5장 데이터 가공

02.

1 gapminder 데이터 셋의 관측 기간 내 (1952~2007) 대한민국 인구의 최대치와 해당 연도를 출력하라. (4장에서 배운 max 함수를 사용하라.)

```
> gapminder %>% filter(country=="Korea, Rep.") %>% select(pop) %>% max()
[1] 49044790
> gapminder %>% filter(country=="Korea, Rep.") %>% filter(pop==49044790)
# A tibble: 1 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Korea, Rep. Asia       2007    78.6 49044790    23348.
```

2 2007년도 아시아 대륙의 인구 총합을 구하라.

```
> gapminder %>% filter(year==2007 & continent=="Asia") %>%
select(pop) %>% sum()
[1] 3811953827
```

03.

1 대한민국, 중국, 일본 세 나라의 1인당 국내총생산과 기대 수명을 전체 관측 기간에 걸쳐 나란히 출력하라.

```
> gapminder %>% filter(country == "Korea, Rep.") %>% select( year, country, gdpPercap, lifeExp)
# A tibble: 12 x 4
    year country       gdpPercap lifeExp
   <int> <fct>             <dbl>   <dbl>
 1  1952 Korea, Rep.       1031.    47.5
 2  1957 Korea, Rep.       1488.    52.7
 3  1962 Korea, Rep.       1536.    55.3
 4  1967 Korea, Rep.       2029.    57.7
 5  1972 Korea, Rep.       3031.    62.6
 6  1977 Korea, Rep.       4657.    64.8
 7  1982 Korea, Rep.       5623.    67.1
 8  1987 Korea, Rep.       8533.    69.8
 9  1992 Korea, Rep.      12104.    72.2
10  1997 Korea, Rep.      15994.    74.6
11  2002 Korea, Rep.      19234.    77.0
12  2007 Korea, Rep.      23348.    78.6
> gapminder %>% filter(country == "China") %>% select( year, country, gdpPercap, lifeExp)
# A tibble: 12 x 4
    year country gdpPercap lifeExp
   <int> <fct>       <dbl>   <dbl>
 1  1952 China         400.    44
 2  1957 China         576.    50.5
```

```
 3   1962 China          488.     44.5
 4   1967 China          613.     58.4
 5   1972 China          677.     63.1
 6   1977 China          741.     64.0
 7   1982 China          962.     65.5
 8   1987 China         1379.     67.3
 9   1992 China         1656.     68.7
10   1997 China         2289.     70.4
11   2002 China         3119.     72.0
12   2007 China         4959.     73.0
> gapminder %>% filter(country == "Japan") %>% select( year, country,
gdpPercap, lifeExp)
# A tibble: 12 x 4
      year country gdpPercap lifeExp
     <int> <fct>        <dbl>   <dbl>
 1   1952 Japan        3217.     63.0
 2   1957 Japan        4318.     65.5
 3   1962 Japan        6577.     68.7
 4   1967 Japan        9848.     71.4
 5   1972 Japan       14779.     73.4
 6   1977 Japan       16610.     75.4
 7   1982 Japan       19384.     77.1
 8   1987 Japan       22376.     78.7
 9   1992 Japan       26825.     79.4
10   1997 Japan       28817.     80.7
11   2002 Japan       28605.     82
12   2007 Japan       31656.     82.6
```

2 아프리카 대륙의 총인구가 유럽의 총인구보다 많았던 해를 모두 구하라.

```
> gapminder %>% filter(continent == "Africa") %>% group_by(year) %>%
summarize(s=sum(pop)) -> s1
> gapminder %>% filter(continent == "Europe") %>% group_by(year) %>%
summarize(s=sum(pop)) -> s2
> s1$s > s2$s
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
TRUE
> s1[s1$s > s2$s, "year"]
# A tibble: 5 x 1
    year
   <int>
1  1987
2  1992
3  1997
4  2002
5  2007
```

3 gapminder 라이브러리에는 gapminder로 가공되기 전의 gapminder_unfiltered라는 데이터 셋이 포함되어 있다. gapminder의 관측 횟수(12회) 이상의 샘플이 기록된 국가 country를 샘플 개수가 많은 순서대로 추출하라.

```
> gapminder_unfiltered %>% group_by(country) %>% summarize(n=n()) %>%
filter(n>12) %>%   arrange(desc(n))
# A tibble: 45 x 2
   country                n
   <fct>              <int>
```

1	Czech Republic	58
2	Denmark	58
3	Finland	58
4	Iceland	58
5	Japan	58
6	Netherlands	58
7	Norway	58
8	Portugal	58
9	Slovak Republic	58
10	Spain	58
11	Sweden	58
12	Switzerland	58
13	Taiwan	58
14	Austria	57
15	Belgium	57
16	Bulgaria	57
17	Canada	57
18	France	57
19	Hungary	57
20	United States	57
21	Australia	56
22	Italy	56
23	New Zealand	55
24	Poland	52
25	Luxembourg	49
26	Latvia	42
27	China	36
28	Slovenia	32
29	Germany	26

30	Russia	20
31	Ukraine	20
32	Belarus	18
33	Estonia	18
34	Lithuania	18
35	Costa Rica	13
36	Cuba	13
37	Greece	13
38	Ireland	13
39	Libya	13
40	Mexico	13
41	Puerto Rico	13
42	Sri Lanka	13
43	Thailand	13
44	Uganda	13
45	United Kingdom	13

04.

1 앞의 예에서 연도 속성에 포함된 X 문자를 제거하는 과정에서 names(elec_gen)=substr (names(elec_gen), 2, nchar(names(elec_gen))) 명령어는 약간의 의도치 않은 문제를 야기하였다. 결과에서 이 문제를 찾아보고 해결 방안을 생각해보라.

p. 172 의 코드 예에서 names(elec_gen) = substr(names(elec_gen), 2, nchar(names(elec_gen))) 명령에 의해 첫번째 열 이름인 "country" 가 "ountry" 가 되는 문제가 발생하였다.
names(elec_gen) 대신 첫번째 원소를 제외한 names(elec_gen)[2: length(names(elec_gen))] 를 사용하면 이런 문제를 방지할 수 있고, 여기서 열 이름의 개수는 length(names(elec_gen)) 을 사용하여 알 수 있다.

2 gapminder 웹 사이트에서 [Education]-[Literacy] 메뉴에 있는 6종류의 관측 데이터 셋을 다운로드한 다음, 결측값을 제거한 후 173~174쪽 예와 같은 방법으로(각 항목을 열 에 할당한 형태로) 병합하라.

```
> laf =
read.csv("literacy_rate_adult_female_percent_of_females_ages_15_above.csv",
header=TRUE, sep=",")
> lam =
read.csv("literacy_rate_adult_male_percent_of_males_ages_15_and_above.csv",
header=TRUE, sep=",")
> lat =
read.csv("literacy_rate_adult_total_percent_of_people_ages_15_and_above.csv",
header=TRUE, sep=",")
> lyf = read.csv("literacy_rate_youth_female_percent_of_females_ages_15_24.csv",
header=TRUE, sep=",")
```

```
> lym = read.csv("literacy_rate_youth_male_percent_of_males_ages_15_24.csv", header=TRUE, sep=",")
> lyt = read.csv("literacy_rate_youth_total_percent_of_people_ages_15_24.csv", header=TRUE, sep=",")

> laf_tidy=gather(laf, -country, key="year", value="adult_female")
> lam_tidy=gather(lam, -country, key="year", value="adult_male")
> lat_tidy=gather(lat, -country, key="year", value="adult_total")
> lyf_tidy=gather(lyf, -country, key="year", value="youth_female")
> lym_tidy=gather(lym, -country, key="year", value="youth_male")
> lyt_tidy=gather(lyt, -country, key="year", value="youth_total")

> literacy = merge(laf_tidy, lam_tidy)
> literacy = merge(literacy, lam_tidy)
> literacy = merge(literacy, lat_tidy)
> literacy = merge(literacy, lyf_tidy)
> literacy = merge(literacy, lym_tidy)
> literacy = merge(literacy, lyt_tidy)
> literacy = na.omit(literacy)
```

주의할 점: 일반적인 데이터프레임의 형태에서는 na.omit 을 사용하여 결측값을 제거한 후 가공을 하는 것이 보통이지만, 이 경우에는 데이터의 구조를 변경한 이후에 na.omit 을 사용해야 한다.

6장 데이터 시각화

01.

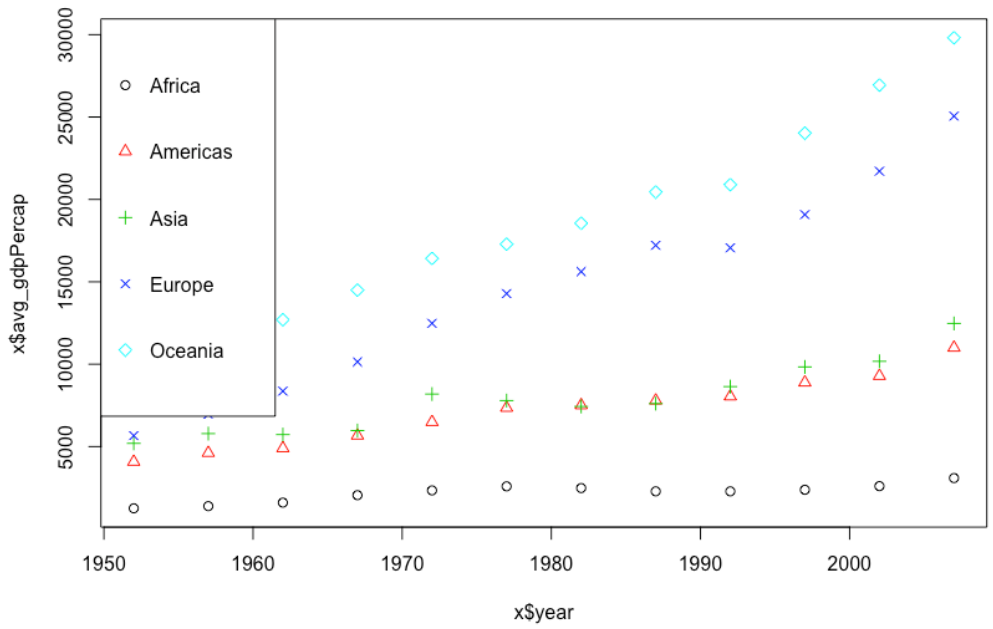
1 gapminder 데이터에서 각 대륙의 gdpPercap 의 평균치를 플롯하고 범례를 추가하라.

먼저 데이터 가공을 통해 각 대륙의 연도별 gdpPercap 의 평균치를 구하여 x 라는 변수에 저장한다.

```
> gapminder %>% group_by(year, continent) %>% summarize(avg_gdpPercap = mean(gdpPercap)) -> x
```

plot 와 legend 함수를 이용해 연도별 변화를 플롯한다.

```
> plot(x$year, x$avg_gdpPercap, col=x$continent, pch=c(1:length(levels(x$continent))))
> legend("topleft", legend=levels(x$continent), col=c(1:length(levels(x$continent))), pch=c(1:length(levels(x$continent))))
```



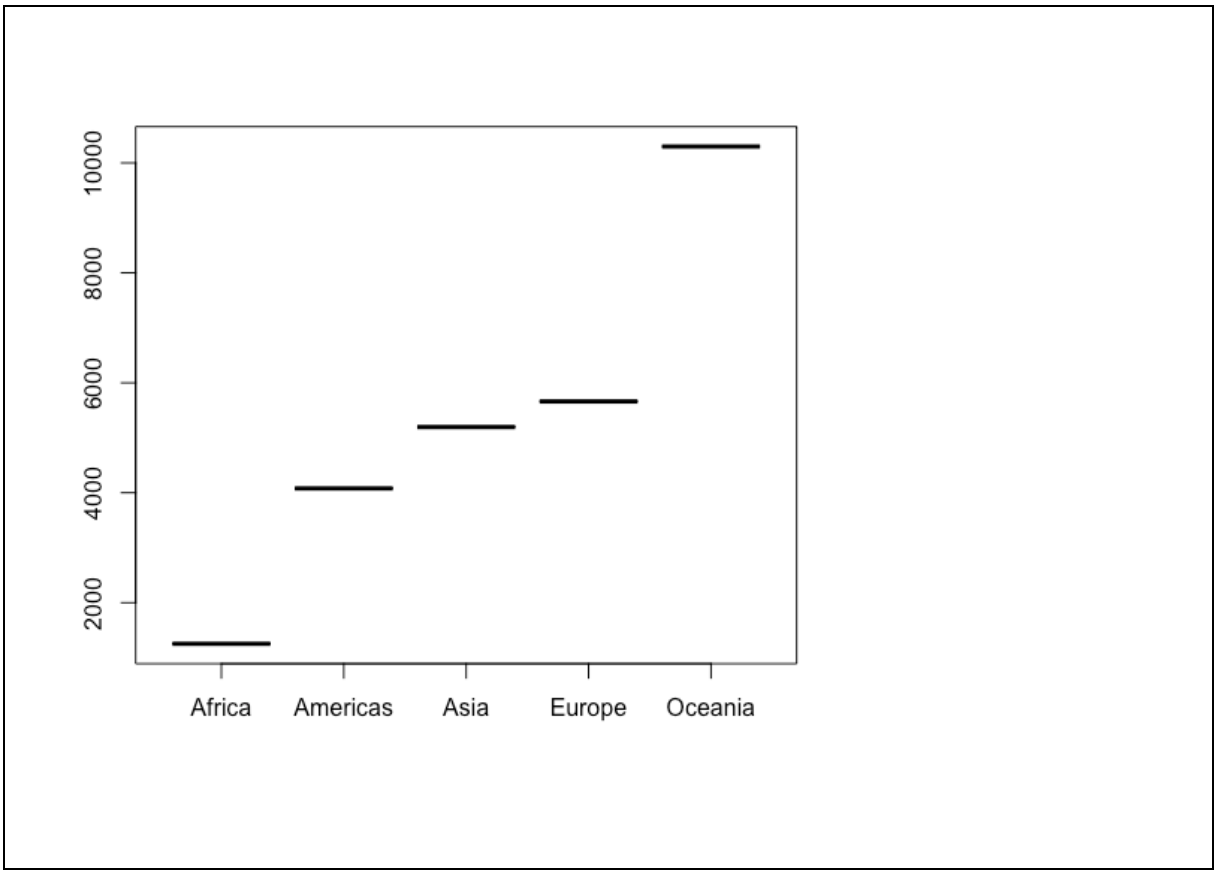
2 gapminder 데이터에서 1952년의 gdpPercap과 lifeExp의 대륙별 평균을 추출한 후, 가로축에는 gdpPercap, 세로축에는 lifeExp를 나타낸 그래프로 시각화하라.

먼저 데이터 가공을 통해 1952년의 gdpPercap과 lifeExp의 대륙별 평균을 추출한 후 x 라는 변수에 저장한다.

```
> gapminder %>% filter(year==1952) %>% group_by(continent) %>%
summarise(m=mean(gdpPercap)) -> x
> x
# A tibble: 5 x 2
  continent         m
  <fct>         <dbl>
1 Africa        1253.
2 Americas      4079.
3 Asia          5195.
4 Europe        5661.
5 Oceania      10298.
```

plot 함수를 이용해 시각화한다.

```
> plot(x$continent, x$m)
```


2000
4000
6000
8000
10000
Africa
Americas
Asia
Europe
Oceania

02.

1 다음 명령어로 시각화한 결과를 [그림 6-13]과 비교하고 차이가 발생하는 이유를 설명하라.

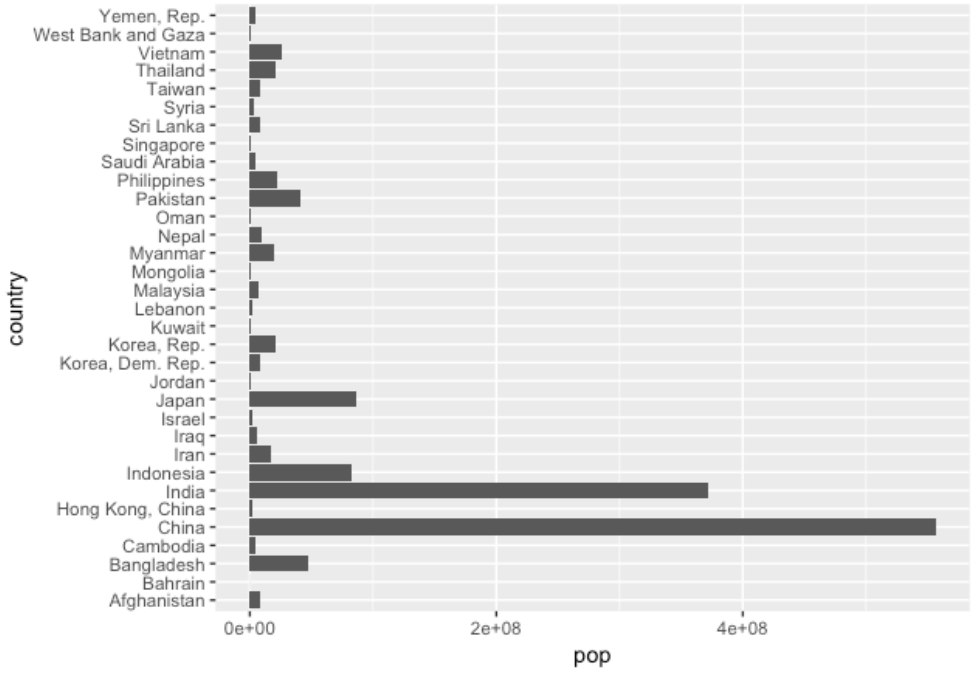
그림 6-13 은 다음 명령어에 의한 시각화이다.

>gapminder%>%filter(year==1952&continent=="Asia")%>%ggplot(aes(reorder(coun try, pop), pop))+geom_bar(stat="identity")+coord_flip()

여기서 reorder(country, pop) 은 country 를 pop 의 크기에 따라 재배열하라는 의미이므로, 그림 6-13 과 같이 그래프의 x 축에 pop 의 크기 순으로 정렬된 country 가 지정되게 된다.

그에 비해, 주어진 명령어는 country 를 재배열하지 않는 시각화이므로 다음과 같은 시각화 결과를 얻게 된다.

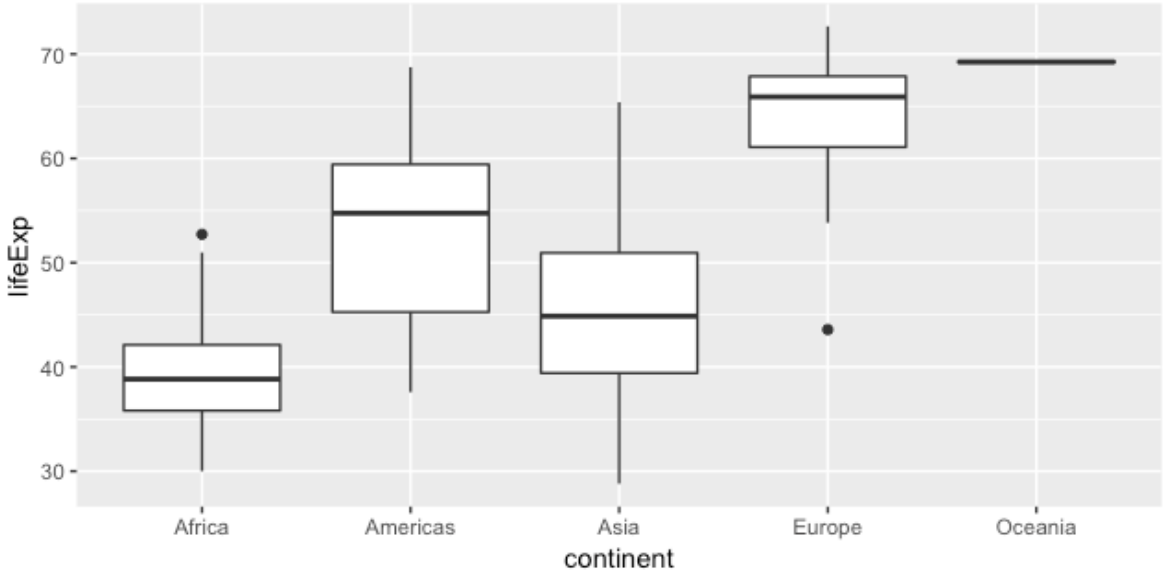
> gapminder %>% filter(year == 1952 & continent == "Asia") %>% ggplot(aes(country, pop))+geom_bar(stat="identity")+coord_flip()



2 다음 명령어로 시각화한 결과를 [그림 6-19]와 비교하라.

그림 6-19 는 다음 명령어에 의한 시각화이다 .

```
x = filter(gapminder, year == 1952)
x %>% ggplot(aes(continent, lifeExp)) + geom_boxplot()
```



주어진 명령어와 시각화 결과는 아래와 같다.

```
ggplot(gapminder, aes(continent, lifeExp)) + geom_point(alpha=0.2, size= 1.0,
position="jitter")
```

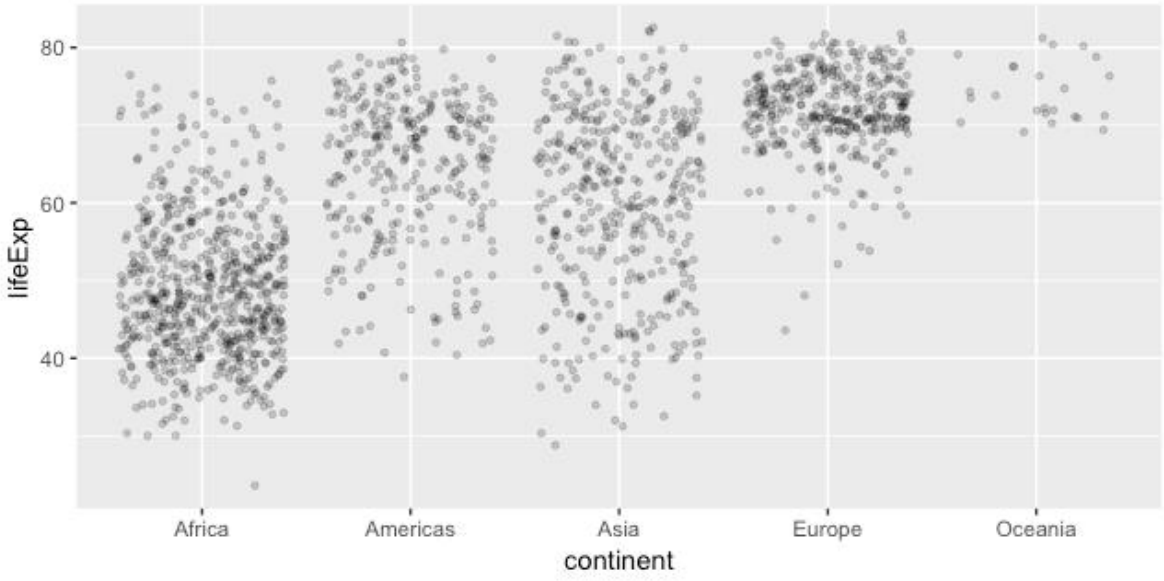
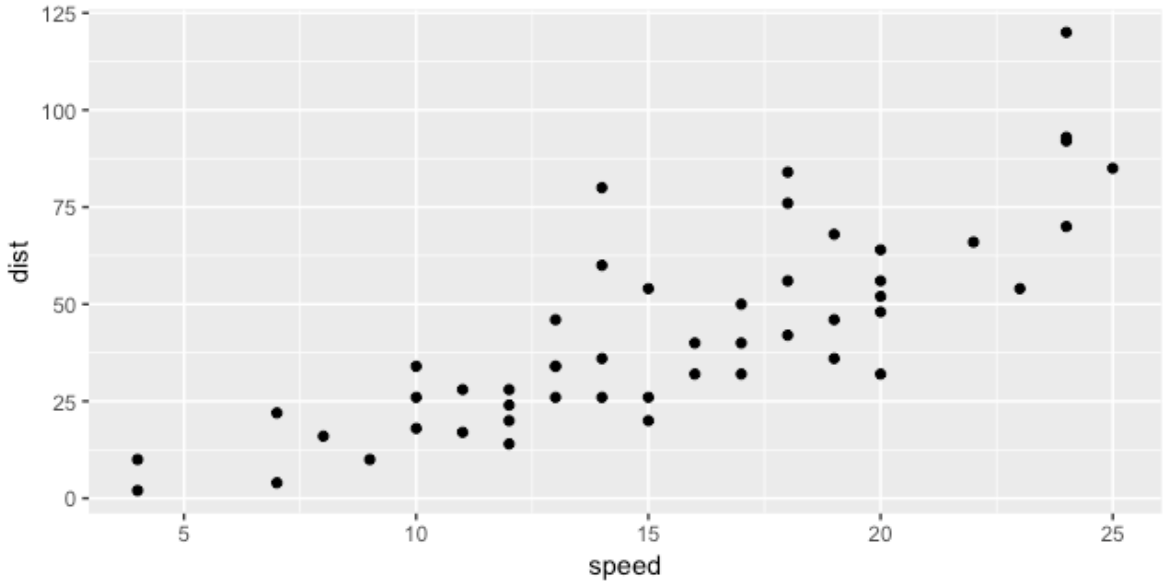


그림 6-19 가 1952년도의 대륙별 기대수명 데이터를 통계적으로 요약한 관점에서 시각화한 결과라면,
주어진 명령어는 같은 데이터를 시각화하되, 개별 데이터의 분포를 사실적으로 확인할 수 있도록 보여준 결과라고 할 수 있다.

03.

1 베이스 R의 plot 함수를 이용한 [그림 6-21]의 그래프를 ggplot2 라이브러리를 이용해 그려라.

```
> cars %>% ggplot(aes(speed, dist)) + geom_point()
```



2 베이스 R의 matplot 함수를 이용한 [그림 6-27]의 그래프를 ggplot2 라이브러리를 이용 해 그려라.

주의할 점: 데이터의 구조 변경이 먼저 이루어져야 ggplot 에 의해 그래프를 그릴 수 있다. (ggplot 에서 멀티플롯을 그리기는 의외로 번거롭다)
우선 그림 6-27 과 같이 가로축에 iris 샘플을 지정하기 위해 id 라는 열을 새로 추가한다.

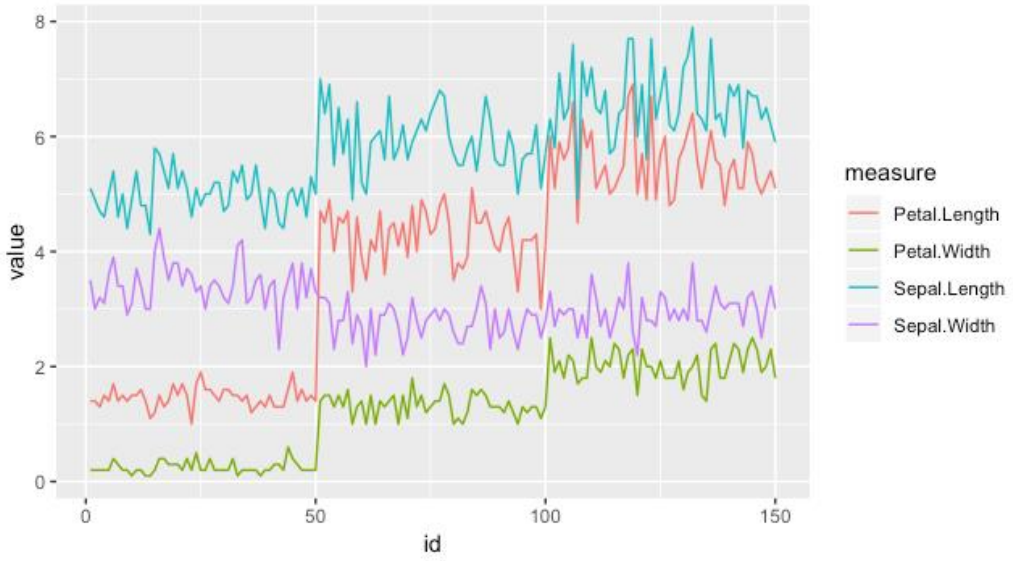
```
> id = 1:150
> iris1 = cbind(id, iris1)
```

그 다음 샘플 1개당 존재하는 4개의 관측 변수를 하나의 행으로 분리하여 데이터의 구조를 변경한다.

```
> iris_tidy = gather(iris1, -id, -Species, key="measure", value="value")
Species         measure value
1          setosa Sepal.Length     5.1
2          setosa Sepal.Length     4.9
3          setosa Sepal.Length     4.7
4          setosa Sepal.Length     4.6
5          setosa Sepal.Length     5.0
6          setosa Sepal.Length     5.4
……
```

이제 ggplot 을 이용하여 x축에 id 를, y축에 관측치를 지정하고, 4개의 변수를 col 로 구분하여 별도의 선그래프로 그린다.

```
> iris_tidy %>% ggplot(aes(id, value, col=measure)) + geom_line()
```



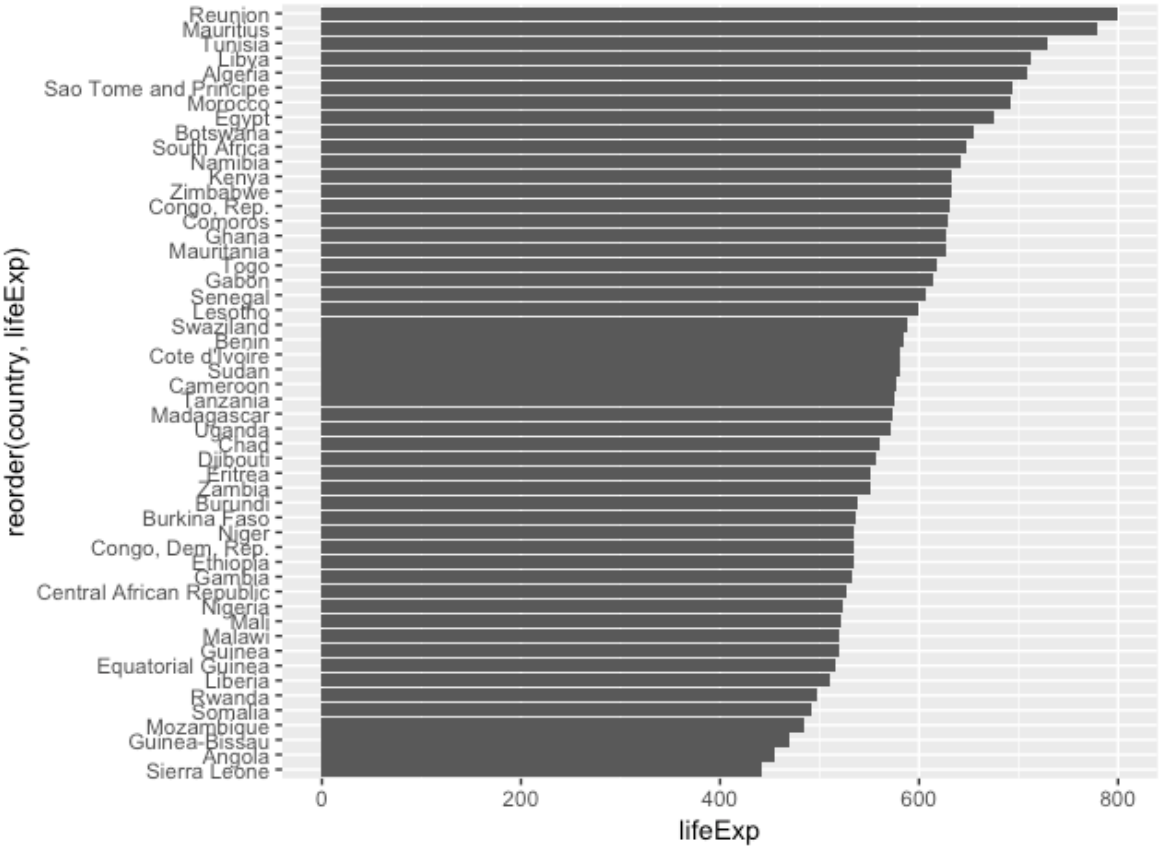
3 [그림 6-35(b)]의 막대그래프를 lifeExp가 큰 순서대로(내림차순으로) 정렬하여 그려라.

그림 6-359(b) 는 아래의 명령어에 의한 시각화 결과이다.

> gapminder %>% filter(continent == "Africa") %>% ggplot(aes(country, lifeExp)) + geom_bar(stat = "identity") + coord_flip()

x축 (coord_flip 함수에 의해 수직축으로 표시되었다) 에 지정된 country 를 lifeExp 의 크기순으로 재배열해야 하므로 다음과 같이 reorder 함수를 이용한다.

> gapminder %>% filter(continent == "Africa") %>% ggplot(aes(reorder(country, lifeExp), lifeExp)) + geom_bar(stat = "identity") + coord_flip()



04.

1 [그림 6-43(a)]에 사용된 다음 명령어는 filter 함수 내의 논리식 결합으로 다소 길다. %in% 연산자를 사용해 간략하게 바꿔보라.

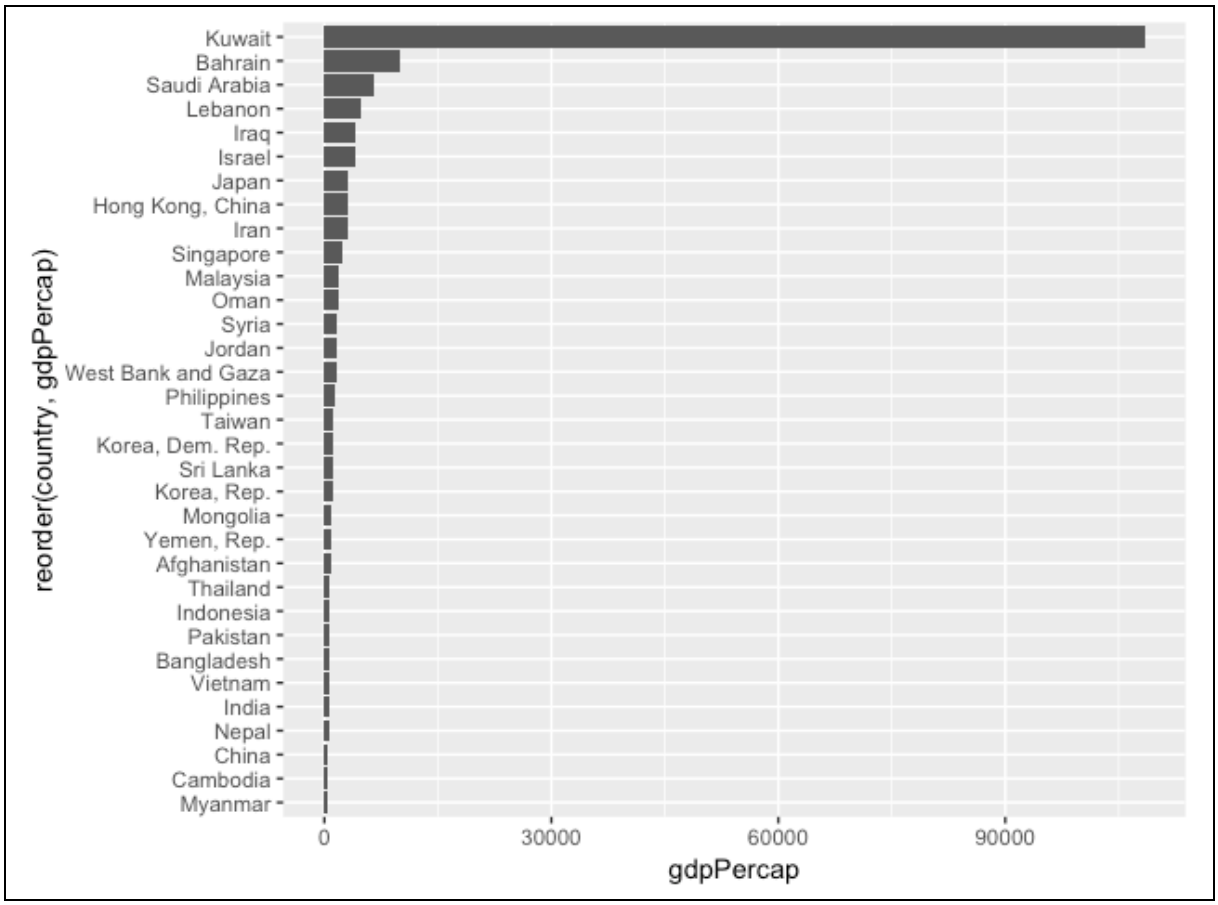
```
gapminder%>%filter(country=="Kuwait"|country=="Saudi Arabia"|country==
"Iraq"|country=="Iran"|country=="Korea, Rep."|country=="China"|country==
"Japan") %>% ggplot(aes(year, gdpPercap, col = country)) + geom_point() +
geom_ line()
```

%in% 는 특수연산자로 벡터 내 특정 값 포함 여부를 확인할 때 사용할 수 있는 연산자이다. %in% 를 이용해 위의 명령어의 논리식 부분을 간략히 표현할 수 있다.

```
gapminder%>%filter(country %in% c("Kuwait", "Saudi Arabia",  "Iraq", "Iran",
"Korea, Rep.", "China"," Japan") %>% ggplot(aes(year, gdpPercap, col =
country)) + geom_point() + geom_ line()
```

2 [그림 6-39]의 1952년도 그래프에 나타난 데이터를 사용해 해당 연도 아시아 국가들의 gdpPercap 순위를 막대그래프로 시각화하라.

```
> gapminder %>% filter(continent=="Asia"&year==1952) %>%
ggplot(aes(reorder(country, gdpPercap), gdpPercap)) +
geom_bar(stat="identity") + coord_flip()
```


Kuwait
Bahrain
Saudi Arabia
Lebanon
Iraq
Israel
Japan
Hong Kong, China
Iran
Singapore
Malaysia
Oman
Syria
Jordan
West Bank and Gaza
Philippines
Taiwan
Korea, Dem. Rep.
Sri Lanka
Korea, Rep.
Mongolia
Yemen, Rep.
Afghanistan
Thailand
Indonesia
Pakistan
Bangladesh
Vietnam
India
Nepal
China
Cambodia
Myanmar
reorder(country, gdpPercap)
0
30000
60000
90000
gdpPercap

12장 프로젝트

02.

1 데이터 정제 과정에서 삭제한 원본 데이터의 10,473행에 어떤 문제가 있었는지 구체적으로 확인하라.

원본 데이터를 View 명령으로 확인해보면,

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content.Rating
10468	FI CFL	FINANCE	3.7	112	3.9M	10,000+	Free	0	Everyone
10469	Tassa.fi Finland	LIFESTYLE	3.6	346	7.5M	50,000+	Free	0	Everyone
10470	TownWiFi | Wi-Fi Everywhere	COMMUNICATION	3.9	2372	58M	500,000+	Free	0	Everyone
10471	Jazz Wi-Fi	COMMUNICATION	3.4	49	4.0M	10,000+	Free	0	Everyone
10472	Xposed Wi-Fi-Pwd	PERSONALIZATION	3.5	1042	404k	100,000+	Free	0	Everyone
10473	Life Made WI-Fi Touchscreen Photo Frame	1.9	19.0	3.0M	1,000+	Free	0	Everyone	
10474	osmino Wi-Fi: free WiFi	TOOLS	4.2	134203	4.1M	10,000,000+	Free	0	Everyone
10475	Sat-Fi Voice	COMMUNICATION	3.4	37	14M	1,000+	Free	0	Everyone
10476	Wi-Fi Visualizer	TOOLS	3.9	132	2.6M	50,000+	Free	0	Everyone
10477	Lennox iComfort Wi-Fi	LIFESTYLE	3.0	552	7.6M	50,000+	Free	0	Everyone
10478	Sci-Fi Sounds and Ringtones	PERSONALIZATION	3.6	128	11M	10,000+	Free	0	Everyone

그림과 같이 10,473 행의 데이터는 Category 데이터가 없고, 그 이후의 속성(Rating, Reviews, Size 등)이 한 칸씩 당겨져 채워져 있는 것을 볼 수 있다. 이대로 데이터를 처리하게 되면 다른 속성들의 데이터형의 일관성에 문제가 발생한다.

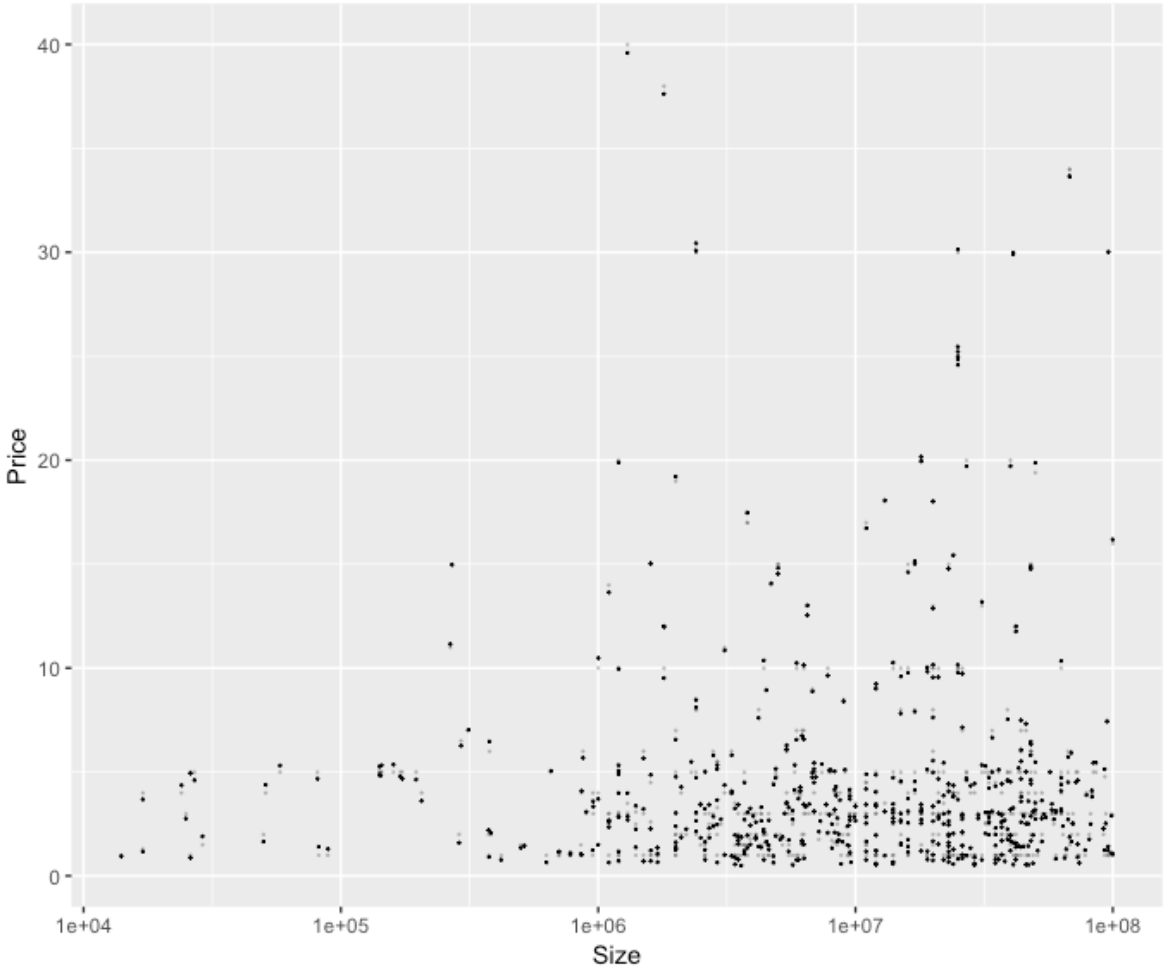
2 숫자형 데이터가 기록되어야 하는 열에 문자가 포함된 경우(Mega byte를 의미하는 M 등) 적절히 찾아 처리하지 않으면 어떤 문제가 발생하는가?

숫자형 데이터로 가정하고 그대로 처리하면 R은 이를 제대로 인식하지 못할 뿐만 아니라, Mega byte 에 해당하는 10의 6제곱 등을 처리하지 않은 상태에서는 정확한 수치로 처리할 수도 없게 된다.

03.

1 앱의 크기는 가격과 어떤 관계가 있는가?

```
> x %>% filter(Type=="Paid" & Price<60) %>% ggplot(aes(Size, Price)) +
geom_point(alpha=0.2, size=0.1) + geom_jitter(size=0.1, height  = 0.5) +
scale_x_log10()
```



1M byte 이하의 앱은 가격이 대부분 10$ 미만이지만, 많은 앱이 분포하고 있는 1M byte ~ 100 Mbyte 구간에서는 가격도 같이 증가하고 있는 것을 볼 수 있다. 높은 가격의 앱들의 개수도 점차로 증가하였다. 단 Size 축이 로그스케일이므로, Size 의 증가에 대해 가격의 증가는 상대적으로 작게 나타난다.

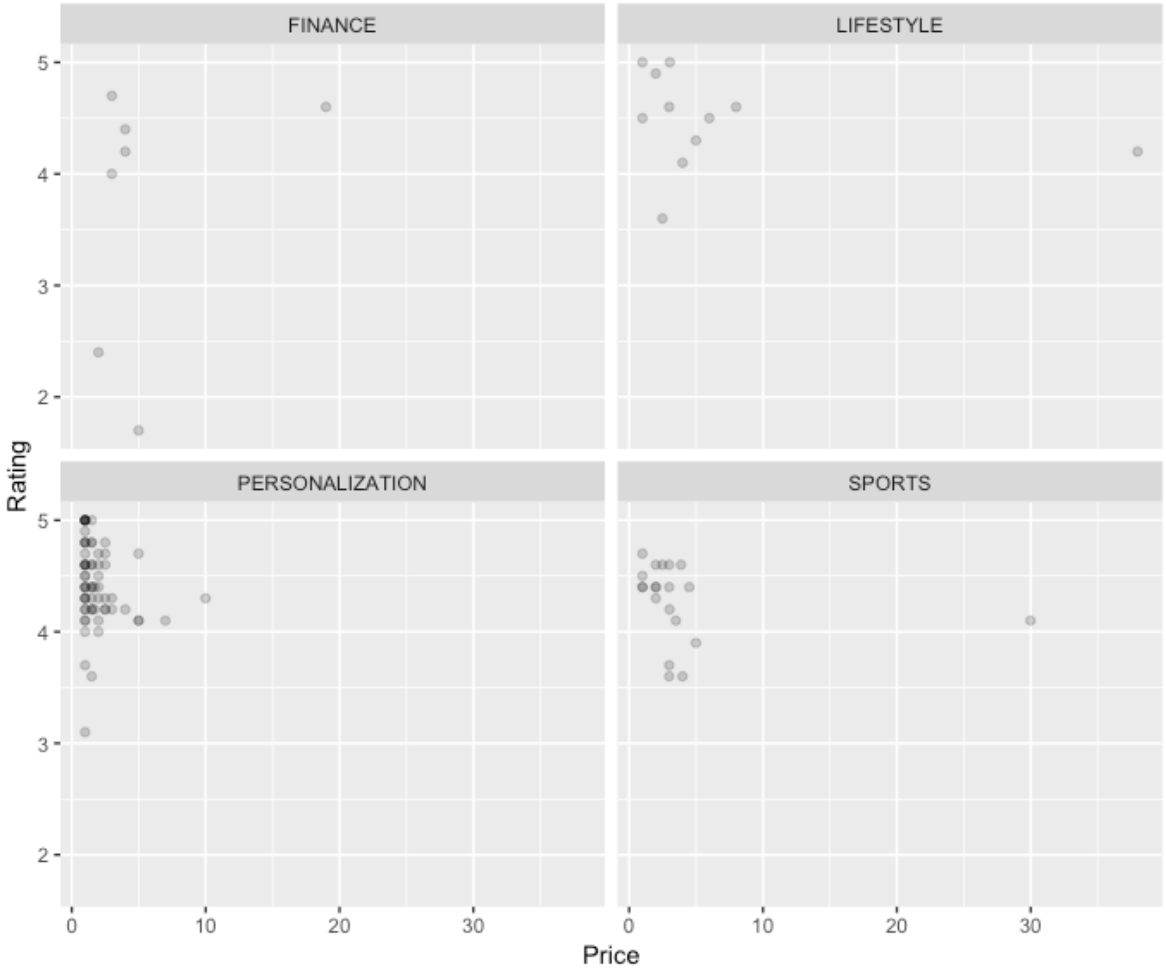
2 [그림 12-16]의 그래프는 어떤 의미일지 생각해보라.

그래프의 해석은 여러가지 관점에서 가능하나, 아래와 같은 해석이 무난할 것이다.
- 많은 수의 앱이 거의 $10 이하에 집중되어 있다.
- 평점 4.5 이상의 앱은 거의 $20 이하에 분포한다.
- 가격이 $20 이상인 앱은 평점이 오히려 감소한다 등.

3 앞에서 분석한 4개 카테고리를 제외한 다른 카테고리에서 평점과 가격의 분포 특성을 시각 화하라.

```
>x %>% group_by(Category) %>% summarize(n=n()) %>%
arrange(desc(n)) %>% head(10)
# A tibble: 10 x 2
   Category             n
   <fct>            <int>
 1 FAMILY            1617
 2 GAME               974
 3 TOOLS              634
 4 MEDICAL            324
 5 LIFESTYLE          280
 6 PERSONALIZATION    280
 7 FINANCE            266
 8 SPORTS             247
 9 BUSINESS           246
10 PHOTOGRAPHY        236
```

```
> x %>% filter(Type=="Paid"&Price<50&Category %in% c("LIFESTYLE",
"PERSONALIZATION", "FINANCE", "SPORTS")) %>% ggplot(aes(Price,
Rating))+geom_point(alpha=0.2)+facet_wrap(~Category)
```



04.

1 선형 회귀 모델 대신 일반화 선형 회귀 모델을 사용한 교차 성능 검증 결과를 산출하라.

설명 변수 조합	모델 성능(예측값의 mse)
Category	0.2872693
Size	0.2945681
Content.Rating	0.2956962
Category + Size	0.2865305
Category + Content.Rating	0.2862756
Size + Content.Rating	0.2936191
Category + Size + Content.Rating	0.2865483

선형 회귀 모델에 의한 교차 성능보다 좋은 결과를 얻을 수 있다.

2 Category + Size를 사용한 모델이 가장 성능이 좋았다. 그 이유는 무엇일까?

결정트리, 랜덤포리스트, SVM 을 이용한 교차검증결과에서 Category + Size 를 설명변수 조합으로 사용한 모델이 가장 성능이 좋았다. 특히 랜덤포리스트를 이용한 모델에서 가장 낮은 mse 를 보였다. 그림 12-6 의 Rating 과 Reviews 의 관계는 Reviews를 통해 (특히 높은 Reviews 의 경우에) Rating 을 일정범위에서 예측할 수 있음을 보여준다. 그러나 Reviews 는 출시 이전에 알 수 없으므로 이를 설명 변수로 사용할 수는 없음을 이미 설명한 바 있다. 그림 12-18 에 나타난 Rating 과 Size 의 관계가 그림 12-6과 유사할 뿐 아니라, 오히려 Size 에 대한 Rating의 분포가 한정되어 있어, 예측 성능은 더 좋을 수 있다. 그림 12-21 역시 Category 내에서 Rating 의 분포가 한정된 경향을 보이고 있어 예측 성능을 기대 할 수 있고, 이 두 변수의 조합된 경우 좀 더 좋은 성능이 나온 것으로 이해할 수 있다.