

- 문제 상황: 주마다 관찰한 값을 열에 기록한다.
- 100주까지 기록하면 열을 100개로 확장?

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

tidy data2

- 해결 방안: 주(week)라는 열을 만들고 순위를 관찰할 때마다 행을 새로 생성한다.

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

사례3: Multiple variables are stored in one column

messy data3

- 문제 상황: 관찰 값이 열로 기록 되어 있고, 열 하나에 변수가 여러 개(성별과 나이) 합쳐져 있다.

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0	—	—
AE	2000	2	4	4	6	5	12	10	—	3
AF	2000	52	228	183	149	129	94	80	—	93
AG	2000	0	0	0	0	0	0	1	—	1
AL	2000	2	19	21	14	24	19	16	—	3
AM	2000	2	152	130	131	63	26	21	—	1
AN	2000	0	0	1	2	0	0	0	—	0
AO	2000	186	999	1003	912	482	312	194	—	247
AR	2000	97	278	594	402	419	368	330	—	121
AS	2000	—	—	—	—	1	1	—	—	—

tidy data3

- 해결 방안: 열에 기록된 관찰 값을 행으로 옮기고, 개별 변수(sex, age)로 나눈다.

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

(a) Molten data

country	year	sex	age	cases
AD	2000	m	0-14	0
AD	2000	m	15-24	0
AD	2000	m	25-34	1
AD	2000	m	35-44	0
AD	2000	m	45-54	0
AD	2000	m	55-64	0
AD	2000	m	65+	0
AE	2000	m	0-14	2
AE	2000	m	15-24	4
AE	2000	m	25-34	4
AE	2000	m	35-44	6
AE	2000	m	45-54	5
AE	2000	m	55-64	12
AE	2000	m	65+	10
AE	2000	f	0-14	3

(b) Tidy data

사례4: Variables are stored in both rows and columns

messy data4

- 문제 상황: 열과 행에 변수가 섞여 있다. (날짜, 최고온도, 최저 온도)

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

tidy data4

- 해결 방안: 행에 있던 관찰 값(date)을 열로 옮기고, 개별 변수(tmax, tmin) 분리

id	date	element	value
MX17004	2010-01-30	tmax	27.8
MX17004	2010-01-30	tmin	14.5
MX17004	2010-02-02	tmax	27.3
MX17004	2010-02-02	tmin	14.4
MX17004	2010-02-03	tmax	24.1
MX17004	2010-02-03	tmin	14.4
MX17004	2010-02-11	tmax	29.7
MX17004	2010-02-11	tmin	13.4
MX17004	2010-02-23	tmax	29.9
MX17004	2010-02-23	tmin	10.7

(a) Molten data

id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2	18.2

(b) Tidy data

사례5: Multiple types of observational units are stored in the same table

tidy data2

- 문제 상황: 다른 관찰 단위의 갑들이 하나의 테이블로 구성되어 있다.
- 관찰 단위1: 노래 정보, 관찰 단위2: 주별 순위

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

tidy data5

- 해결 방안: 노래 정보와 주별 순위 정보를 다른 테이블로 구성한다.

id	artist	track	time
1	2 Pac	Baby Don't Cry	4:22
2	2Ge+her	The Hardest Part Of ...	3:15
3	3 Doors Down	Kryptonite	3:53
4	3 Doors Down	Loser	4:24
5	504 Boyz	Wobble Wobble	3:35
6	98^0	Give Me Just One Nig...	3:24
7	A*Teens	Dancing Queen	3:44
8	Aaliyah	I Don't Wanna	4:15
9	Aaliyah	Try Again	4:03
10	Adams, Yolanda	Open My Heart	5:30
11	Adkins, Trace	More	3:05
12	Aguilera, Christina	Come On Over Baby	3:38
13	Aguilera, Christina	I Turn To You	4:00
14	Aguilera, Christina	What A Girl Wants	3:18
15	Alice Deejay	Better Off Alone	6:50

id	date	rank
1	2000-02-26	87
1	2000-03-04	82
1	2000-03-11	72
1	2000-03-18	77
1	2000-03-25	87
1	2000-04-01	94
1	2000-04-08	99
2	2000-09-02	91
2	2000-09-09	87
2	2000-09-16	92
3	2000-04-08	81
3	2000-04-15	70
3	2000-04-22	68
3	2000-04-29	67
3	2000-05-06	66

dplyr 패키지로 데이터 가공하기

왜 dplyr 패키지를 사용하나?

- 데이터 탐색과 가공이 매우 편리하다.
- chaining 문법으로 직관적인 코딩이 가능하다.
    - 파이프 연산 가능: '%>%'
    - 'then' operator
- 속도가 빠르다.

데이터 설명

- 데이터 파일 다운로드 링크: https://goo.gl/GiluLd

nycflights13

: On-time data for all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013.

format

- year,month,day: Date of departure
- dep_time,arr_time: Departure and arrival times, local tz.
- dep_delay,arr_delay: Departure and arrival delays, in minutes. Negative times represent early departures/arrivals.
- hour,minute: Time of departure broken in to hour and minutes
- carrier: Two letter carrier abbreviation.
- tailnum: Plane tail number
- flight: Flight number
- origin,dest: Origin and destination.
- air_time: Amount of time spent in the air
- distance: Distance flown

source

RITA, Bureau of transportation statistics,
http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236

분석을 위한 준비

1. 패키지 설치하고 불러오기
```
install.packages('dplyr')
library(dplyr)
```

2. 데이터 읽기
    a. 데이터 파일 확인
    ```
    filename <- list.files(path='.', pattern='*.csv')
    ```
    b. 데이터 파일 읽기
    ```
    flights <- read.csv(file=filename,
    ```

```
stringsAsFactors=FALSE)
```
중요!: stringsAsFactors=FALSE

c. 데이터 확인하기

```
head(flights)
str(flights)
summary(flights)
View(flights)
```

d. data.frame 변환하기

```
flights <- tbl_df(flights)
```

그림으로 데이터 탐색하기

1. 질문1: 출발 시간에 따라 도착 지연이 달라지나?

```
ggplot(flights, aes(x=factor(hour), y=dep_delay)) +
geom_boxplot()
```

2. 질문2: 비행 거리가 길면 도착 지연이 길어질까?

```
ggplot(flights, aes(x=distance, y=arr_delay)) +
geom_point(alpha=.1)
```

3. 질문3: 출발 지연이 길면 도착 지연도 길어질까?

```
ggplot(flights, aes(x=dep_delay, y=arr_delay)) +
geom_point(alpha=.1)
```

- 참고: http://docs.ggplot2.org/current/

dplyr의 주요 동사들

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

dplyr의 기본 문법

- DO(1st.argu, 2nd.argu, …)
- 1st. argument: data.frame
- 2nd. ~ argument: what to do with data.frame
- 결과는 항상 data.frame

fliter()

- 원하는 데이터 행을 선택

1. 1월 1일에 출발한 모든 비행기 기록을 조회
   ```
   filter(flights, month==1, day==1)
   ```

2. 1월이나 혹은 2월에 출발한 모든 비행기 기록을 조회
   ```
   filter(flights, month==1 | month==2)
   ```

3. To SFO or OAK
   ```
   filter(flights, dest %in% c('SFO', 'OAK'))
   ```

4. 데이터의 1행부터 10행까지 조회
   ```
   slice(flights, 1:10)
   ```

   Q. 출발 지연이 60분 이상인 비행 기록
   Q. 도착 지연이 출발 지연의 2배를 초과하는 비행 기록

arrange()

- 데이터 행의 순서를 원하는 기준으로 정렬

1. 연, 월, 일 순서로 정렬
   ```
   arrange(flights, year, month, day)
   ```

2. 도착 지연 내림차순 정렬
   ```
   arrange(flights, desc(arr_delay))
   ```

   Q. 비행 중 가장 시간을 많이 따라잡은 비행 기록

select()

- 원하는 데이터 열을 선택

1. 연, 월, 일 열을 선택
   ```
   select(flights, year, month, day)
   ```

2. year부터 day까지 모든 열 선택
   ```
   select(flights, year:day)
   ```

3. year부터 day까지 제외한 모든 열 선택
   ```
   select(flights, -(year:day))
   ```

4. 열의 이름을 바꾸면서 선택
   ```
   select(flights, tail_num = tailnum)
   ```

5. 열의 이름만 바꾸기
```
rename(flights, tail_num = tailnum)
```

distinct()

- 유일한(unique) 행만 추출

1. 유일한 tailnum
```
distinct(select(flights, tailnum))
```

2. 유일한 항공로
```
distinct(select(flights, origin, dest))
```

mutate()

- 새로운 열을 만들기

1. 시간 증가, 비행 속도 열 만들기
```
mutate(flights,
       gain = arr_delay - dep_delay,
       speed = distance / air_time * 60)
```

2. 시간 당 시간 증가 열 만들기
```
mutate(flights,
       gain = arr_delay - dep_delay,
       gain_per_hour = gain / (air_time / 60)
```

Q. km/h 속도 계산하기

summarise()

- 행 하나로 요약하기

1. 평균 지연 시간
```
summarise(flights,
       delay = mean(dep_delay, na.rm = TRUE))
```

sample_n(), sample_frac()

- 무작위 표본 추출하기

1. 10개의 비행 기록 표본 추출하기
```
sample_n(flights, 10)
```

2. 전체의 1% 규모 표본 추출하기
```
sample_frac(flights, .01)
```

Grouped summarise

- 그룹을 지정하면 해당 그룹 내에서 요약 함수를 사용할 수 있다.

1. 개별 비행기의 운행 횟수, 평균 운항 거리, 평균 도착 지연 시간을 계산

```
by_tailnum <- group_by(flights, tailnum)
delay <- summarise(by_tailnum,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE))
```

   a. 운항 횟수가 20회 초과하고, 평균 운항 거리가 2천 마일 미만인 비행기

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

   b. 평균 운항 거리와 평균 도착 지연은 관계가 있을까?

```
ggplot(delay, aes(dist, delay)) +
      geom_point(aes(size = count), alpha = 1/2) +
      geom_smooth() +
      scale_size_area()
```

2. 날짜 별로 평균 출발 지연, 출발 지연 중위수, 상위 75분위수, 15분 이상 평균 출발 지연, 30분 이상 평균 출발 지연, 60분 이상 평균 출발 지연

```
flights <- mutate(flights, date = as.Date(paste(year, month,
day, sep='.'), format='%Y.%m.%d'))
by_date <- group_by(flights, date)
delays <- summarise(by_date,
   mean = mean(dep_delay, na.rm = TRUE),
   median = median(dep_delay, na.rm = TRUE),
   q75 = quantile(dep_delay, 0.75, na.rm = TRUE),
   over_15 = mean(dep_delay > 15, na.rm = TRUE),
   over_30 = mean(dep_delay > 30, na.rm = TRUE),
   over_60 = mean(dep_delay > 60, na.rm = TRUE))
```

3. 착륙 공항에 따른 유일한 비행기 숫자와 비행편 수 집계

```
destinations <- group_by(flights, dest)
summarise(destinations,
  planes = n_distinct(tailnum),
  flights = n())
```

4. 일별, 월별, 연별 비행편 수 집계

```
daily <- group_by(flights, year, month, day)
(per_day   <- summarise(daily, flights = n()))
(per_month <- summarise(per_day, flights = sum(flights)))
(per_year  <- summarise(per_month, flights = sum(flights)))
```

   - 여러 개의 변수에 그룹을 지정하면, summarise() 함수를 한 번 쓸 때마다 그룹 지정이 하나씩 제거된다.

Pipeline operator(chaining)

원하는 데이터

- 일별 평균 도착 지연과 평균 출발 지연을 구하고, 평균 도착 지연과 평균 출발 지연 시간이 30분 초과인 날짜만 추출

해결 1: 단계별로 저장하기(multiple assignment)

```
a1 <- group_by(flights, year, month, day)
a2 <- select(a1, arr_delay, dep_delay)
a3 <- summarise(a2,
  arr = mean(arr_delay, na.rm = TRUE),
  dep = mean(dep_delay, na.rm = TRUE))
a4 <- filter(a3, arr > 30 | dep > 30)
```

- 문제: 필요없는 변수를 만들어낸다. a1, a2, a3은 어디에 쓰나?

해결 2: 감싸기(neting)

```
filter(
  summarise(
    select(
      group_by(flights, year, month, day),
      arr_delay, dep_delay
    ),
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)
  ),
  arr > 30 | dep > 30
)
```

- 문제: 보기 어렵다. 작성하기 어렵다. 코딩 순서와 생각의 순서가 다르다.

the pipe operator: '%>%'

- x %>% f(y) -> f(x, y)
- dplyr에서는 데이터를 왼쪽에서 오른쪽으로 연결한다.

해결 3: 파이핑(pipe operator)

```
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarise(arr = mean(arr_delay, na.rm = TRUE),
            dep = mean(dep_delay, na.rm = TRUE)) %>%
  filter(arr > 30 | dep > 30)
```

- 장점: 사고의 순서와 코딩의 순서가 같다. 생각 나는대로 적으면 코드가 완성 된다.
- 단축키: Cmd+Shift+M, Ctrl+Shift+M

Grouped mutate / filter

그룹을 지정했을 때, 요약과 변수 생성의 차이점

- Grouped summarise
  - n rows -> 1 rows by a group
- Grouped mutate / filter
  - n rows -> n rows by a group

각 비행기의 평균 지연 시간1

```
flights %>%
    filter(tailnum != '') %>%
    group_by(tailnum) %>%
    summarise(arr = mean(arr_delay, na.rm = T),
              dep = mean(dep_delay, na.rm = T))
```

각 비행기의 평균 지연 시간2

```
flights %>%
    filter(tailnum != '') %>%
    group_by(tailnum) %>%
    mutate(arr = mean(arr_delay, na.rm = T),
           dep = mean(dep_delay, na.rm = T)) %>%
    select(tailnum, arr_delay, arr, dep_delay, dep) %>%
    arrange(desc(tailnum))
```

참고

dplyr vignettes

- window functions - http://cran.rstudio.com/web/packages/dplyr/vignettes/window-functions.html
- two table verbs - http://cran.rstudio.com/web/packages/dplyr/vignettes/two-table.html
- database - http://cran.rstudio.com/web/packages/dplyr/vignettes/databases.html

code guide

- Google's R Style Guide - http://google-styleguide.googlecode.com/svn/trunk/Rguide.xml