HW1-2 데이터프레임 연습

김서준

2023 11 12

1.NBA dataset

1.1

```
##출처: https://www.kaggle.com/datasets/bryanchungweather/nba-player-stats-dataset-for-the-2023-2024?rvi=1
nba_player <- read.csv("https://ajouackr-my.sharepoint.com/:x:/g/personal/rlatjwns234_ajou_ac_kr/EXoBO5ZRDKZHvGzUf7jsuNMBq9NaR3IC3pcFUHVGV3c5ZA?e=w9VOIL&download=1")
#각 컬럼에 대한 정보는 다음과 같다.
#Player(선수): 선수명, Pos(포지션): 포지션, Age(나이): 나이, Tm(팀): 소속된 팀의 약자, G(게임 수): 출전한 게임 수, GS(선발 출전 수): 선발로 출전한 게임 수, MP(평균 출장 시간): 평균 출장 시간(분 단위), FG(필드 골 성공 횟수): 필드 골 (Field Goals) 성공 횟수, FGA(필드 골 시도 횟수): 필드 골 시도 횟수, FG.(필드 골 성공률): 필드 골 성공률 (FG/FGA), X3P(3점슛 성공 횟수): 3점슛 성공 횟수, X3PA(3점슛 시도 횟수): 3점슛 시도 횟수, X3P.(3점슛 성공률): 3점슛 성공률 (3P/3PA), X2P(2점슛 성공 횟수): 2점슛 성공 횟수, X2PA(2점슛 시도 횟수): 2점슛 시도 횟수, X2P.(2점슛 성공률): 2점슛 성공률 (2P/2PA), eFG.(효과적인 필드 골 성공률): 효과적인 필드 골 성공률 ([(FG + 0.5 * 3P) / FGA]), FT(자유투 성공 횟수): 자유투 성공 횟수, FTA(자유투 시도 횟수): 자유투 시도 횟수, FT.(자유투 성공률): 자유투 성공률 (FT/FTA), ORB(공격 리바운드): 공격 리바운드 (Offensive Rebounds), DRB(수비 리바운드): 수비 리바운드 (Defensive Rebounds), TRB(총 리바운드): 총 리바운드 (Total Rebounds), AST(어시스트): 어시스트 (Assists), STL(스틸): 스틸 (Steals), BLK(블록): 블록 (Blocks), TOV(턴오버): 턴오버 (Turnovers), PF(개인 파울): 개인 파울 (Personal Fouls), PTS(득점): 득점 (Points)
```

1.2.

```
#이 중 Player(선수): 선수명, Pos(포지션): 포지션, Age(나이): 나이, Tm(팀): 소속된 팀의 약자, G(게임 수): 출전한 게임 수, GS(선발 출전 수), X3P.(3점슛 성공률): 3점슛 성공률, X2P.(2점슛 성공률): 2점슛 성공률, TRB(총 리바운드): 총 리바운드 (Total Rebounds), AST(어시스트): 어시스트 (Assists), STL(스틸): 스틸 (Steals), BLK(블록): 블록 (Blocks), TOV(턴오버): 턴오버 (Turnovers), PTS(득점): 득점 (Points) 만을 가지고 다시 데이터 프레임을 생성한다.
nba_player <- data.frame(
      "선수명" = nba_player$Player,
      "포지션" = nba_player$Pos,
      "나이" = nba_player$Age,
      "팀" = nba_player$Tm,
      "출전게임수" = nba_player$G,
      "선발출전게임수" = nba_player$GS,
      "3점슛성공률" = nba_player$X3P.,
      "2점슛성공률" = nba_player$X2P.,
      "리바운드" = nba_player$TRB,
      "어시스트" = nba_player$AST,
      "스틸" = nba_player$STL,
      "블록" = nba_player$BLK,
      "턴오버" = nba_player$TOV,
      "평균득점" = nba_player$PTS,
      stringsAsFactors = FALSE
    )
```

1.3.

```
str(nba_player)
```

```
## 'data.frame':    461 obs. of  14 variables:
##  $ 선수명        : chr  "Joel Embiid" "Devin Booker" "Luka Don?i?" "De'Aaron Fox" ...
##  $ 포지션        : chr  "C" "SG" "PG" "PG" ...
##  $ 나이          : int  29 27 24 26 27 35 35 28 25 25 ...
##  $ 팀            : chr  "PHI" "PHO" "DAL" "SAC" ...
##  $ 출전게임수    : int  7 2 8 3 7 9 8 9 7 7 ...
##  $ 선발출전게임수: int  7 2 8 3 7 9 8 9 7 7 ...
##  $ X3점슛성공률  : num  0.4 0.533 0.388 0.375 0.364 0.473 0.355 0.333 0.411 0.321 ...
##  $ X2점슛성공률  : num  0.556 0.6 0.573 0.543 0.621 0.593 0.551 0.707 0.625 0.556 ...
##  $ 리바운드      : num  10.9 7.5 9.3 4.3 4.9 4.8 6.6 12.9 9.7 7 ...
##  $ 어시스트      : num  5.9 10.5 8.8 6 5.6 4.2 4.6 8 3.6 6.4 ...
##  $ 스틸          : num  0.7 0.5 1.4 1.3 2.3 1 1 1.1 1.6 2.3 ...
##  $ 블록          : num  2.3 0 0.6 0.7 0.6 0.1 1.3 0.8 0.1 0.6 ...
##  $ 턴오버        : num  3.7 5.5 4.9 2.7 2.7 3.6 4.3 3.4 3 2.6 ...
##  $ 평균득점      : num  31.7 31.5 31.5 31.3 30.7 30 29.3 29.1 28.4 28.1 ...
```

```
#각 컬럼들의 자료형을 확인한 결과 형변환을 해줘야하는 값은 따로 없다고 판단.
#그러나 팀이나 포지션별로 무언가 점추정치를 얻으려 할때 평균득점이 0인 즉 팀에 등록만 되있고 뛰지 않은 선수들은 방해가 된다고 여기어 데이터 프레임의 행에서 삭제한다.
dim(nba_player)
```

```
## [1] 461  14
```

```
nba_player <- nba_player[nba_player$평균득점 != 0.0, ]
dim(nba_player)
```

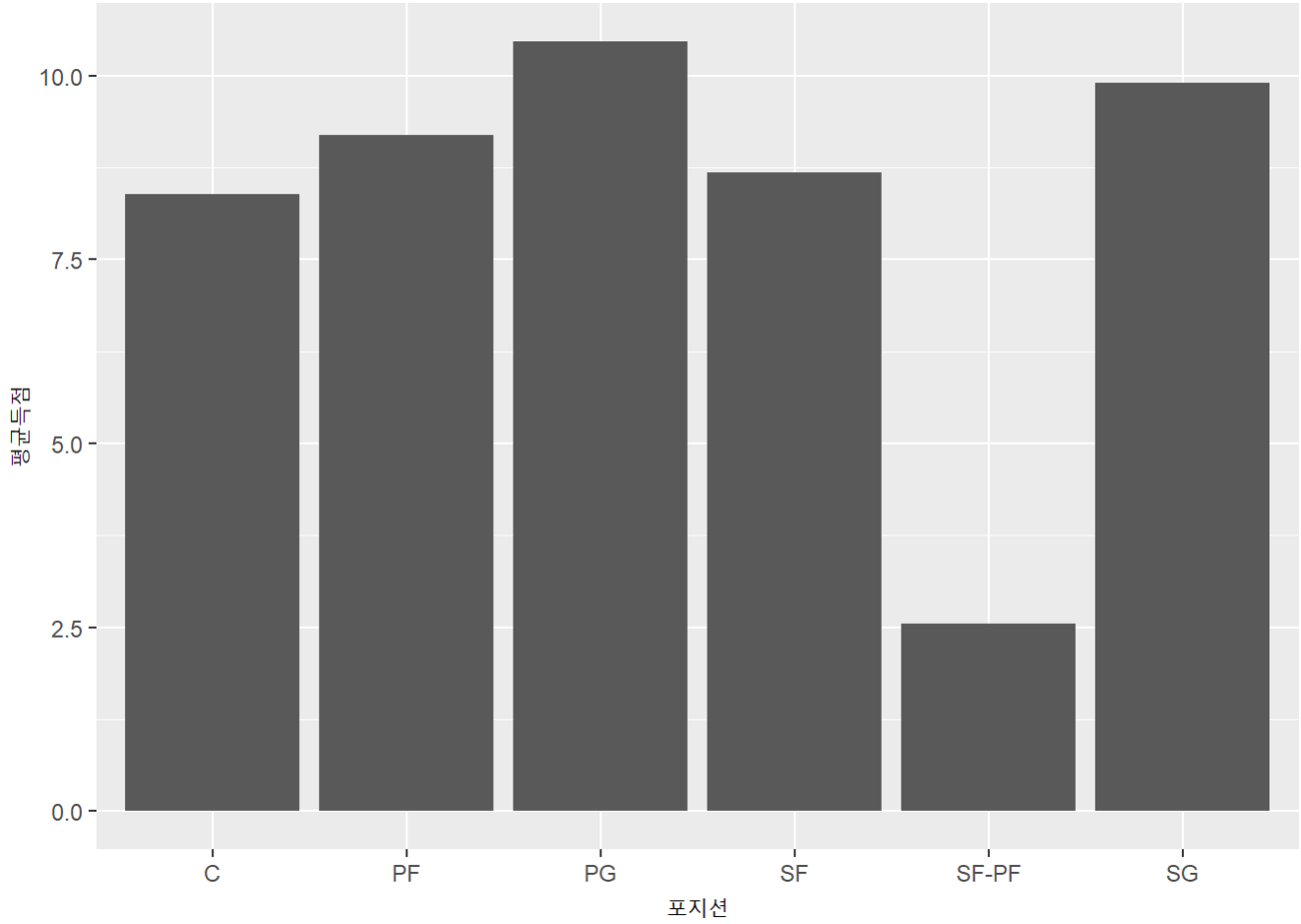
```
## [1] 427  14
```

```
#득점순위 10위까지를 score_top10변수에 저장
score_top10 <- nba_player[order(nba_player$평균득점, decreasing = TRUE), ][1:10, "선수명"]
score_top10 <- data.frame("득점순위" = score_top10, "평균득점" = nba_player[order(nba_player$평균득점, decreasing = TRUE), ][1:10,"평균득점"])
# 결과 출력
print(score_top10)
```

```
##                   득점순위 평균득점
## 1              Joel Embiid     31.7
## 2             Devin Booker     31.5
## 3              Luka Don?i?     31.5
## 4             De'Aaron Fox     31.3
## 5         Donovan Mitchell     30.7
## 6            Stephen Curry     30.0
## 7             Kevin Durant     29.3
## 8             Nikola Joki?     29.1
## 9             Jayson Tatum     28.4
## 10 Shai Gilgeous-Alexander     28.1
```

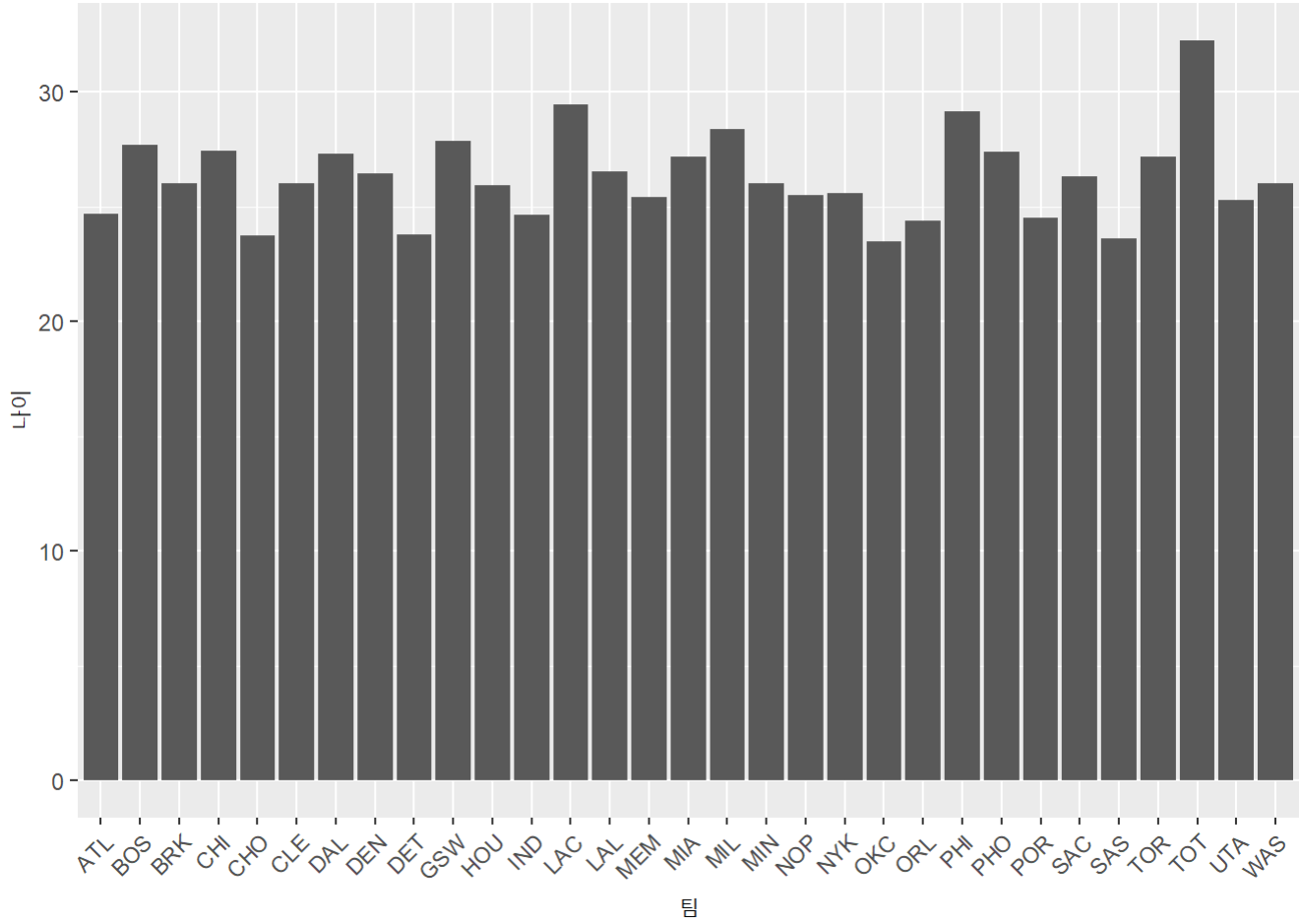
1.4. 포지션 별 평균 득점 시각화

```
#포지션별 평균득점을 aggregate 를 사용하여 avg_score_by_pos(포지션 별 평균 득점) 라는 변수에 담음
avg_score_by_position <- aggregate(평균득점 ~ 포지션, data = nba_player, FUN = mean)
library(ggplot2)#시각화 패키지 ggplot2사용
ggplot(avg_score_by_position, aes(x = 포지션, y = 평균득점))  + geom_bar(stat = 'identity') #데이터는 데이터프레임 nba_player를 사용하고 x축엔 포지션이 y축엔 평균 득점을 나타내는 그래프를 그린다. 텍스트 겹침을 피하기 위해 45도 돌리고 간격을 조절한다.
```



#1.5 팀별 평균 나이 시각화

```
#팀별 평균나이을 aggregate 를 사용하여 avg_age_by_team(팀별 평균나이) 라는 변수에 담음
avg_age_by_team <- aggregate(나이 ~ 팀, data = nba_player, FUN = mean)
library(ggplot2)#시각화 패키지 ggplot2사용
ggplot(data = avg_age_by_team, mapping = aes(x = 팀, y = 나이)) + geom_bar(stat = 'identity')+t
heme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#1.6

```
#가장 평균나이가 많은 팀과 적은 팀을 알아보자!
head(avg_age_by_team[order(avg_age_by_team$나이, decreasing = TRUE), ])
```

```
##     팀     나이
## 29 TOT 32.25000
## 13 LAC 29.43750
## 23 PHI 29.13333
## 17 MIL 28.37500
## 10 GSW 27.84615
## 2  BOS 27.69231
```

```
tail(avg_age_by_team[order(avg_age_by_team$나이, decreasing = TRUE), ])
```

```
##     팀     나이
## 25 POR 24.50000
## 22 ORL 24.37500
## 9  DET 23.76923
## 5  CHO 23.72727
## 27 SAS 23.60000
## 21 OKC 23.46667
```

#가장 평균나이가 많은 팀은 토론토 랩터스로 평균나이는 32.25세이며 가장 평균나이가 적은 팀은 오클라호마 시티 썬더스로 23.46세이다. 두 팀간 약 9세의 평균나이차가 있음을 알 수 있다.

1.7.

```
# 주전 여부를 나타내는 새로운 컬럼 추가
#"선발출전게임수" 컬럼의 데이터가 6이상이면 주전
# 3 이상 6미만이면 반주전
# 3미만이라면  후보라는 값을 가지는 새로운 컬럼을 추가

nba_player$출전상태 <- ifelse(nba_player$선발출전게임수 >= 6, "주전",ifelse(nba_player$선발출전
게임수 >= 3, "반주전", "후보"))
head(nba_player)
```

```
##             선수명 포지션 나이  팀 출전게임수 선발출전게임수 X3점슛성공률
## 1      Joel Embiid      C   29 PHI          7             7        0.400
## 2     Devin Booker     SG   27 PHO          2             2        0.533
## 3      Luka Don?i?     PG   24 DAL          8             8        0.388
## 4     De'Aaron Fox     PG   26 SAC          3             3        0.375
## 5 Donovan Mitchell     SG   27 CLE          7             7        0.364
## 6    Stephen Curry     PG   35 GSW          9             9        0.473
##   X2점슛성공률 리바운드 어시스트 스틸 블록 턴오버 평균득점 출전상태
## 1        0.556     10.9      5.9  0.7  2.3    3.7     31.7     주전
## 2        0.600      7.5     10.5  0.5  0.0    5.5     31.5     후보
## 3        0.573      9.3      8.8  1.4  0.6    4.9     31.5     주전
## 4        0.543      4.3      6.0  1.3  0.7    2.7     31.3   반주전
## 5        0.621      4.9      5.6  2.3  0.6    2.7     30.7     주전
## 6        0.593      4.8      4.2  1.0  0.1    3.6     30.0     주전
```

```
nba_player$출전상태 <- ifelse(nba_player$선발출전게임수 >= 6, "주전",ifelse(nba_player$선발출전
게임수 >= 3, "반주전", "후보"))
```

1.8. 포지션별

```
# 포지션별 스탯을 알아보기 위해 aggregate 함수와 cbind를 이용하였다.
stats_by_pos <- aggregate(cbind(X3점슛성공률, X2점슛성공률, 리바운드, 어시스트, 스틸, 블록) ~
포지션, data = nba_player, FUN = mean)
stats_by_pos
```

```
##   포지션 X3점슛성공률 X2점슛성공률 리바운드 어시스트      스틸      블록
## 1      C    0.2813488    0.6198372 6.279070 2.088372 0.5790698 0.8186047
## 2     PF    0.3325342    0.5813151 4.728767 1.879452 0.6794521 0.5657534
## 3     PG    0.3503000    0.5024000 2.771429 3.794286 0.8485714 0.2657143
## 4     SF    0.3180698    0.5464070 3.383721 1.545349 0.6406977 0.3825581
## 5  SF-PF    0.2680000    0.5415000 3.200000 0.750000 1.3500000 0.5500000
## 6     SG    0.3454900    0.5308200 2.701000 2.351000 0.7210000 0.3060000
```

1.9. 팀을 동부, 서부로 분리하기

```
#NBA는 동부, 서부로 Division이 분리되어있다.
#우선 팀이 무었무었이 있는지 알아보자 unique()함수 사용
unique(nba_player$팀)
```

```
##  [1] "PHI" "PHO" "DAL" "SAC" "CLE" "GSW" "DEN" "BOS" "OKC" "MIN" "BRK" "LAC"
## [13] "MEM" "WAS" "MIL" "LAL" "DET" "UTA" "IND" "MIA" "POR" "ATL" "NYK" "CHI"
## [25] "CHO" "NOP" "TOR" "HOU" "SAS" "ORL" "TOT"
```

```
length(unique(nba_player$팀))
```

```
## [1] 31
```

```
# 동부 지역에 속하는 팀
eastern_teams <- c("BOS", "BKN", "NYK", "PHI", "TOR", "CHI", "CLE", "DET", "IND", "MIL", "ATL",
"CHA", "MIA", "ORL", "WAS")

# 서부 지역에 속하는 팀
western_teams <- c("DAL", "DEN", "GSW", "HOU", "LAC", "LAL", "MEM", "MIN", "NOP", "OKC", "PHO",
"POR", "SAC", "SAS", "UTA")

# 팀별 지역 정보를 나타내는 새로운 컬럼 추가
nba_player$디비전 <- ifelse(nba_player$팀 %in% eastern_teams, "동부", "서부")

head(nba_player)
```

```
##             선수명 포지션 나이  팀 출전게임수 선발출전게임수 X3점슛성공률
## 1      Joel Embiid     C   29 PHI          7              7        0.400
## 2     Devin Booker    SG   27 PHO          2              2        0.533
## 3      Luka Don?i?    PG   24 DAL          8              8        0.388
## 4     De'Aaron Fox    PG   26 SAC          3              3        0.375
## 5 Donovan Mitchell    SG   27 CLE          7              7        0.364
## 6    Stephen Curry    PG   35 GSW          9              9        0.473
##   X2점슛성공률 리바운드 어시스트 스틸 블록 턴오버 평균득점 출전상태 디비전
## 1        0.556     10.9      5.9  0.7  2.3    3.7     31.7     주전   동부
## 2        0.600      7.5     10.5  0.5  0.0    5.5     31.5     후보   서부
## 3        0.573      9.3      8.8  1.4  0.6    4.9     31.5     주전   서부
## 4        0.543      4.3      6.0  1.3  0.7    2.7     31.3   반주전   서부
## 5        0.621      4.9      5.6  2.3  0.6    2.7     30.7     주전   동부
## 6        0.593      4.8      4.2  1.0  0.1    3.6     30.0     주전   서부
```

2 laptop dataset

2.1.

```
#read.csv를 통해 onedrive에 있는 데이터셋을 불러온다.
#출처: https://www.kaggle.com/datasets/muhammetvarl/laptop-price?rvi=1
laptop <- read.csv("https://ajouackr-my.sharepoint.com/:x:/g/personal/rlatjwns234_ajou_ac_kr/Ea
W-BieN9RNJmTrKOh9eh7lBlZED2lCR8AuvMH-1y7lS_Q?e=yVshK9&download=1")
```

2.2.

```
str(laptop)
```

```
## 'data.frame':    1303 obs. of  13 variables:
##  $ laptop_ID       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Company         : chr  "Apple" "Apple" "HP" "Apple" ...
##  $ Product         : chr  "MacBook Pro" "Macbook Air" "250 G6" "MacBook Pro" ...
##  $ TypeName        : chr  "Ultrabook" "Ultrabook" "Notebook" "Ultrabook" ...
##  $ Inches          : num  13.3 13.3 15.6 15.4 13.3 15.6 15.4 13.3 14 14 ...
##  $ ScreenResolution: chr  "IPS Panel Retina Display 2560x1600" "1440x900" "Full HD 1920x108
0" "IPS Panel Retina Display 2880x1800" ...
##  $ Cpu             : chr  "Intel Core i5 2.3GHz" "Intel Core i5 1.8GHz" "Intel Core i5 7200U
2.5GHz" "Intel Core i7 2.7GHz" ...
##  $ Ram             : chr  "8GB" "8GB" "8GB" "16GB" ...
##  $ Memory          : chr  "128GB SSD" "128GB Flash Storage" "256GB SSD" "512GB SSD" ...
##  $ Gpu             : chr  "Intel Iris Plus Graphics 640" "Intel HD Graphics 6000" "Intel HD
Graphics 620" "AMD Radeon Pro 455" ...
##  $ OpSys           : chr  "macOS" "macOS" "No OS" "macOS" ...
##  $ Weight          : chr  "1.37kg" "1.34kg" "1.86kg" "1.83kg" ...
##  $ Price_euros     : num  1340 899 575 2537 1804 ...
```

```
#str()을 통해 데이터프레임안에 데이터들의 자료형을 살펴본 결과
#Ram컬럼과 Weight컬럼이 각각 int형과 num형으로 바뀔 필요가 있는것 같다고 판단된다.
laptop <- data.frame(
  "제조사" = laptop$Company,
  "제품명" = laptop$Product,
  "화면크기" = laptop$Inches,
  "Cpu" = laptop$Cpu,
  "램" = laptop$Ram,
  "메모리" = laptop$Memory,
  "운영체제" = laptop$OpSys,
  "무게" = laptop$Weight,
  "가격_유로" = laptop$Price_euros,
  stringsAsFactors = FALSE
)
```

##2.3. 특정 컬럼에서 특정 문자 제거

```
#자료형을 변환시켜주기위해 먼저 Ram컬럼과 Weight컬럼에서 GB와 kg를 삭제한다.
laptop$램 <- gsub("GB", "", laptop$램)
laptop$무게 <- gsub("kg", "", laptop$무게)
#as.자료형을 사용해서 바꾸어보자
laptop$램 <- as.integer(laptop$램)
laptop$무게 <- as.numeric(laptop$무게)

str(laptop)
```

```
## 'data.frame':    1303 obs. of  9 variables:
##  $ 제조사   : chr  "Apple" "Apple" "HP" "Apple" ...
##  $ 제품명   : chr  "MacBook Pro" "Macbook Air" "250 G6" "MacBook Pro" ...
##  $ 화면크기 : num  13.3 13.3 15.6 15.4 13.3 15.6 15.4 13.3 14 14 ...
##  $ Cpu      : chr  "Intel Core i5 2.3GHz" "Intel Core i5 1.8GHz" "Intel Core i5 7200U 2.5GH
z" "Intel Core i7 2.7GHz" ...
##  $ 램       : int  8 8 8 16 8 4 16 8 16 8 ...
##  $ 메모리   : chr  "128GB SSD" "128GB Flash Storage" "256GB SSD" "512GB SSD" ...
##  $ 운영체제 : chr  "macOS" "macOS" "No OS" "macOS" ...
##  $ 무게     : num  1.37 1.34 1.86 1.83 1.37 2.1 2.04 1.34 1.3 1.6 ...
##  $ 가격_유로: num  1340 899 575 2537 1804 ...
```

##2.4. 유로를 원화로 환전하여 컬럼 추가하기.

```
#2023 11 12 기준 1유로화는 1414원이다.
#laptop$가격_유로 컬럼에 있는 데이터값들에 1414원을 곱해서 원화가격을 구하고 이를 새로운 컬럼
#가격_원화에 추가한다.
laptop$가격_원화 <- laptop$가격_유로 * 1414
head(laptop$가격_원화)
```

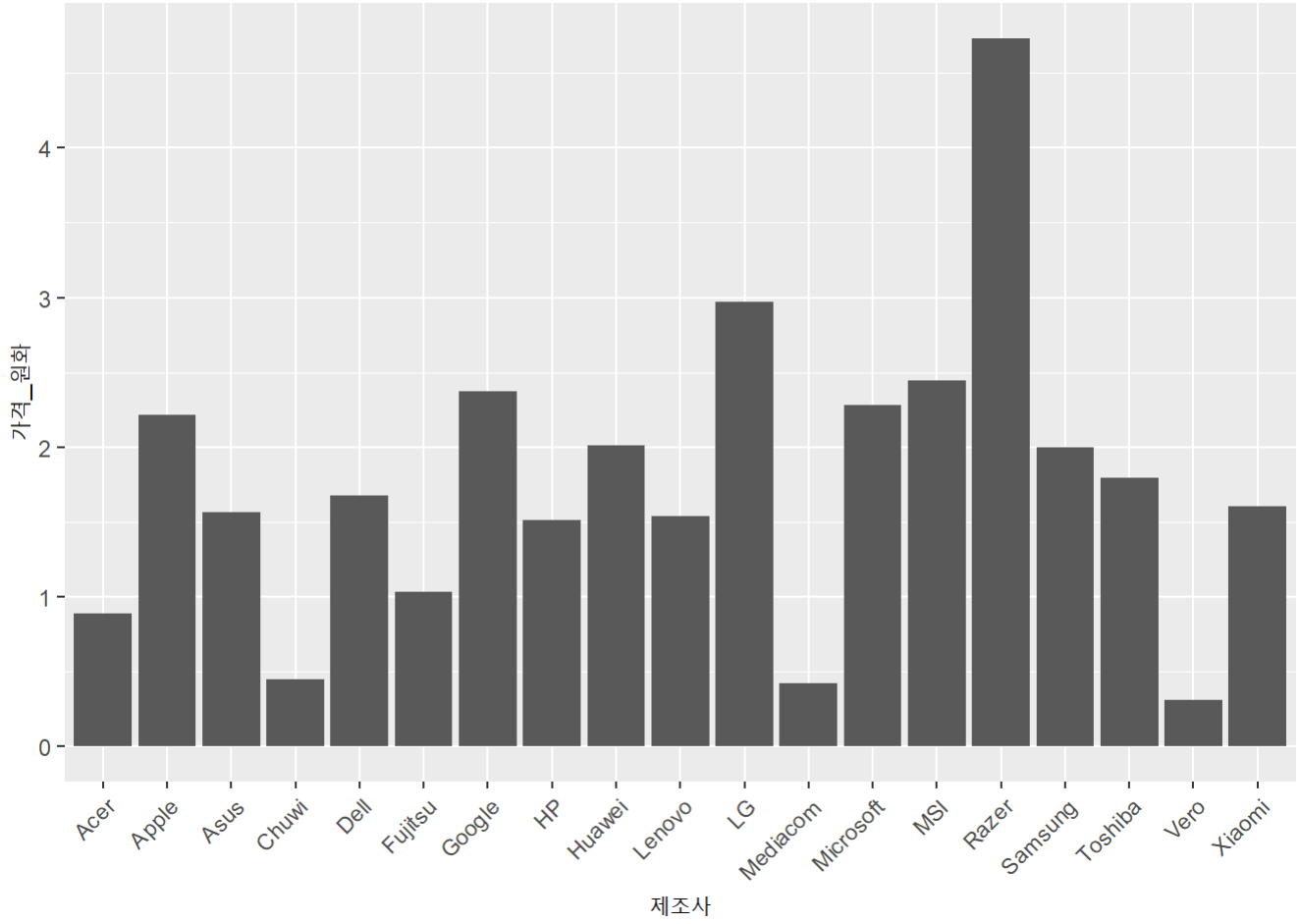
```
## [1] 1894322 1271101  813050 3587954 2550290  565600
```

##2.4. 제조사별 평균가격 알아보기, 시각화

```
#제조사별 평균 가격을 알아보기 위해 먼저 제조사의 갯수를 알아보자
#length()와 unique()를 사용한다.
length(unique(laptop$제조사))
```

```
## [1] 19
```

```
#제조사별 평균 가격을 aggregate 를 사용하여 avg_price_by_com(제조사별 평균가격) 라는 변수에 담
음
avg_price_by_com <- aggregate(가격_원화 ~ 제조사, data = laptop, FUN = mean)
library(ggplot2)#시각화 패키지 ggplot2사용
#y축의 단위는 백만원으로 한다.
ggplot(data = avg_price_by_com, mapping = aes(x = 제조사, y = 가격_원화)) + scale_y_continuous
(labels = scales::comma_format(scale = 1e-6)) + geom_bar(stat = 'identity')+theme(axis.text.x =
element_text(angle = 45, hjust = 1))
```



3. 부산시 주차장 데이터 셋

3.1.

```
#출처 : https://www.data.go.kr/tcs/dss/selectApiDataDetailView.do?publicDataPk=15070822
# json파일을 로드하기 위해 jsonlite 라이브러리 사용
library(jsonlite)
serviceKey <- "qzyPoUonu9Kh1QHn8bcEq99acdp7FQTJeKtS5NC%2BNGbPLflMbSlzKyAd9WOqLoypM%2F36S7wDOrnf
cmeiPYPmiA%3D%3D"
numofRows <- "255" #이 파일은 데이터 행의 수가 255개임
pageNo <- "1"
resultType <- 'json'

url_json <- paste0('http://apis.data.go.kr/3330000/HeaundaeParkingInfoService/getParkingLotLis
t?serviceKey=',serviceKey,
                   '&numOfRows=',numofRows,
                   '&pageNo=',pageNo,
                   '&resultType=', resultType)

result <- fromJSON(url_json)
parking_Busan <- result$getParkingLotList$item
str(parking_Busan)
```

```
## 'data.frame':    193 obs. of  9 variables:
##  $ parkingLotName: chr  "반여2동 제1공영" "우동2로30번길제1호(한적길)" "대천공원 공영" "반여2동 제2공영(민방위교육장)" ...
##  $ address       : chr  "재반로211번길 62" "우동2로30번길 30 근처(우1동 228-14 경덕사옆)" "좌3동 1394" "재반로184번길 31" ...
##  $ grade         : chr  "4" "" "3" "4" ...
##  $ pNum          : chr  "266" "7" "79" "234" ...
##  $ charge        : chr  "민간위탁" "우1동 주민자치위원회" "민간위탁" "민간위탁" ...
##  $ year          : chr  "2000.11.01" "" "2018-09-06" "2004-11-01" ...
##  $ lat           : chr  "35.195824" "35.168006" "35.178415" "35.194326" ...
##  $ lng           : chr  "129.127424" "129.161112" "129.168655" "129.132311" ...
##  $ clsName       : chr  "공영주차장" "소규모공동주차장" "시설관리공단 공영주차장" "공영주차장" ...
```

3.2. 자료형 바꾸기

```
#위도와 경도의 자료형을 numeric형 데이터로 바꾼다.
parking_Busan$lat <- as.numeric(parking_Busan$lat)
parking_Busan$lng <- as.numeric(parking_Busan$lng)
print(class(parking_Busan$lat))
```

```
## [1] "numeric"
```

```
print(class(parking_Busan$lng))
```

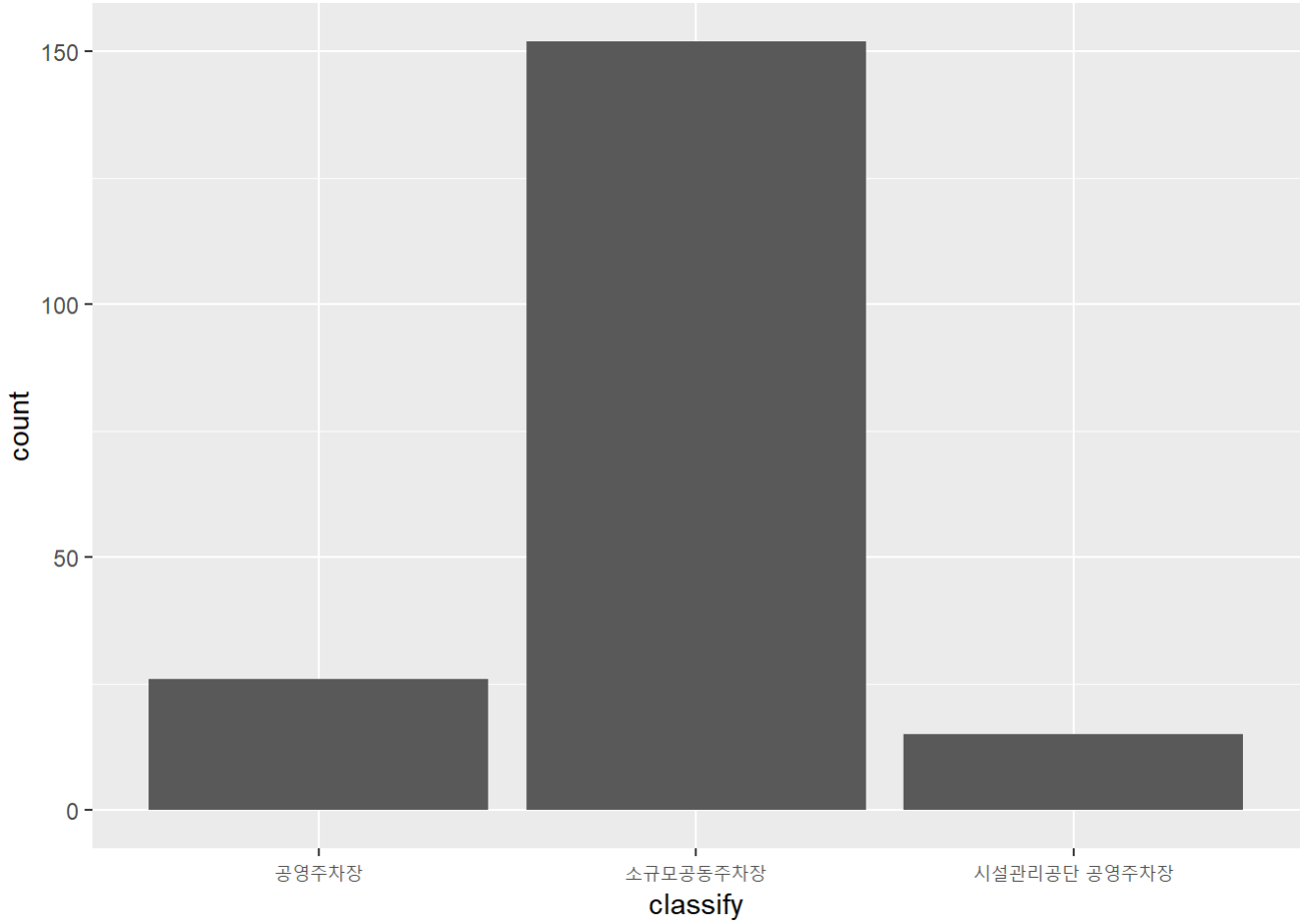
```
## [1] "numeric"
```

3.3. 분류명 별로 몇개씩의 주차장이 있는지 시각화 하기

```
length(unique(parking_Busan$clsName)) # 데이터의 종류 갯수 확인
```

```
## [1] 3
```

```
classify <- parking_Busan$clsName
library(ggplot2) #그래프 시각화 패키지를 불러온다.
ggplot(data = parking_Busan, mapping = aes(x = classify)) + geom_bar()+theme()
```


150
100
50
0
count
공영주차장
소규모공동주차장
시설관리공단 공영주차장
classify