1. 비교(Compare)의 시각화

비교의 시각화는특정 변수의 변량에 따른 데이터 값들간의 순서를 비교하는 시각화이다. 비교의 시각화는 다른 시각화와 다른 두가지 특징이 있다.

첫번째 특징은 시각화 그래프 내에서 한번 더 통계처리가 필요하다는 특성을 가진다. 보통 비교의 시각화는 데이터에 값에 따라 시각화 한 후 그 값들을 다시 정렬함으로써 시각화를 완성하게 된다. 따라서 데이터 시각화 후에 ggplot객체 내에서 정렬과 같은 통계 처리가 한번 더 일어나게 된다.

두번째 특징은 그 관심의 대상이 데이터 자체의 값 보다는 비교되는 대상 내에서의 상대적 위치에 더 쏠려있다는 점이다. 비교되는 대상 중에 가장 값이 크거나 작은 변량이 무엇인지에 관심이 있는 시각화이다. 비교되는 대상들의 상대적 위치를 명확하게 구분되어야 하기 때문에 비교되는 대상들이 명확하게 구분되어야 하고 이 대상들을 명학하게 구분하기 위해 비교를 위한 변수는 이산형 변수를 사용하는 것이 일반적이다.

비교의 시각화는 막대 그래프, 순위 막대 그래프, 롤리팝 그래프, 도트 그래프 등이 있다.

# 막대 그래프를 그리는 세가지 방법 - 막대 그래프

막대 그래프는 이산형 변수로 구분되는 연속형 수치형 변수를 시각화하기 위해 사용한다. 각각의 이산형 변수에 하나씩의 막대가 표시되고 막대의 길이를 사용하여 데이터 값을 표현한다. 막대 그래프는 데이터를 쉽게 읽을 수 있을 뿐아니라 데이터간의 비교가 상대적으로 쉽다. 따라서 데이터간의 비교에 매우 효과적인 시각화 방법이다.

막대그래프의 가장 일반적인 형태는 세로 막대가 가로축으로 배열되어 있는 형태이다. 과거의 한정된 지면에 표현하던 때나 데이터가 한정적이던 때는 비교할 이산형 변수의 변량이 많지 았았지만 지금같이 데이터가 많이지고 웹사이트와 같이 위아래로 스크롤이 가능한 화면에서는 오히려 가로형 막대를 세로축에 표현하는 경우도 많다.

막대 그래프를 실습해보기 전에 실습에 필요한 데이터를 다음과 같이 데이터를 준비하도록 하겠다. 다음의 데이터는 df\_취업통계 데이터를 과정구분, 대계열, 중계열 별로 그룹화하여 각각의 열의 합계값을 만들고 각각의 중계열별 취업률을 산출한다.

df\_취업통계\_계열별 <- df\_취업통계 |>   
 group\_by(과정구분, 대계열, 중계열) |>   
 summarise(졸업자 = sum(졸업자\_계),   
 취업자 = sum(취업자\_합계\_계),   
 교외취업자 = sum(취업자\_교외취업자\_계),   
 교내취업자 = sum(취업자\_교내취업자\_계),   
 해외취업자 = sum(취업자\_해외취업자\_계),   
 농림어업종사자 = sum(취업자\_농림어업종사자\_계),   
 개인창작활동종사자 = sum(취업자\_개인창작활동종사자\_계),   
 일인창사업자 = sum(`취업자\_1인창(사)업자\_계`),   
 프리랜서 = sum(취업자\_프리랜서\_계),   
 진학자 = sum(진학자\_계),   
 입대자 = sum(입대자),  
 취업불가능자 = sum(취업불가능자\_계),   
 외국인유학생 = sum(외국인유학생\_계),   
 제외인정자 = sum(제외인정자\_계),   
 기타 = sum(기타\_계),   
 미상 = sum(미상\_계),   
 ## 백분률인 취업률은 그 자체로 합계나 평균을 낼 수 없으니 각 그룹별로 재계산  
 취업률 = 취업자 / (졸업자 - (진학자+입대자+취업불가능자+외국인유학생+제외인정자))) |>  
 ## 계열의 표시 순서를 설정하기 위해 레벨을 재조정  
 mutate(대계열 = fct\_relevel(대계열, '인문계열', '사회계열', '교육계열', '자연계열', '공학계열', '의약계열', '예체능계열'))  
  
df\_취업통계\_계열별 |> head()

## # A tibble: 6 x 20  
## # Groups: 과정구분, 대계열 [1]  
## 과정구분 대계열 중계열 졸업자 취업자 교외취업자 교내취업자 해외취업자  
## <chr> <fct> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 대학과정 공학계열 건축 7697 4943 4634 64 6  
## 2 대학과정 공학계열 교통ㆍ운송 4206 2443 2311 37 6  
## 3 대학과정 공학계열 기계ㆍ금속 11422 6206 5759 142 46  
## 4 대학과정 공학계열 기타 3962 2328 2141 65 17  
## 5 대학과정 공학계열 산업 3594 2249 2105 46 8  
## 6 대학과정 공학계열 소재ㆍ재료 6143 2948 2682 85 14  
## # ... with 12 more variables: 농림어업종사자 <dbl>, 개인창작활동종사자 <dbl>,  
## # 일인창사업자 <dbl>, 프리랜서 <dbl>, 진학자 <dbl>, 입대자 <dbl>,  
## # 취업불가능자 <dbl>, 외국인유학생 <dbl>, 제외인정자 <dbl>, 기타 <dbl>,  
## # 미상 <dbl>, 취업률 <dbl>

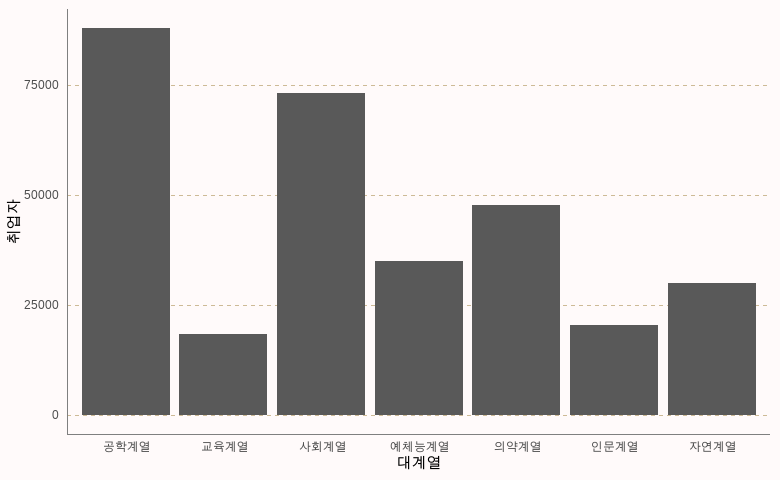
## stack, dodge, fill 타입의 막대 그래프

우리가 가장 흔하게 생각하는 막대 그래프는 각 변량별로 하나의 막대로 데이터를 표현하는 막대그래프이다. 하지만 그 막대를 구성하는 데이터들은 다시 세분화될 수 있다. 이렇게 하나의 막대로 표현된 세부 데이터를 다시 표현해야할 경우를 대비하여 ggplot2는 세가지 형태의 막대 그래프를 지원한다.

* stack : 막대의 세부 그룹을 각각 쌓아 올려 하나의 단일 막대를 생성
* dodge : 막대의 세부 그룹을 각각 옆으로 붙여 세부 그룹 수 만큼의 막대 생성
* fill : 막대의 세부 그룹을 각각 쌓아 올려 하나의 단일 막대를 만들되 전체 막대크 크기를 모두 동일하게 맞춰 세부 그룹들의 비율을 표현

다음의 기본 막대 그래프를에서 파생되는 세가지 형태의 막대 그래프의 형태를 비교해 보도록 한다.

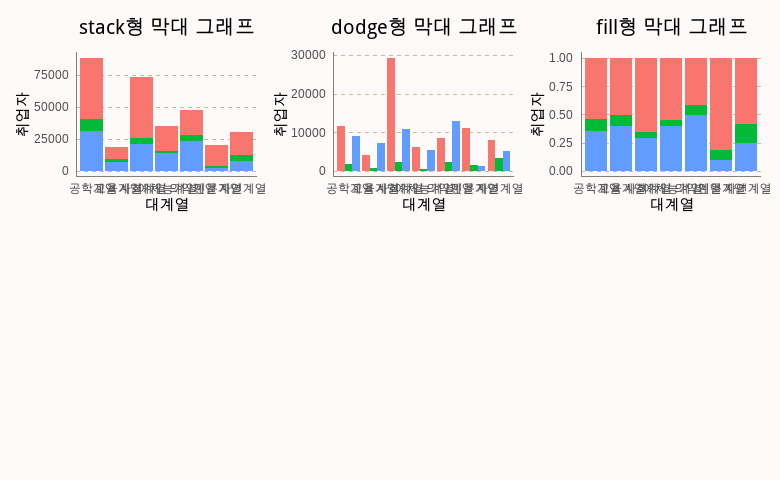
p\_bar\_type <- df\_취업통계\_계열별 |>  
 ## ggplot 객체 생성  
 ggplot()  
  
p\_bar\_type +  
 ## X축을 대계열, Y축을 취업자열로 매핑한 geom\_col 레이어 생성  
 ## geom\_bar를 사용하려면 geom\_bar(aes(x = 대계열, y = 취업자), stat = 'identity')  
 geom\_col(aes(x = 대계열, y = 취업자))



실행결과 8-기본형 막대 그래프

geom\_col()과 geom\_bar()의 막대 타입은 막대의 위치를 뜻하는 position 매개변수로 변경할 수 있다. 기본값은 ’stack’으로 전체 막대 길이는 전체 데이터 값이며 각각의 세부 그룹들이 위에 쌓아올려진 형태의 막대 그래프이다.

p\_bar\_type +  
 ## X축을 대계열, Y축을 취업자, fill을 과정구분으로 매핑한 stack형태 geom\_col 레이어 생성  
 geom\_col(aes(x = 대계열, y = 취업자, fill = 과정구분), position = 'stack', show.legend = FALSE) +   
 labs(title = 'stack형 막대 그래프')  
  
p\_bar\_type +  
 ## X축을 대계열, Y축을 취업자, fill을 과정구분으로 매핑한 dodge형태 geom\_col 레이어 생성  
 geom\_col(aes(x = 대계열, y = 취업자, fill = 과정구분), position = 'dodge', show.legend = FALSE) +   
 labs(title = 'dodge형 막대 그래프')  
  
p\_bar\_type +  
 ## X축을 대계열, Y축을 취업자, fill을 과정구분으로 매핑한 fill형태 geom\_col 레이어 생성  
 geom\_col(aes(x = 대계열, y = 취업자, fill = 과정구분), position = 'fill', show.legend = FALSE) +   
 labs(title = 'fill형 막대 그래프')



실행결과 8-막대그래프의 position 비교

# 막대만 있으면 안돼. 숫자도 있어야 해 – 막대 그래프의 변형

앞서 생성한 막대 그래프들에는 각각의 데이터 값이 표현되지 않아 데이터들을 비교하기가 어렵다. 각각의 막대 그래프의 타입에 따라 데이터 값을 표현하는 방법도 다 다르다.[[1]](#footnote-22)

## stack 막대 그래프의 데이터 값 넣기

stack 막대 그래프에 데이터 값을 넣으려면 가장 효과적인 것이 각각의 세부 막대 중간에 해당 데이터 값을 표현하고 전체 데이터 값을 맨위에 표현해 주는 방법이다. 이를 위해서는 두번의 작업이 필요하다.

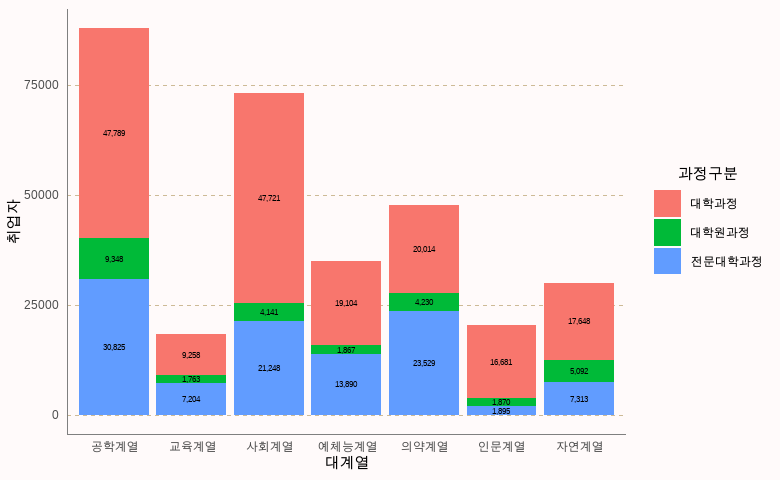
우선 막대를 구성하는 그룹별 합계값을 구해야 한다. 앞의 막대들은 대계열 막대를 과정구분으로 나누었기 때문에 대계열별 과정구분별 데이터 값(취업자수)가 필요하다. 이후 막대 그래프를 그리고 이 안에 geom\_text()로 데이터 값을 표현한다.

각 그룹의 중간에 데이터를 표현하기 위해서는 position\_stack()을 사용한다. 이 함수는 각각의 그룹 위에 기하요소를 올려서 표현해 주는 함수이다.

position\_stack(vjust = 1, reverse = FALSE)  
- vjust : 수직 정렬 위치 설정  
- reverse : 순서의 역순을 설정하는 논리값

이 함수는 geom\_col(), geom\_bar()이외의 기하요소에서도 사용할 수 있기 때문에 데이터 값을 표현하는데 사용하는 geom\_text()와 geom\_label()에 사용하면 해당 그룹에 데이터 값을 표현할 수 있다. 또 각 막대의 중간에 표현해야 하니 수평정렬 매개변수인 vjust를 0.5로 설정하면 중간에 위치하는 데이터 값을 표현할 수 있다.

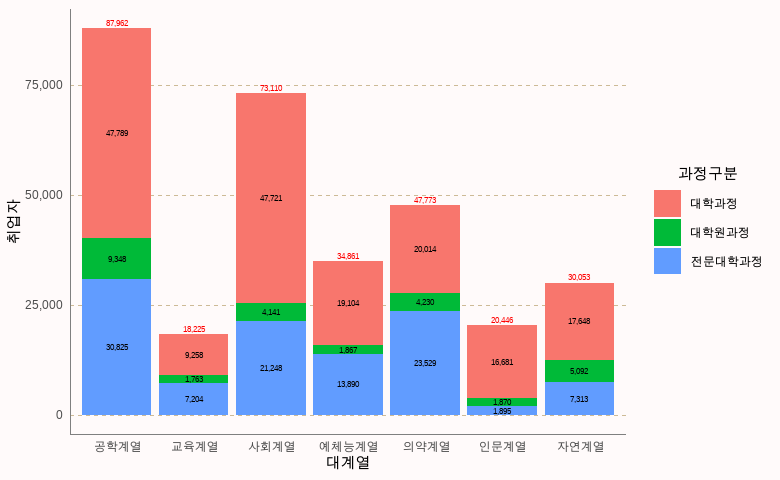
## 데이터 값을 표현하기 위해 대계열별 과정구분으로 그룹화하고  
p\_bar\_data\_stack <- df\_취업통계\_계열별 |> group\_by(대계열, 과정구분) |>  
 ## 합계값을 산출  
 summarise(취업자 = sum(취업자)) |>  
 ggplot() +  
 ## X축이 대계열, Y축이 취업자, fill이 과정구분으로 매핑되고 position이 stack으로 설정된 geom\_col 레이어 생성  
 geom\_col(aes(x = 대계열, y = 취업자, fill = 과정구분), position = 'stack') +  
 ## X축이 대계열, Y축이 취업자, lable이 comma형 취업자, fill은 과정구분으로 매핑되고 position은 position\_stack()으로 vjust가 0.5인 stack형으로 설정한 geom\_text 레이어 추가  
 geom\_text(aes(x = 대계열, y = 취업자, label = scales::comma(취업자, accuracy = 1), fill =과정구분), position = position\_stack(vjust = 0.5), size = 2)  
  
p\_bar\_data\_stack



실행결과 8-stack형 막대그래프에 데이터 값 표현

다음으로 전체 데이터 값을 표현해보자. 앞서 세부 그룹의 합계값을 구했기 때문에 이들의 합계값을 산출해주면 된다. 이를 위해 fun을 sum으로 설정한 stat\_summary()를 사용한다.

p\_bar\_data\_stack1 <- p\_bar\_data\_stack +  
 ## X축이 대계열, Y축이 취업자, lable이 comma형 ..y..(fun으로 설정한 함수의 결과값)으로 매핑되고 적용할 함수(fun)은 sum, 표현할 기하요소는 text, vjust는 0.5, size는 2인 stat\_summary 레이어 생성   
 stat\_summary(aes(x = 대계열, y = 취업자, label = scales::comma(..y.., accuracy = 1)), fun = 'sum', geom = 'text', color = 'red', vjust = -0.5, size = 2) +   
 scale\_y\_continuous(labels = scales::comma)  
  
p\_bar\_data\_stack1



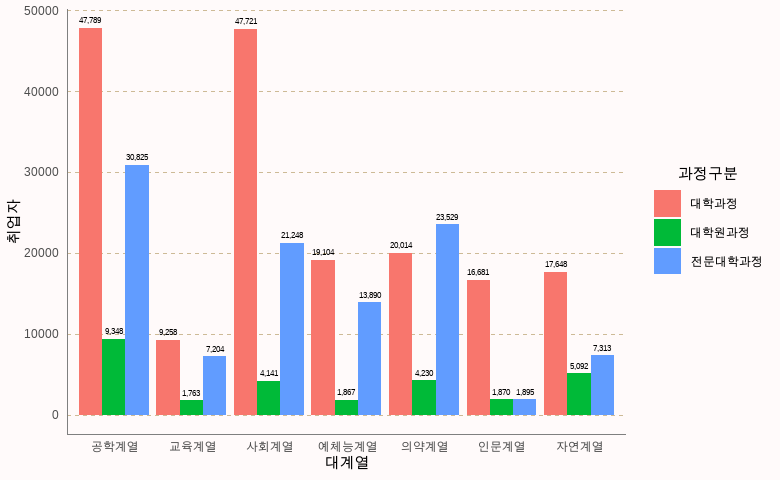
실행결과 8-stat\_\*()를 사용한 stack형 막대그래프에 데이터 값 표현

## dodge 막대 그래프의 데이터 값 넣기

dodge 막대 그래프는 stack 막대 그래프보다는 데이터 값을 넣기가 쉽다. 앞서 stack 막대 그래프에 geom\_text()의 위치를 설정하는데에는 postion\_stack()을 사용했다면 dodge 막대 그래프에는 position\_dodge()를 사용한다.

position\_dodge(width = NULL, preserve = c(“total”, “single”))  
- width : 옆으로 배열되는 너비 설정, 기본적인 막대의 너비는 0.9임  
- preserve : 막대 너비를 전체에 맞출지 각각의 개벌 막대에 맞출지 설정

## 데이터 값을 표현하기 위해 대계열별 과정구분으로 그룹화하고  
p\_bar\_data\_dodge <- df\_취업통계\_계열별 |> group\_by(대계열, 과정구분) |>  
 ## 합계값을 산출  
 summarise(취업자 = sum(취업자)) |>  
 ggplot() +  
 ## X축이 대계열, Y축이 취업자, fill이 과정구분으로 매핑되고 position이 stack으로 설정된 geom\_col 레이어 생성  
 geom\_col(aes(x = 대계열, y = 취업자, fill = 과정구분), position = 'dodge') +  
 ## X축이 대계열, Y축이 취업자, lable이 comma형 취업자, fill은 과정구분으로 매핑되고 position은 position\_stack()으로 vjust가 0.5인 stack형으로 설정한 geom\_text 레이어 추가  
 geom\_text(aes(x = 대계열, y = 취업자, label = scales::comma(취업자, accuracy = 1), fill =과정구분), position = position\_dodge(width = 0.9), size = 2, vjust = -1)  
  
p\_bar\_data\_dodge



실행결과 8-dodge형 막대그래프에 데이터 값 표현

# 숫자만 가지고도 안돼. 그림도 넣어야 해 – 축 라벨에 이미지 넣기

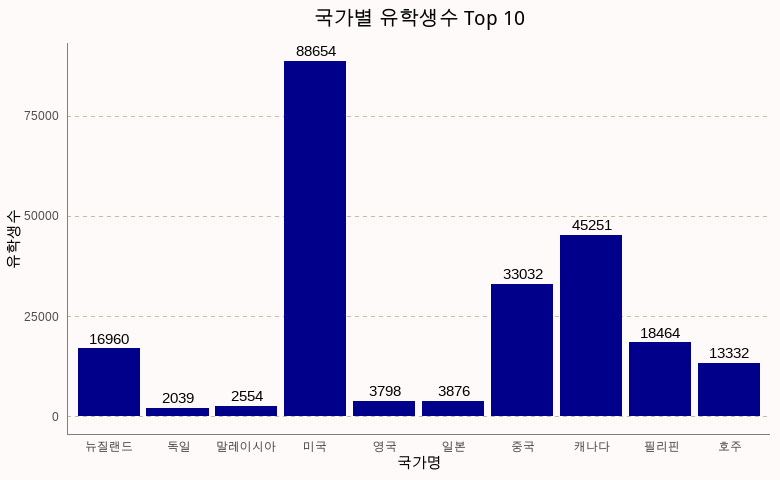
축에 이미지를 넣어 표현하면 시각화를 보다 보기 좋게 꾸며줄 수 있다. 이 방법은 단지 막대 그래프에서만 사용할 수 있는 방법이 아니고 축을 가진 모든 시각화에서 사용이 가능하다. 축 라벨에 이미지를 넣는 실습을 위해 국가별 데이터가 포함된 데이터를 사용하도록 하겠다. 다음과 같이 전처리한 데이터를 사용한다.

## 연도별 유학국가별 유학생수 엑셀파일에서   
df\_nation <- read\_xlsx('파일경로/연도별 유학국가별 유학생수.xlsx',  
 ## Sheet0 시트에서 처음 2라인을 스킵하고  
 sheet = 'Sheet0', skip = 2,   
 ## 열 타입을 설정  
 col\_types = c('numeric', 'text', rep('numeric', 25)),  
 ## 첫번째 행은 열 이름  
 col\_names = TRUE)  
  
## 읽어온 데이터에서  
df\_nation <- df\_nation |>  
 ## 학년도가 NA가 아니면서 학제가 '소계'인 데이터를 필터링하고  
 filter(!is.na(학년도), 학제 == '소계') |>  
 ## 필요하지 않는 열은 제거  
 select(!contains(c('계', '학제', '기타', '미확인', '그외동남아', '남미'))) |>  
 ## 학년도 열을 제외한 모든 열을 국가명열로, 데이터는 유학생수로 설정하는 긴 형태의 데이터 변화  
 pivot\_longer(-'학년도', names\_to = '국가명', values\_to = '유학생수')  
  
df\_nation\_top10 <- df\_nation |>  
 ## 국가명으로 그룹화하고  
 group\_by(국가명) |>  
 ## 유학생수의 합계를 산출  
 summarise(sum = sum(유학생수)) |>  
 ## 합계의 역순으로 정렬하고  
 arrange(desc(sum)) |>  
 ## 위에서 10개의 행만 저장  
 top\_n(10)  
  
df\_nation\_top10

## # A tibble: 10 x 2  
## 국가명 sum  
## <chr> <dbl>  
## 1 미국 88654  
## 2 캐나다 45251  
## 3 중국 33032  
## 4 필리핀 18464  
## 5 뉴질랜드 16960  
## 6 호주 13332  
## 7 일본 3876  
## 8 영국 3798  
## 9 말레이시아 2554  
## 10 독일 2039

유학생수가 많은 10개 나라를 저장한 df\_nation\_top10을 막대 그래프로 다음과 같이 시각화한다. 아직은 국가별 정렬이 이루어지지 않아 그래프가 다소 들쭉날쭉 보일 것이다.

p\_nation\_top10 <- df\_nation\_top10 |>  
 ggplot(aes(x = 국가명, y = sum)) +  
 geom\_col(fill = 'dark blue') +  
 geom\_text(aes(x = 국가명, y = sum, label = sum), vjust = -0.5) +   
 labs(title = '국가별 유학생수 Top 10', y = '유학생수')  
  
p\_nation\_top10



실행결과 8-유학생 top 10 국가 막대그래프

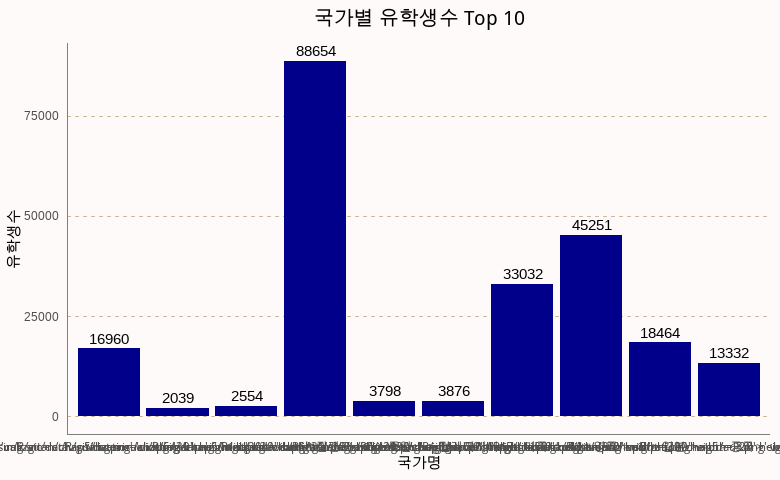
이제 X축의 국가 이름대신 각 국가의 국기를 표시해보도록 하겠다. 그렇다면 먼저 국기가 필요하겠다. 국기는 나무위키의 국가별 GDP 순위 사이트[[2]](#footnote-26)에서 해당 국가의 국기를 다운로드했다. 각각의 국기 파일명을 flag\_국가명으로 저장하고 이를 국가명, 파일 경로를 가지는 데이터프레임을 만들었다. 이 데이터프레임을 사용하여 HTML형태의 문자열을 만들고 이 문자열을 국가명으로 접근할 수 있도록 각각의 문자열에 setNames()를 사용하여 이름을 붙여준다. setNAmes()는 개별 객체에 이름을 붙여주는 함수로 이름을 설정해준 이유는 각각의 국가이름 라벨 대신 각각의 국가의 이름에 할당된 HTML 문자열로 대체해주기 위해서이다.

## 다운로드된 국기 이미지의 파일 경로를 개별 변수에 저장(이미지 저장 경로는 사용자의 경로를 설정하라.)  
flag\_usa <- '아이콘 이미지 저장 폴더 경로/usa.png'  
flag\_canada <- '아이콘 이미지 저장 폴더 경로/can.png'  
flag\_china <- '아이콘 이미지 저장 폴더 경로/chi.png'  
flag\_phi <- '아이콘 이미지 저장 폴더 경로/phi.png'  
flag\_nz <- '아이콘 이미지 저장 폴더 경로/nz.png'  
flag\_aus <- '아이콘 이미지 저장 폴더 경로/aus.png'  
flag\_jap <- '아이콘 이미지 저장 폴더 경로/jap.png'  
flag\_eng <- '아이콘 이미지 저장 폴더 경로/eng.png'  
flag\_mal <- '아이콘 이미지 저장 폴더 경로/mal.png'  
flag\_ger <- '아이콘 이미지 저장 폴더 경로/ger.png'  
  
## 국가명과 국기 경로명을 가지는 데이터프레임 생성  
flags <- data.frame(nations = c('미국', '캐나다', '중국', '뉴질랜드', '필리핀', '호주', '영국', '일본', '말레이시아', '싱가폴'), flag\_path = c(flag\_usa, flag\_canada, flag\_china, flag\_nz, flag\_phi, flag\_aus, flag\_eng, flag\_jap, flag\_mal, flag\_sing))  
  
## 이미지 소스와 가로, 세로 크기를 설정하는 HTML 태그을 만들고 각각의 국가이름을 붙여준 문자열 벡터를 생성  
labels <- setNames(  
 paste0("<img src='", flags$flag\_path, "' width='30' height = '20'> <br> ", flags$nations), flags$nations)

## 미국   
## "<img src='c:/R/git/datavisualization/chap5/usa.png' width='30' height = '20'> <br> 미국"

이제 국기 경로명이 설정된 HTML 태그 문자열 벡터가 완성되었다. 이 과정이 복잡하다면 문자열 벡터를 수작업으로 하나하나 만들어도 관계없다. 만들어진 문자열 벡터를 X축의 scale\_\*()에 labels로 설정하면 다음과 같이 표현된다.

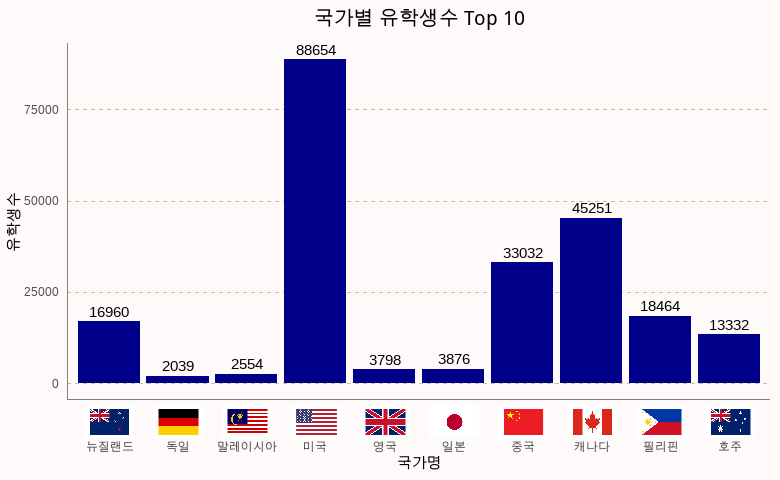
p\_nation\_top10\_1 <- p\_nation\_top10 +  
 ## X축을 이산형 스케일로 설정하고 앞에서 만든 html 형 라벨을 붙임   
 scale\_x\_discrete(labels = labels)  
  
p\_nation\_top10\_1



실행결과 8-라벨에 HTML 태그를 붙인 막대그래프

국기가 표시될 것으로 예상했던 X축에 알수없는 문자들만 가득하다. 이는 HTML 태그를 HTML 형태로 해석하고 동작하는 것이 아니고 단순 문자열로 해석하고 동작했기 때문이다. HTML 태그를 HTML 형태로 해석하고 동작시키기 위해서는 ggtext패키지의 element\_markdown()을 사용한다. element\_markdown()은 theme() 함수내에서 사용되어 테마요소를 마크다운(markdwon)[[3]](#footnote-28)으로 인식하고 실행시키는 함수이다.

## ggtext 패키지 설치  
if(!require(ggtext)) {  
 install.packages('ggtext')  
 library(ggtext)  
}  
  
p\_nation\_top10\_1 +  
 ## axis.text.x의 테마요소를 마크다운 형태로 설정  
 theme(axis.text.x = ggtext::element\_markdown())



실행결과 8-라벨에 이미지를 붙인 막대그래프

# top 10을 구하라 - 순위 막대 그래프

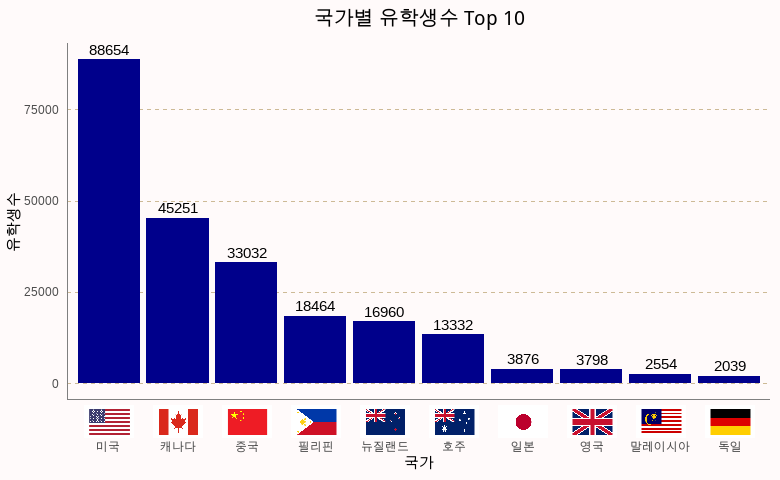
앞서 그렸던 유학생 막대 그래프를 보면 순서가 가나다 순서로 되어 있어 국가별로 순위를 비교하가기가 어렵다. 이를 데이터 값에 따라 정렬할 필요가 있다.

막대그래프를 정렬하기 위해서는 먼저 데이터의 순서대로 X축의 변량을 정렬해주어야 한다. 이를 위해 fct\_reorder()를 사용하였다. fct\_reorder()는 tidyverse 패키지 중 forcats 패키지에서 제공하는 함수로 팩터나 문자열 벡터의 순서를 다시 설정해주는 함수이다.

fct\_reorder(.f, .x, .fun = median, …, .desc = FALSE)  
- .f : 순서를 조절할 팩터나 문자열 벡터  
- .x : 정렬의 기준이 될 팩터나 문자열 벡터  
- fun : 정렬에 사용할 함수 설정  
- … : 정렬에 사용할 함수에 필요한 매개변수  
- .desc : 내림차순으로 정렬할지를 설정하는 논리값

앞선 막대 그래프에서 X축으로 사용했던 국가명을 유학생 합계(sum)을 기준으로 내림차순으로 순서를 다시 정렬한 후 그래프를 다시 그려주면 유학생수로 정렬된 막대그래프가 그려진다.

## fct\_reorder를 사용하여 df\_nation\_top10$sum을 기준으로 내림차순 정렬한 df\_nation\_top10$국가명을 팩터로 변환하여 df\_nation\_top10$국가명에 설정  
df\_nation\_top10$국가명 <- fct\_reorder(df\_nation\_top10$국가명, df\_nation\_top10$sum, .desc = TRUE)  
  
## ggplot에 매핑된 X축 변수가 변경되었으므로 처음부터 다시 코딩  
df\_nation\_top10 |>  
 ggplot(aes(x = 국가명, y = sum)) +  
 geom\_col(fill = 'dark blue') +  
 geom\_text(aes(x = 국가명, y = sum, label = sum), vjust = -0.5) +   
 labs(title = '국가별 유학생수 Top 10', x = '국가', y = '유학생수') +  
 scale\_x\_discrete(labels = labels) +  
 ## X축 텍스트의 형태를 element\_markdown()을 사용하여 마크다운 언어로 적용  
 theme(axis.text.x = ggtext::element\_markdown())



실행결과 8-데이터가 정렬된 막대그래프

# 여러 종류이 널리 퍼져있는 데이터는? - 업셋 그래프

중복이 허용되면서 변량이 많지 않은 데이터를 시각화하는 방법으로 많이 활용되는 것이 벤다이어그램이다. 벤다이어그램은 10장에서 설명하겠지만 원으로 변량을 표현해 중복에 대한 표현이 가능한 장점이 있다. 하지만 원(변량)이 3개를 넘어가면 매우 혼란스러워진다. 이 벤다이어그램의 단점을 극복하기 위해 2014년에 제안된 시각화 방법이 업셋 그래프이다.[[4]](#footnote-31)

업셋 그래프는 그래프의 아래쪽에 해당 데이터를 구성하는 세트의 조합이 표현되고 위쪽에는 해당 세트의 조합에 대한 데이터 값을 표현하는 막대 그래프가 표현된다. 이렇게 구성함으로써 단일 세트만 가능했던 막대 그래프에 변량의 세트에 대한 표현이 가능해져서 벤다이어그램에서 해석이 어려웠던 다양한 데이터의 조합에 대한 시각화에 매우 효과적이다.

업셋 그래프는 3개 이상 30개 미만의 집합 데이터에 가장 적합하다 . 4세트 미만의 경우 익숙한 벤 다이어그램이 친숙하다. 또 업셋 그래프는 집합 데이터의 분포를 분석하는 데 적합하기때문에 교차하는 집합의 조합의 빈도확인에 매우 효과적이다.[[5]](#footnote-38)

그럼 이제 업셋 그래프를 만들어 보겠다. df\_취업통계 데이터는 각 학과를 7개의 학제(전문대학(2년제), 전문대학(3년제), 대학교, 산업대학 등)로 구분되어 있다. 따라서 같은 학과명을 가지더라도 서로 학제가 달라 최대 7개의 행이 있을 수 있다. 그래서 동일한 학과명이 여러 학제에 걸쳐 존재하는 경우가 많은데 이 분포를 알아내기 위해 업셋 그래프를 만든다.

이 업셋 그래프를 만들기 위해 사용하는 데이터는 이 df\_취업통계 데이터로 각각의 학과명이 속한 학제를 리스트로 만든 열이 필요하다. 이 열을 기준으로 빈도를 표현한 막대 그래프를 그릴 것이다. 이를 위해 다음과 같이 데이터를 전처리 한다.

## df\_취업통계에서  
df\_과정구분\_upset <- df\_취업통계 |>  
 ## 학과명으로 그룹화  
 group\_by(학과명) |>  
 ## 학제를 리스트로 만든 학과리스트 열을 생성(각각의 그룹의 행에 학과리스트 열은 모두 같은 값을 가지기 때문에 맨 아래에서 중복을 제거)  
 mutate(학제리스트 = list(unique(sort(학제)))) |>  
 ## 필요한 열만 선택  
 select(3:6, 8, 학제리스트, 11) |>  
 ## 중복을 제거  
 unique()

## df\_과정구분\_upset |> View() 로 데이터를 확인 가능  
View(df\_과정구분\_upset) ## 'V'는 대문자임에 주의

전처리가 끝났으면 본격적으로 업셋 그래프를 만든다. 이를 위해 먼저 ggupset 패키지를 설치해야한다.

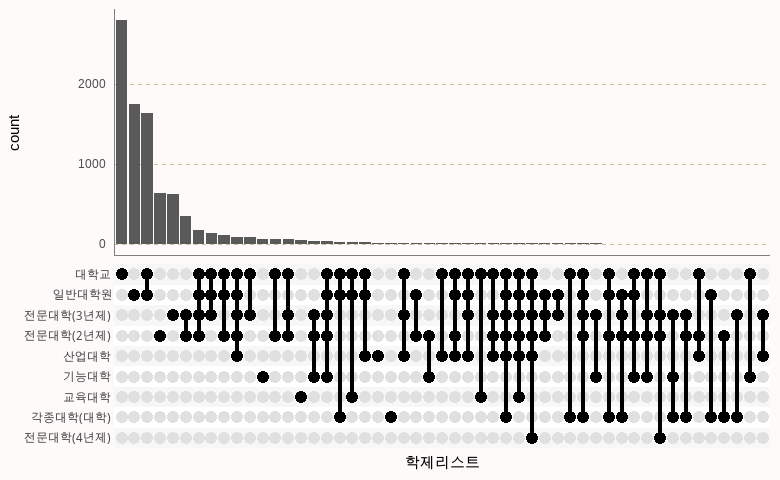
## ggupset 패키지 설치  
if(!require(ggupset)) {  
 install.packages('ggupset')  
 library(ggupset)  
}

업셋 그래프를 그리는 코드는 다음과 같다. 먼저 학과명의 조합 리스트로 구성된 학과명을 X축으로 매핑한 geom\_bar 레이어를 생성한다. 이후 X축의 스케일를 업셋 스케일로 바꾸어주면 업셋의 기본 그래프가 생성된다. 축의 스케일을 업셋 스케일로 바꾸려면 scale\_x\_upset()을 사용한다.

scale\_x\_upset(order\_by = c(“freq”, “degree”), n\_sets = Inf, n\_intersections = Inf, sets = NULL, intersections = NULL, reverse = FALSE, ytrans = “identity”, …, position = “bottom”)  
- order\_by : X축의 순서 설정을 어떻게 할지 설정  
- n\_sets : 전체 셋의 최대값 설정  
- n\_intersections : 표시될 교차 셋의 최대값 설정  
- sets : 표시하고 싶은 셋의 이름 벡터 설정  
- intersections : 표시될 교차 셋의 이름 벡터 설정  
- reverse : 교차값의 역순으로 설정할 논리값 설정  
- ytrans : Y축의 변환 설정  
- … : 이산형 스케일과 관련한 추가 매개변수  
- position : X축을 아래에 위치할 것인지 위에 위치할 것인지 설정

빈도수를 표현하는 geom\_bar()의 X축 스케일을 scale\_x\_upset()으로 설정하여 업셋 그래프를 그리는 코드는 다음과 같다.

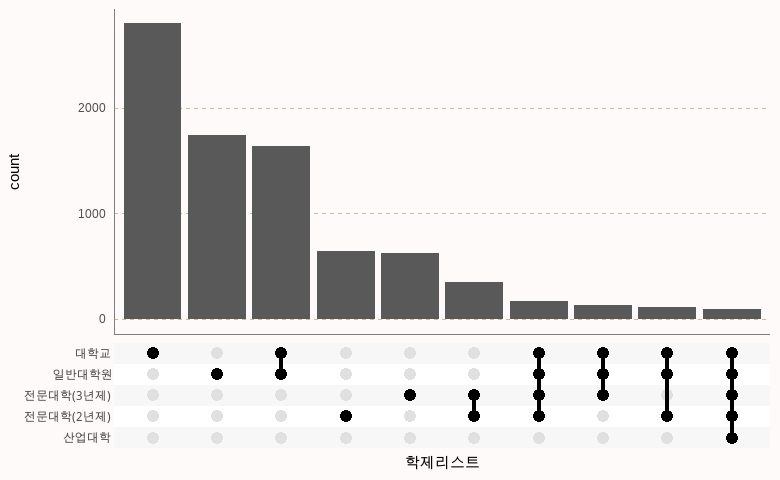
df\_과정구분\_upset |> ggplot() +  
 ## X축을 학제리스트로 매핑한 geom\_bar 레이어 생성  
 geom\_bar(aes(x=학제리스트)) +  
 ## X축 스케일을 업셋 스케일로 설정  
 scale\_x\_upset()



실행결과 8-기본 업셋 그래프

7개의 학제에 대한 조합이 표현되다보니 조합의 수가 너무 많아 보인다. 이 중 빈도가 많은 10개의 조합만 표시하도록하는 코드는 다음과 같다.

ggplot(df\_과정구분\_upset, aes(x=학제리스트)) +  
 geom\_bar() +  
 ## n\_intersection을 10으로 설정해서 X축 변량을 10개로 한정  
 scale\_x\_upset(n\_intersections = 10)



실행결과 8-기본 업셋 그래프

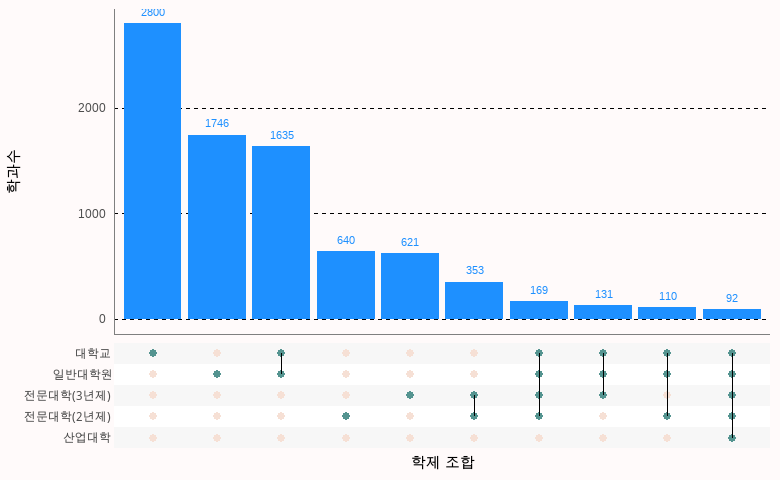
앞의 그래프를 보면 대학교에만 존재하는 학과가 가장 많고 다음은 일반대학원이다. 세번째로 많은 분포는 대학교와 일반대학원에 같이 존재하는 학과가 있다는 것을 알 수 있다.

기본 업셋 그래프가 만들어졌으니 이제 세부 설정을 해서 그래프를 꾸며보겠다. 업셋 그래프에서 사용하는 추가적인 테마를 설정하기 위해서는 theme\_combmatrix()를 사용한다.

theme\_combmatrix(combmatrix.label.make\_space = TRUE, combmatrix.label.width = NULL, combmatrix.label.height = NULL, combmatrix.label.extra\_spacing = 3, combmatrix.label.total\_extra\_spacing = unit(10, “pt”), combmatrix.label.text = NULL, combmatrix.panel.margin = unit(c(1.5, 1.5), “pt”), combmatrix.panel.striped\_background = TRUE, combmatrix.panel.striped\_background.color.one = “white”, combmatrix.panel.striped\_background.color.two = “#F7F7F7”, combmatrix.panel.point.size = 3, combmatrix.panel.line.size = 1.2, combmatrix.panel.point.color.fill = “black”, combmatrix.panel.point.color.empty = “#E0E0E0”, …)  
- combmatrix.label.make\_space : Y축 라벨이 없어져도 업셋 스케일의 매트릭스를 남길것인지를 설정하는 논리값  
- combmatrix.label.width, combmatrix.label.height : 업셋 라벨의 너비와 높이 설정  
- combmatrix.label.extra\_spacing : 라벨 행의 높이에 추가할 공간의 설정  
- combmatrix.label.text : element\_text()를 사용하여 X축 라벨의 세부 설정  
- combmatrix.panel.margin : 조합 매트릭스와 플롯간의 여백 설정  
- combmatrix.panel.striped\_background : 플롯의 배경을 줄무늬 형태로 설정하는 논리값  
- combmatrix.panel.striped\_background.color.one, combmatrix.panel.striped\_background.color.two : 줄무의 배경을 설정할 경우 설정할 색 설정  
- combmatrix.panel.point.size : 조합 매트릭스에 사용할 포인트의 크기 설정  
- combmatrix.panel.line.size : 조합 매트릭스에 사용할 선의 크기 설정  
- combmatrix.panel.point.color.fill : 조합 매트릭스에 사용할 포인트의 내부 색 설정  
- combmatrix.panel.point.color.empty : 조합 매트릭스에 사용할 빈 포인트의 내부 색 설정  
- … : theme()의 추가적인 매개변수

theme\_combmatrix()를 사용하여 세부설정한 업셋 그래프의 코드는 다음과 같다.

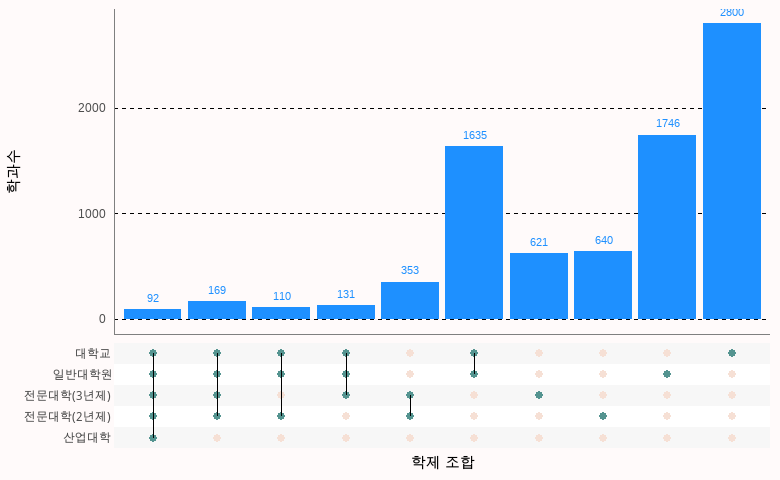
df\_과정구분\_upset |>   
 ggplot() +  
 ## X축을 학제리스트로 매핑하고 fill을 설정한 geom\_bar 레이어 생성  
 geom\_bar(aes(x=학제리스트), fill = "dodgerblue") +  
 ## 막대에 데이터 값을 표기하는 geom\_text 레이어 추가  
 geom\_text(aes(x=학제리스트, label = ..count..), stat = 'count', vjust = -1, size = 3, color = "dodgerblue") +  
 ## X축의 스케일을 upset 스케일로 설정하는데 조합수를 10개로 한정  
 scale\_x\_upset(n\_intersections = 10) +  
 ## theme\_combmatrix()를 사용하여 업셋 그래프의 세부 테마를 설정   
 theme\_combmatrix(combmatrix.label.make\_space = TRUE,   
 combmatrix.panel.point.size = 2,   
 combmatrix.panel.line.size = 0.4,  
 combmatrix.panel.point.color.fill = "#549490",  
 combmatrix.panel.point.color.empty = "#f6e0d5",  
 panel.grid = element\_blank()) +  
 # theme\_minimal() +  
 ylab("학과수") + xlab("학제 조합") +  
 theme(panel.grid = element\_blank())



실행결과 8-세부 설정된 업셋 그래프

앞의 업셋 그래프의 조합 매트릭스의 조합이 많은 순으로 정렬 순서를 바꾸면 다음과 같이 바꿀 수 있다.

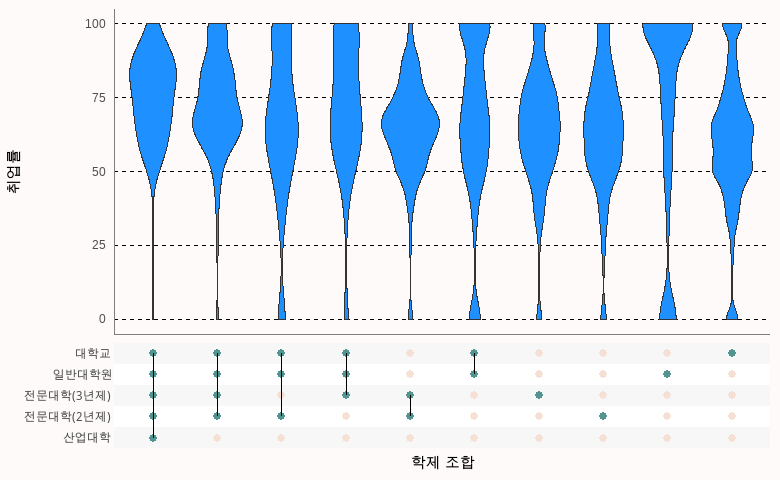
df\_과정구분\_upset |>  
 ggplot() +  
 geom\_bar(aes(x=학제리스트), fill = "dodgerblue") +  
 geom\_text(aes(x=학제리스트, label = ..count..), stat = 'count', vjust = -1, size = 3, color = "dodgerblue") +  
 ## order\_by를 'degree'로 설정하여 매트릭스 조합을 정렬 기준으로 맞추고 내림차순으로 설정하기 위해 reverse를 설정  
 scale\_x\_upset(n\_intersections = 10,   
 order\_by="degree",   
 reverse = TRUE) +  
 theme\_combmatrix(combmatrix.label.make\_space = TRUE,   
 combmatrix.panel.point.size = 2,   
 combmatrix.panel.line.size = 0.4,  
 combmatrix.panel.point.color.fill = "#549490",  
 combmatrix.panel.point.color.empty = "#f6e0d5",  
 panel.grid = element\_blank()) +  
 ylab("학과수") + xlab("학제 조합") +  
 theme(panel.grid = element\_blank())



실행결과 8-조합순으로 정렬된 업셋 그래프

업셋 그래프에 사용되는 상위 그래프는 막대 그래프외에 모든 그래프를 사용할 수 있다. 다만 X축의 조합 매트릭스를 설정함으로써 데이터를 잘 설명할 수 있는지 고려하여 설정하는 것이 좋다. 다음은 앞의 빈도수의 막대 그래프를 취업률에 대한 바이올린 그래프로 바꾼 코드이다.

df\_과정구분\_upset |>   
 ggplot() +  
 ## 상부 그래프의 레이어를 geom\_violin 레이어로 생성  
 geom\_violin(aes(x=학제리스트, y = 취업률\_계), fill = "dodgerblue") +  
 scale\_x\_upset(n\_intersections = 10, order\_by="degree", reverse = TRUE) +  
 ## 조합 행렬 테마의 설정  
 theme\_combmatrix(combmatrix.label.make\_space = TRUE,   
 combmatrix.panel.point.size = 2,   
 combmatrix.panel.line.size = 0.4,  
 combmatrix.panel.point.color.fill = "#549490",  
 combmatrix.panel.point.color.empty = "#f6e0d5",   
 panel.grid = element\_blank()) +  
 ylab("취업률") + xlab("학제 조합") +  
 theme(panel.grid = element\_blank())



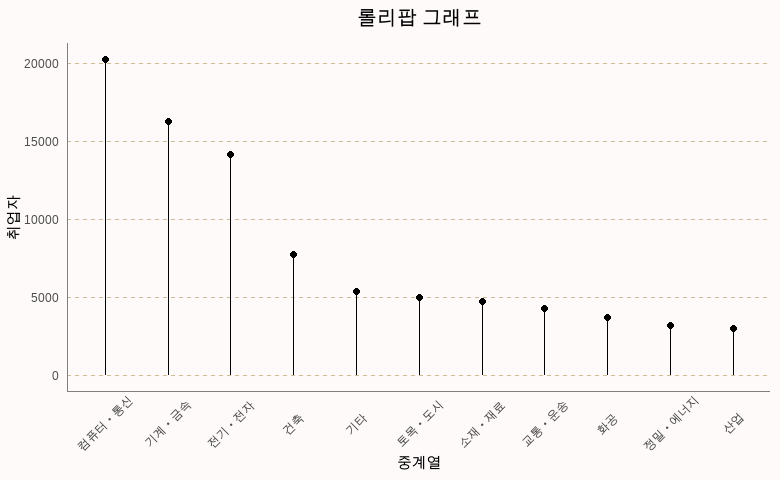
실행결과 8-바이올린 업셋 그래프

# 이것은 막대 사탕이 아니다. - 롤리팝 그래프

롤리팝 그래프는 막대사탕의 상품명에서 유래된 그래프이다. 막대그래프와 유사하지만 그 표현을 막대가 아닌 막대사탕처럼 표현한다는데에서 유래했다. 원으로 표시된 데이터 점으로부터 축까지를 선으로 이어 데이터를 표현하는 방식의 그래프이다. 이 롤리팝 그래프는 전용 함수가 제공되는 것이 아니고 geom\_point()와 geom\_segment()를 사용하여 구현할 수 있다.

## 롤리팝 그래프를 위해 데이터 전처리  
df\_lolipop <- df\_취업통계\_계열별 |>  
 ## 공학계열의 중계열을 그룹화  
 filter(대계열 == '공학계열') |> group\_by(중계열) |>  
 ## 취업자 합계를 구하고 취업자의 내림차순으로 정렬  
 summarise(취업자 = sum(취업자)) |> arrange(desc(취업자))  
  
## fct\_reorder로 중계열을 취업자를 기준으로 내림차림 정렬  
df\_lolipop$중계열 <- fct\_reorder(df\_lolipop$중계열, df\_lolipop$취업자, .desc = TRUE)  
  
## ggplot 객체 생성  
p\_lolipop <- df\_lolipop|>  
 ggplot() +   
 labs(x = '중계열', y = '취업자')

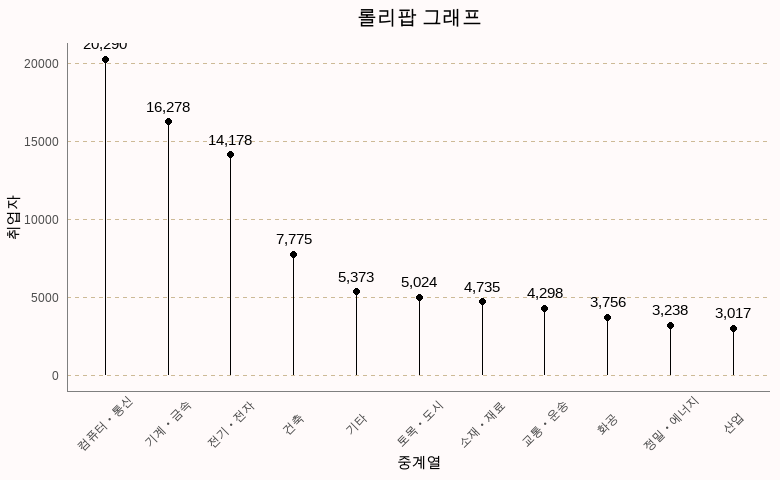
p\_lolipop1 <- p\_lolipop +  
 ## Y축을 0부터 취업자까지 선을 그린 geom\_segement 레이어 생성  
 geom\_segment(aes(x = 중계열, xend = 중계열, y = 0, yend = 취업자)) +  
 ## X축에 중계열, Y축에 취업자로 매핑한 geom\_point 레이어 추가  
 geom\_point(aes(x = 중계열, y = 취업자)) +   
 labs(title = '롤리팝 그래프') +   
 ## X축 텍스트 겹침 방지를 위해 45도 기울임  
 theme(axis.text.x = element\_text(angle = 45))  
  
p\_lolipop1



실행결과 8-롤리팝 그래프

롤리팝 그래프에 데이터 값을 표현하려면 다음과 같이 geom\_text()레이어를 추가해 준다.

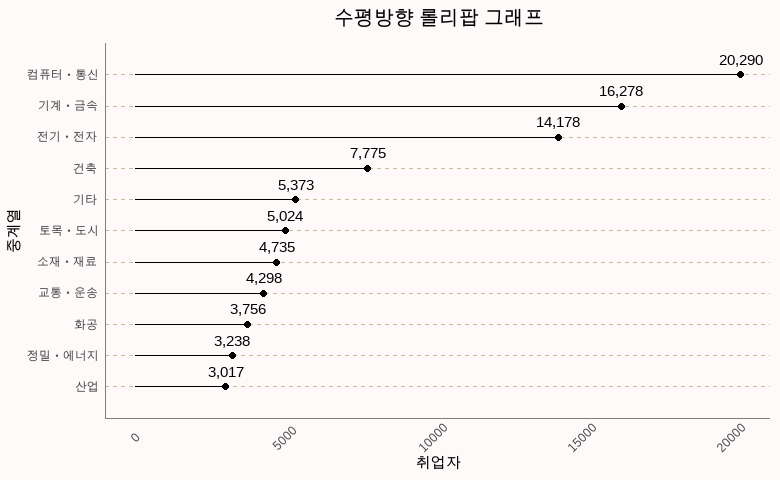
p\_lolipop2 <- p\_lolipop1 +   
 ## X축이 중계열, Y축이 취업자, label이 천단위 구분자가 포함된 취업자로 매핑되고 vjust를 설정한 geom\_text 레이어 추가  
 geom\_text(aes(x = 중계열, y = 취업자, label = scales::comma(취업자)), vjust = -1)  
  
p\_lolipop2



실행결과 8-데이터 값이 표현된 롤리팝 그래프

앞의 롤리팝 그래프는 X축에 표현되는 변량의 수가 적절하기 때문에 X축에 표현되어도 별다른 혼동을 느끼지 않는다. 하지만 이름이 다소 긴 중계열은 서로 계열 이름이 겹처 어쩔 수 없이 45도 기울줄 수 밖에 없었다. 이러한 상황을 피하기 위해 최근에는 아래 방향으로 길게 늘여뜨려주는 형태로 많이 만든다. 이를 위해서 막대를 수직방향에서 수평방향으로 전환하여야하는데 coord\_flip()으로 축을 전환함으로써 가능하다.

p\_lolipop2 +   
 ## X축과 Y축의 위치를 바꿈  
 coord\_flip() +   
 ## X축의 순서를 반전하고 축의 범위를 1칸 늘려줌  
 scale\_x\_discrete(limits = rev, expand = c(0,1)) +  
 labs(title = '수평방향 롤리팝 그래프')



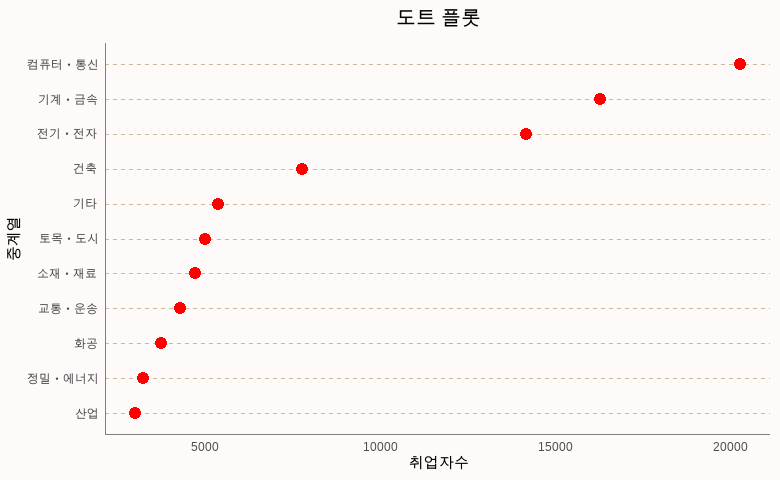
실행결과 8-수평방향 롤리팝 그래프

# 점 하나만으로도 강렬하게 – 도트 그래프

도트 플롯는 세로형 롤리팝 그래프와 거의 동일하다. 단 하나의 차이는 축부터 데이터 점까지을 잇는 선이 없다는 것이다. 실제 값과 관련해 순위와 데이터 값간의 거리에 초점을 맞추어 볼 수 있는 시각화의 방법이다. 도트 플롯도 도트 플롯을 위한 전용 함수가 제공되지 않고 geom\_point()를 사용하여 생성할 수 있다.

p\_dot <- df\_lolipop|>  
 ggplot()+  
 labs(x = '중계열', y = '취업자수')

p\_dot1 <- p\_dot +  
 ## X축을 중계열, Y축을 취업자, group을 중계열로 매핑한 geom\_point 레이어 생성  
 geom\_point(aes(x = 중계열, y = 취업자, group = 중계열), size = 3, color = 'red') +   
 ## X축의 순서를 역순으로 한 이산형 스케일 설정  
 scale\_x\_discrete(limits = rev) +  
 ## 축 전환  
 coord\_flip() +   
 labs(title = '도트 플롯')  
  
p\_dot1



실행결과 8-도트 플롯

1. fill 형태의 막대 그래프에 대한 데이터 값 넣기는 다음장에서 설명한다. [↑](#footnote-ref-22)
2. <https://namu.wiki/w/%EA%B5%AD%EA%B0%80%EB%B3%84%20%EB%AA%85%EB%AA%A9%20GDP%20%EC%88%9C%EC%9C%84> [↑](#footnote-ref-26)
3. Markdown은 텍스트 기반의 마크업언어로 HTML과 호환이 가능하다. 최근 깃헙(Github)의 활용이 활발해지면서 부각된 언어로 특수문자와 일반 텍스트 만으로 서식문서를 만들수 있다는 장점이 있어 많이 사용되고 있다. [↑](#footnote-ref-28)
4. [Alexander Lex](https://vdl.sci.utah.edu/team/lex/), [Nils Gehlenborg](http://www.gehlenborg.com/), [Hendrik Strobelt](http://hendrik.strobelt.com/), Romain Vuillemot, [Hanspeter Pfister](http://vcg.seas.harvard.edu/people/hanspeter-pfister)  
   [**UpSet: Visualization of Intersecting Sets**](https://sci.utah.edu/~vdl/papers/2014_infovis_upset.pdf)  
   IEEE Transactions on Visualization and Computer Graphics (InfoVis), 20(12): 1983–1992, [doi:10.1109/TVCG.2014.2346248](http://dx.doi.org/10.1109/TVCG.2014.2346248), 2014. [↑](#footnote-ref-31)
5. <https://upset.app/> [↑](#footnote-ref-38)