# VI. 분포(Distribution)의 시각화

분포의 시각화는 전체 데이터가 어떻게 세부적으로 분포하는지를 표현하는 시각화 방법을 말한다. 보통 전체 데이터를 특정 구분자에 의해 분류하고 이에 대한 사례수나 비율을 나타내는 시각화가 이에 속한다. 대표적인 분포의 시각화는 히스토그램, 밀도 분포 그래프, 박스 플롯 등이 있다.

## 1. 히스토그램(Histogram)

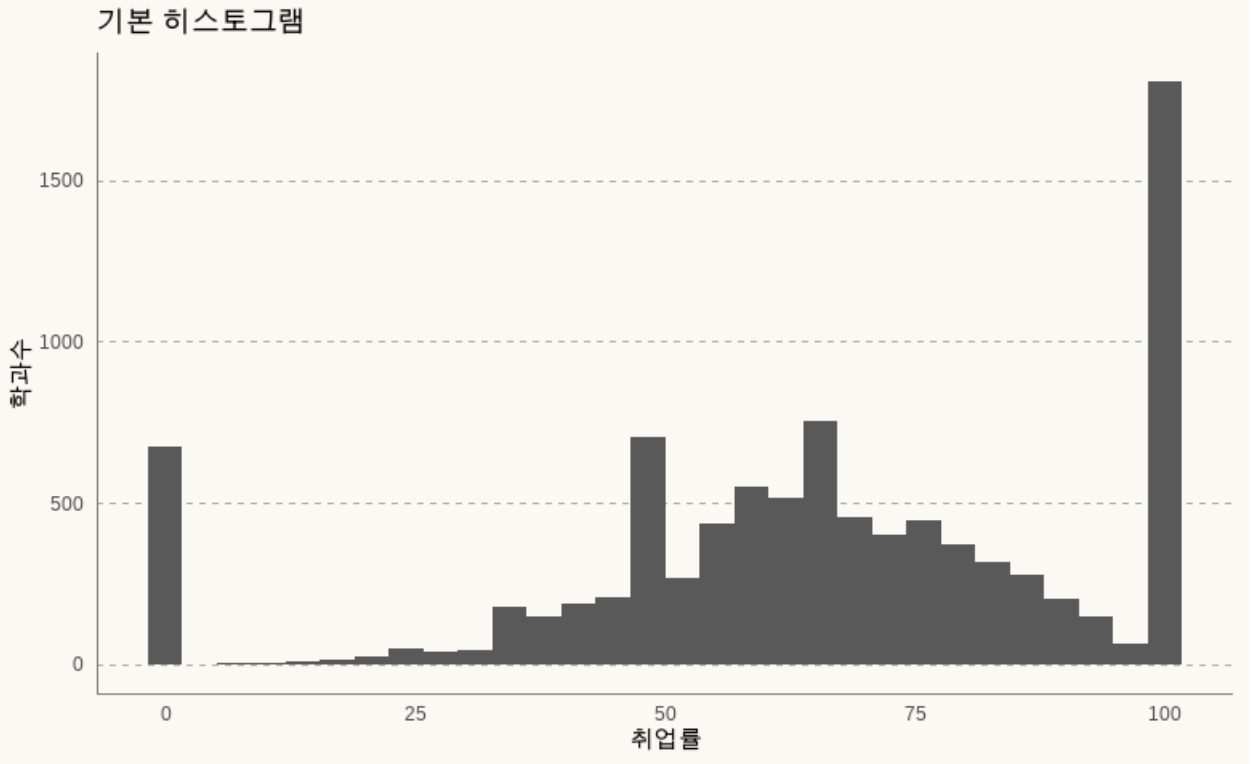
히스토그램은 데이터의 특정 변수에 따른 사례수를 나타내는 시각화이다. 특정 변수의 일정한 급간에 몇개의 사례가 있는지는 도수분포라고하고 이를 시각화한 것이 도수분포표, 즉 히스토그램니다. 분포의 시각화에 가장 대표적인 시각화가 히스토그램이고 우리가 그래프를 배울때 가장 먼저 배우는 그래프가 막대그래프를 사용한 히스토그램이다.

실제 업무에서 만드는 문서에 데이터의 시각화를 넣을때 히스토그램을 넣는 경우는 사실 좀 드물다. 보통 화려한 스킬이 잔뜩 동원되고 컬러풀한 다양한 시각화가 사람들을 설득하기에 매우 효과적이라는 데에 대해서는 이견이 없다. 하지만 이 히스토그램은 그 화려하고 컬러풀한 시각화가 왜곡된 시각화가 아니라는 증명에 꼭 필요한 시각화이다. 시각화 작성자가 사용하고 있는 데이터가 한쪽으로 치우쳐진 왜곡된 데이터인지부터 확인해야하는 것이다. 이를 위해 문서에는 포함되지 않을 수 있지만 반드시 가장 처음에 확인해야 하는 시각화가 바로 이 히스토그램이다.

`ggplot2` 에서 히스토그램을 만드는 방법은 `geom_histogram()`을 사용하는 방법, `geom_bar()`를 이용하는 방법, `geom_col()`을 이용하는 방법, `stat_*()`를 이용하는 방법 등 다양하게 만들 수 있다. 앞선 3 장에서 이들에 대해 다루었기 때문에 이번 장에서는 실제 히스토그램을 만들때 주의해야할 점 위주로 설명하도록 하겠다.

```
## df_취업통계를 사용해 ggplot 객체 생성
p_histo <- df_취업통계 |>
  ggplot() +
  labs(x = '취업률', y = '학과수')

## geom_histogram()을 사용하여 히스토그램을 생성, 내부적으로 생성되는 Y 축 값을 표현
```

```
p_histo +
  geom_histogram(aes(x = 취업률_계, y = ..count..)) +
  labs(title = '기본 히스토그램')
```

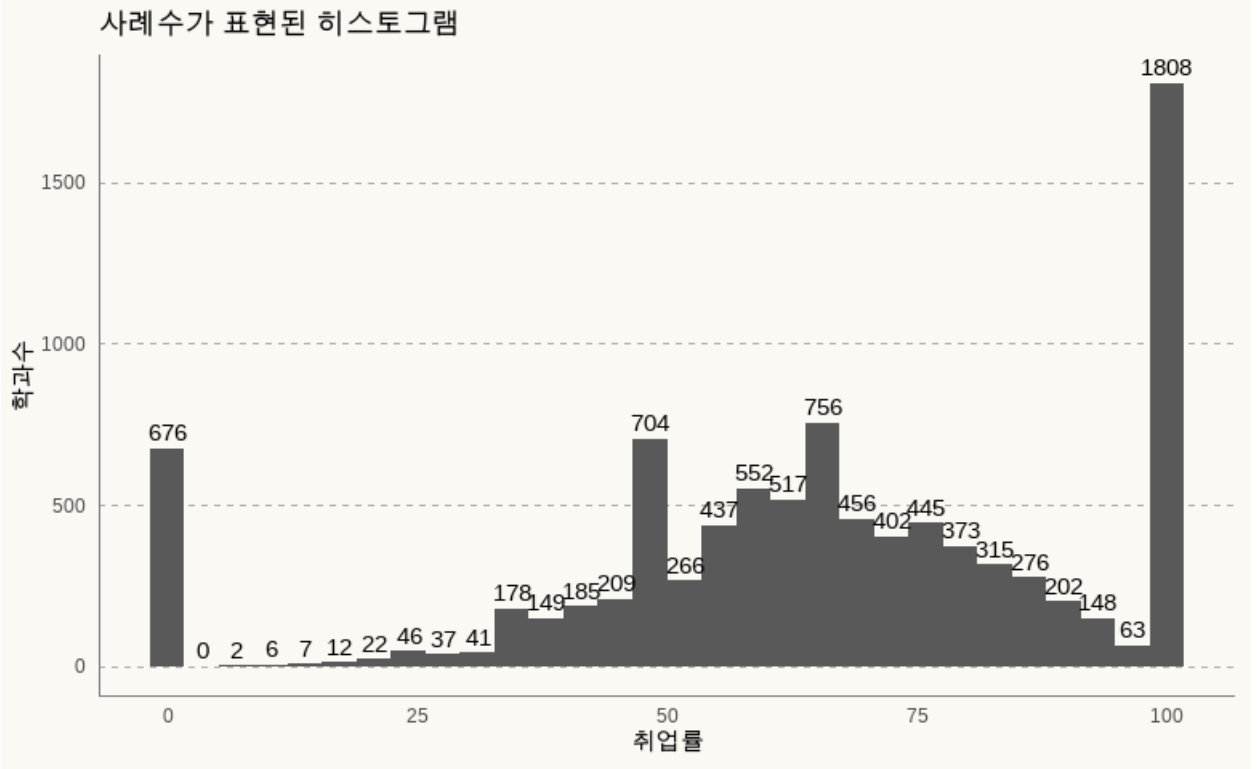



*실행결과 6-1. 기본 히스토그램*

## 1.1. 히스토그램 사례수 표현

`geom_histogram()`에서 제공하는 기본 히스토그램에는 단순 막대만 표현된다. 하지만 히스토그램을 통해 사용자는 각 구간의 정확한 값을 전달해야할 경우가 있다. 각 막대에 사례수를 표현하기 위해서는 `stat_bin()`을 사용하여 표현할 수 있다.

```
p_histo +
  geom_histogram(aes(x = 취업률_계)) +
  ## stat_bin 을 사용해 사례수를 표현, ..count..는 사례수를 산출하는 통계 변환값을 나타냄
  stat_bin(aes(x = 취업률_계, y=..count.., label=..count..), geom="text", vjust=-.5) +
  labs(title = '사례수가 표현된 히스토그램')
```




*실행결과 6-2. 사례수가 표현된 히스토그램*

## 1.2. bin, binwidth 의 설정

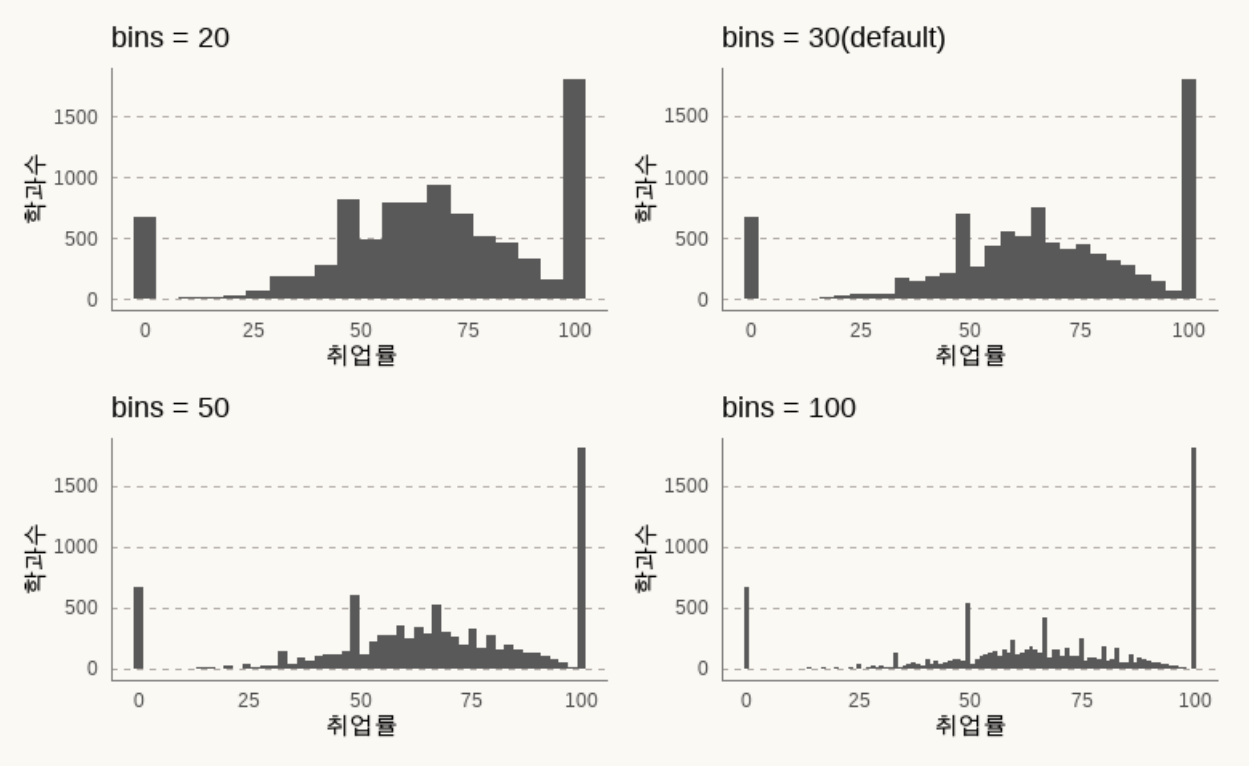
`geom_histogram()`은 대표적인 연속형 일변수 데이터의 시각화 방법이다. 앞서 설명한 바와 같이 일변량을 사용하긴 하지만 통계적으로 변환되어 감추어진 변수가 존재한다고 기술하였다. 여기에 하나 추가적으로 설명한 것이 변량의 급간을 설정하는 bin 과 binwidth 이다.

bin 은 히스토그램에 표현되는 막대의 개수를 말한다. bin 을 크게 설정하면 전체 X 축의 범위에 더 많은 막대가 나타나게 되므로 변수의 급간은 매우 작아지게 된다. 반면 binwidth 는 급간의 크기를 직접 설정하는 것이기 때문에 binwidth 를 크게 설정하면 X 축의 범위에 더 적은 막대가 나타나게 된다.

다음은 bin 값의 설정에 따른 히스토그램의 변화를 보여준다.

```
## bin 을 각기 달리한 히스토그램 생성
p_histo +
  geom_histogram(aes(x = 취업률_계), bins = 20) +
  labs(title = 'bins = 20')
```

```r
p_histo +
  geom_histogram(aes(x = 취업률_계), bins = 30) +
  labs(title = 'bins = 30(default)')

p_histo +
  geom_histogram(aes(x = 취업률_계), bins = 50) +
  labs(title = 'bins = 50')

p_histo +
  geom_histogram(aes(x = 취업률_계), bins = 100) +
  labs(title = 'bins = 100')
```

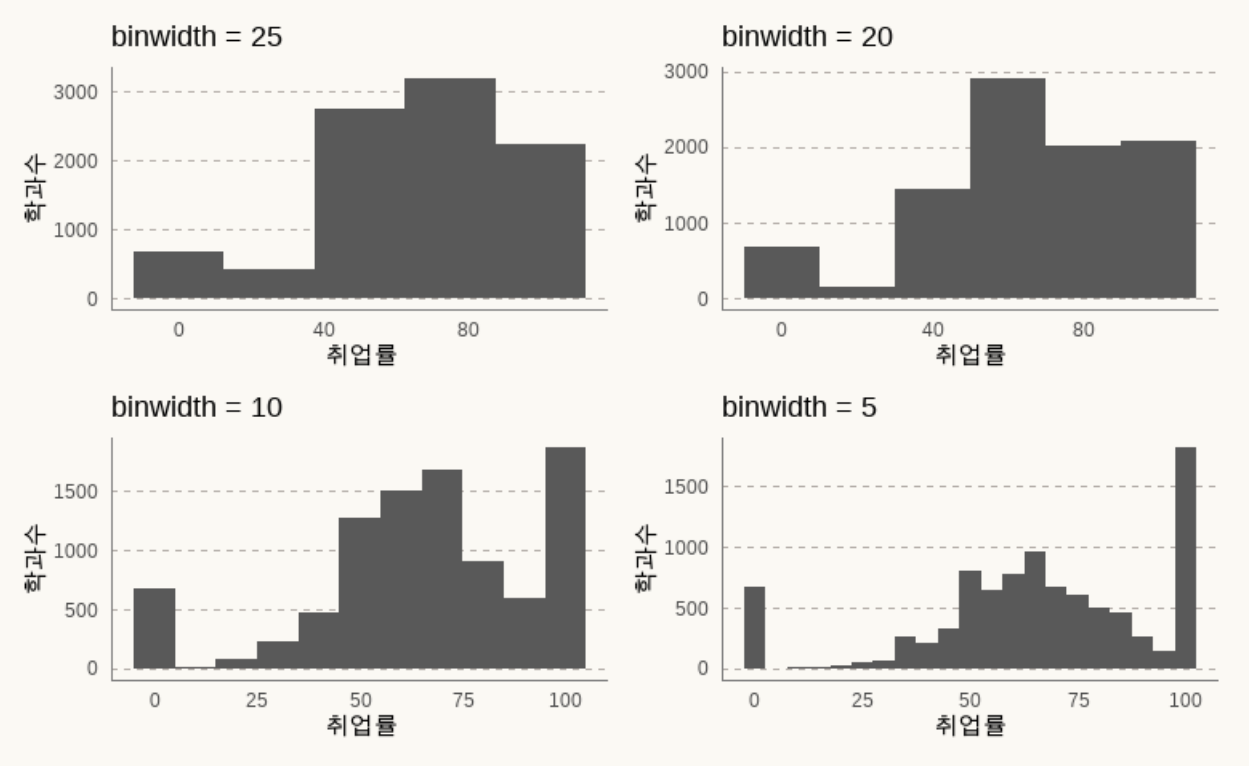



*실행결과 6-3. bin 에 따른 히스토그램*

다음은 binwidth 의 변화에 따른 히스토그램의 변화를 보여준다.

```r
p_histo +
  geom_histogram(aes(x = 취업률_계), binwidth = 25) +
  labs(title = 'binwidth = 25')

p_histo +
  geom_histogram(aes(x = 취업률_계), binwidth = 20) +
  labs(title = 'binwidth = 20')

p_histo +
```

```
  geom_histogram(aes(x = 취업률_계), binwidth = 10) +
  labs(title = 'binwidth = 10')

p_histo +
  geom_histogram(aes(x = 취업률_계), binwidth = 5) +
  labs(title = 'binwidth = 5')
```




*실행결과 6-4. binwidth 에 따른 히스토그램*

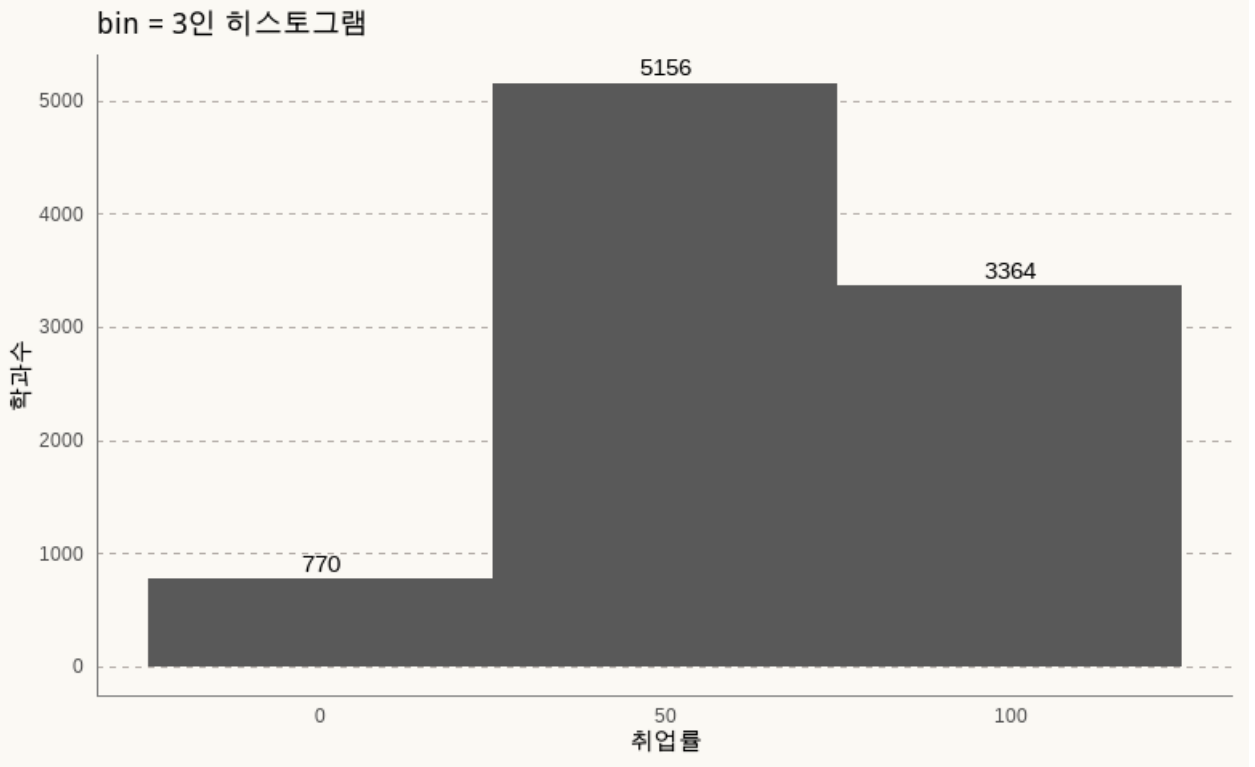
## 1.3. bin 분할 원리

히스토그램에서 bin 의 수는 하나 두개 정도로 표현되는 경우는 드물기 때문에 bin 의 범위에 대해 정확히 파악하기가 어렵다. 또 X 축의 범위가 넓은 경우에는 각 bin 의 범위를 아는 것이 데이터 분석에 큰 의미를 찾기 어려운 경우가 있고 X 축에 범위를 효과적으로 표현하기도 어렵다. 하지만 bin 이 나뉘는 원리는 알고 있어야 할 필요는 있을 것같다.

다음의 코드는 앞에서 그렸던 히스토그램의 bin 을 3 개로 한정시킨 히스토그램을 만드는 코드이다.

```
p_histo_bin <- p_histo +
  ## bin = 3 으로 설정한 geom_histogram 생성
  geom_histogram(aes(x = 취업률_계), bins = 3) +
```

```
  ## stat_bin()을 사용하여 빈도수 텍스트 레이어 생성
  stat_bin(aes(x = 취업률_계, y=..count.., label=..count..), bins = 3, geom="text
", vjust=-.5) +
  labs(title = 'bin = 3 인 히스토그램')

p_histo_bin
```



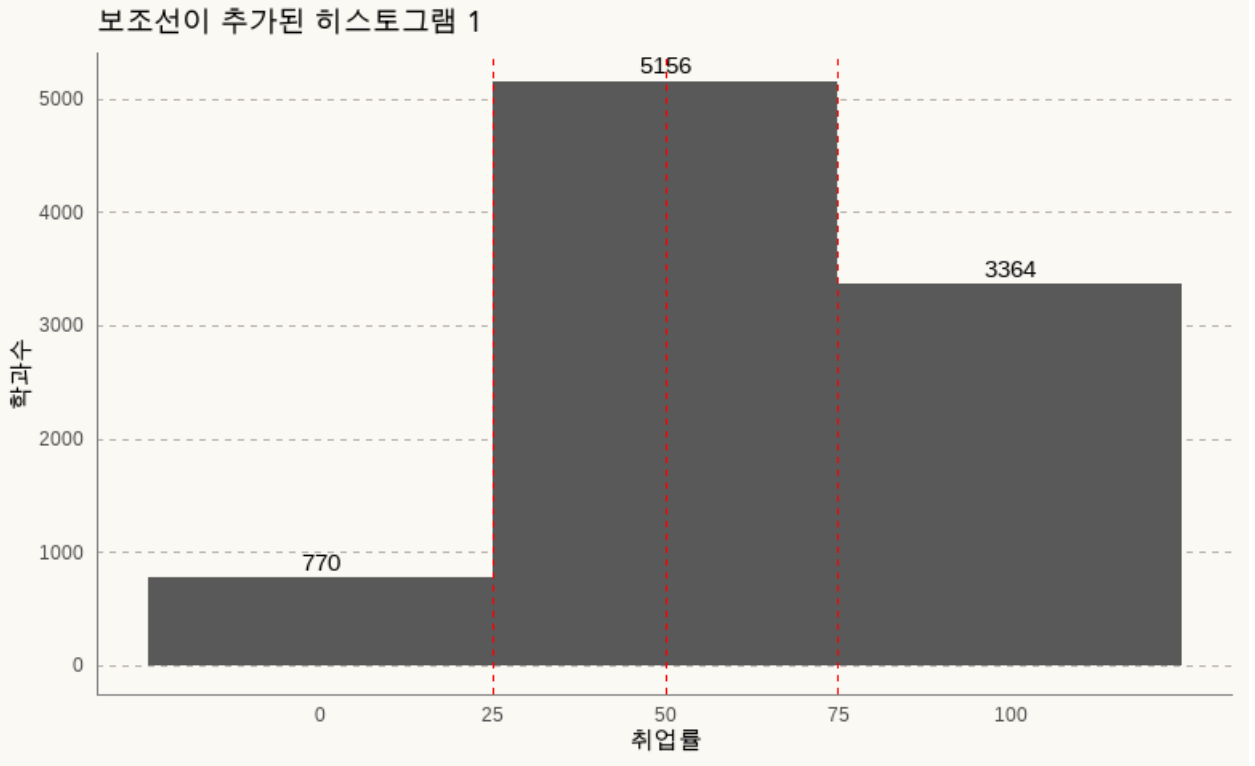

*실행결과 6-5. bin = 3 인 히스토그램*

앞의 실행결과를 보면 취업률의 범위인 0 부터 100 까지의 범위를 벗어나 막대가 표기된 것을 볼 수 있을 것이다. 그리고 값이 표현된 막대의 중간이 위치한 X 값이 0 과 100 임을 볼 수 있을것이다. 이를 좀 더 확실하게 확인해보면 다음과 같다.

다음의 코드는 X 축의 25%, 50%, 75%에 보조선을 그려보았다. 이 선을 보면 bin 이 구분된 곳이 어디인지 확인이 가능하다.

```
## 전체 X 축의 범위를 range.x 에 저장
range.x <- max(df_취업통계$취업률_계) - min(df_취업통계$취업률_계)

## bin 이 3 인 히스토그램에서 중 X 축의 중간에 geom_vline 레이어 추가
p_histo_bin +
  ## X 축의 중간위치에 geom_vline() 추가
```

```r
geom_vline(xintercept = range.x/2, color = 'red', linetype = 2) +
## X 축의 1/4 위치에 geom_vline() 추가
geom_vline(xintercept = range.x/2/2, color = 'red', linetype = 2) +
## X 축의 3/4 위치에 geom_vline() 추가
geom_vline(xintercept = range.x/2/2*3, color = 'red', linetype = 2) +
## 수직선 위치에 값 표시
scale_x_continuous(breaks = c(min(df_취업통계$취업률_계),
                              range.x/2/2,
                              range.x/2,
                              range.x/2/2*3,
                              max(df_취업통계$취업률_계)
                              )
) +
labs(title = '보조선이 추가된 히스토그램 1')
```

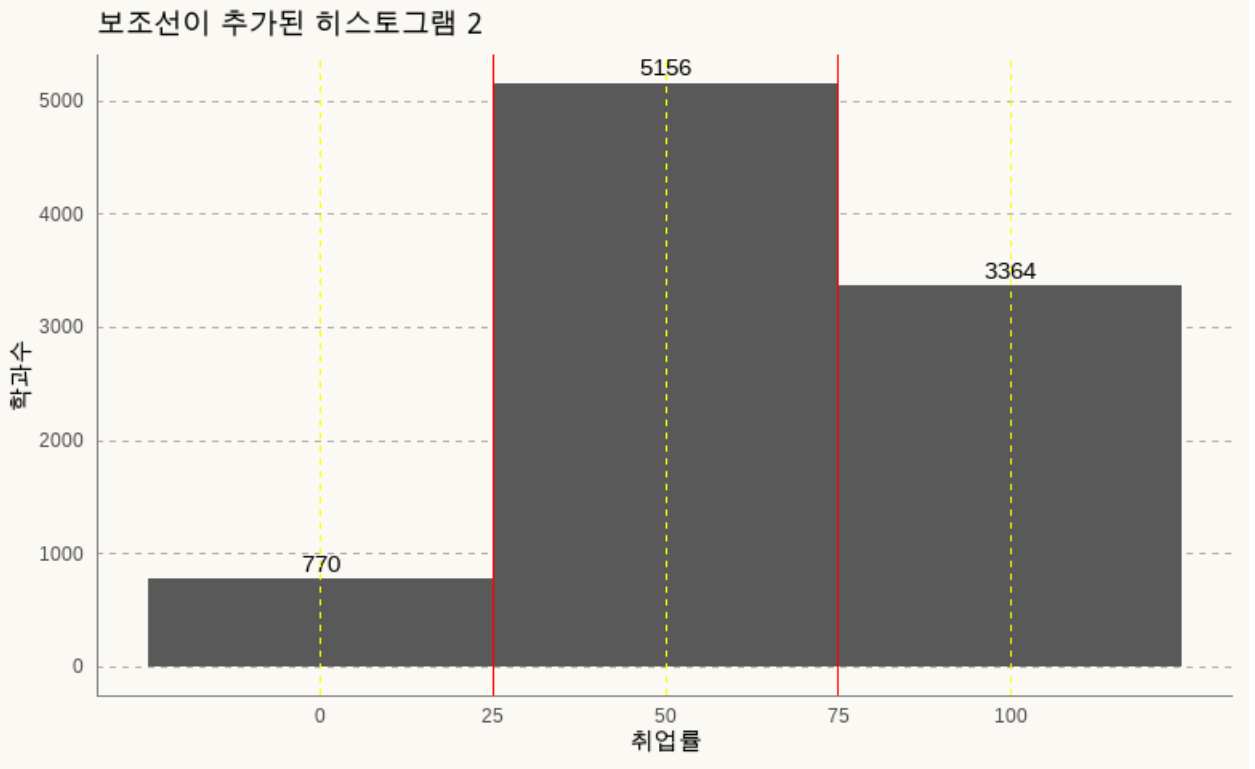



*실행결과 6-6. 보조선이 추가된 히스토그램 1*

이번에는 각각의 bin 의 중간점이 어디인지 확인해보자. 다음의 코드는 각 bin 의 중간점을 노란색 보조선으로 표현하였다. 이를 보면 처음 시각하는 bin 의 중간값을 최소값에서 시작한다는 것을 알 수 있다. 처음 bin 은 표현상으로는 -25 부터 25 까지 너비가 50 인 bin 으로 보이지만 최소값이 0 임을 감안하면 너비가 25(0 부터 25 까지)인 bin 이다. 반면 중간

bin 은 너비가 50 인 bin(25 부터 75 까지)이다. 그러다보니 처음과 마지막 bin 의 값의 범위는 중간 bin 의 범위와는 다르다는 것이다.

```
p_histo_bin +
  ## X 축이 50 인 곳에 수직선 레이어 추가
  geom_vline(xintercept = 50, color = 'yellow', linetype = 2) +
  ## X 축이 0 인 곳에 수직선 레이어 추가
  geom_vline(xintercept = 0, color = 'yellow', linetype = 2) +
  ## X 축이 100 인 곳에 수직선 레이어 추가
  geom_vline(xintercept = 100, color = 'yellow', linetype = 2) +
  ## X 축이 25 인 곳에 수직선 레이어 추가
  geom_vline(xintercept = 25, color = 'red') + ## X 축의 1/4 위치
  ## X 축이 75 인 곳에 수직선 레이어 추가
  geom_vline(xintercept = 75, color = 'red') + ## X 축의 3/4 위치
  ## X 축 스케일의 눈금 설정
  scale_x_continuous(breaks = c(min(df_취업통계$취업률_계),
                                range.x/2/2,
                                range.x/2,
                                range.x/2/2*3,
                                max(df_취업통계$취업률_계)
                                )
  ) +
  labs(title = '보조선이 추가된 히스토그램 2')
```




*실행결과 6-7. 보조선이 추가된 히스토그램 2*

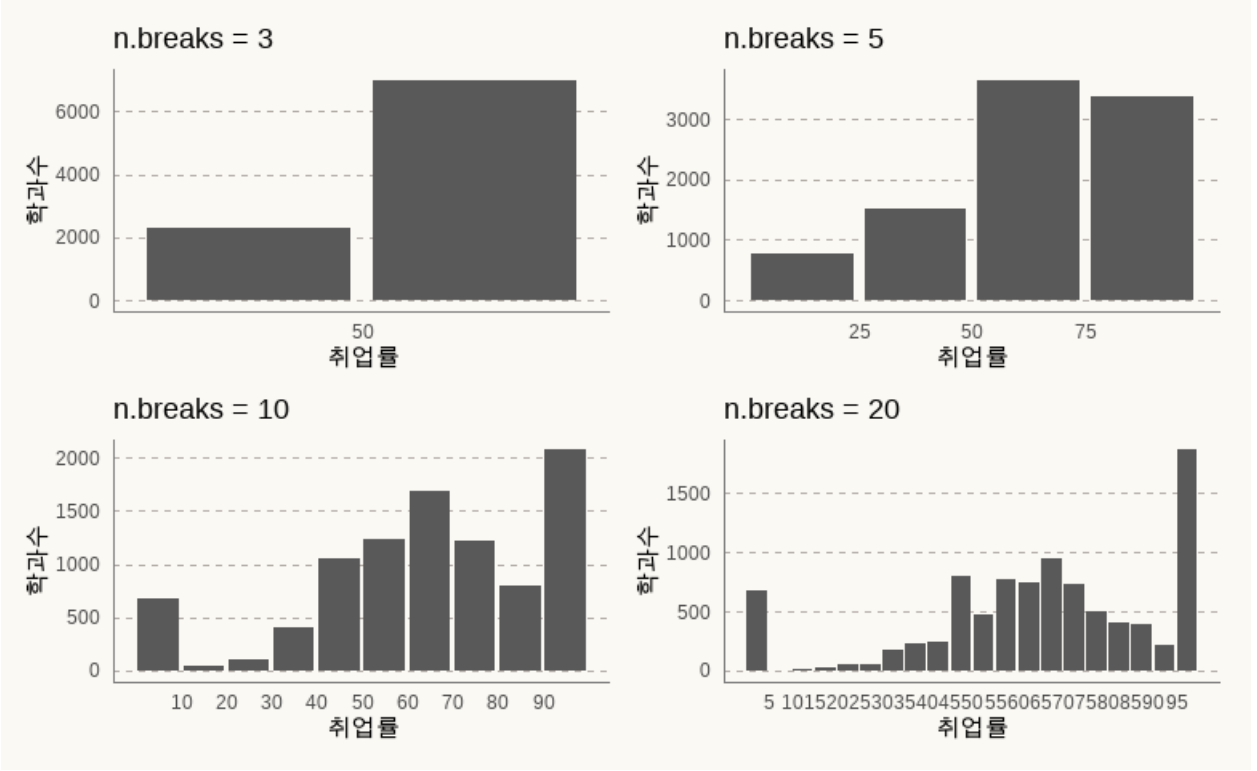
## 1.4. geom_bar 를 사용한 히스토그램

이처럼 `geom_histogram()`을 사용한 히스토그램은 모든 bin 의 너비가 일정하지 않다는 단점이 있고 X 축의 값 범위를 벗어난 막대는 시각화를 보는 사용자에게 친화적이지 않다. 또 히스토그램의 막대들이 붙어있다보니 보기에도 다소 자연스럽지 않다. 이러한 단점이 보완하기 위해서는 `geom_bar()`과 `scale_x_binned()`를 사용한 히스토그램을 사용할 수 있다.

`scale_x_binned()`는 연속형 데이터 스케일을 bin 으로 구분된 이산형 데이터 스케일로 변환하는 함수이다. `scale_x_binned()`에는 다른 `scale_*()`에서 사용하는 매개변수에는 없는 `n.break` 매개변수가 있다. 이 매개변수는 주어진 연속형 데이터 스케일을 `n.break` 의 값만큼 분할하여 bin 을 생성해 준다. 하지만 정확히 `n.break` 만큼 bin 이 생성되는 것은 아니고 가장 가까운 값만큼의 bin 이 생성된다. 또한 bin 간에 약간의 공백도 생기고 각각의 범위를 벗어나지 않기 때문에 보기에도 훨씬 좋아보인다.

```
p_histo +
  geom_bar(aes(x = 취업률_계)) +
  scale_x_binned(n.breaks = 3, right = T) +
  labs(title = 'n.breaks = 3')
```

```
p_histo +
  geom_bar(aes(x = 취업률_계)) +
  scale_x_binned(n.breaks = 5, right = T) +
  labs(title = 'n.breaks = 5')

p_histo +
  geom_bar(aes(x = 취업률_계)) +
  scale_x_binned(n.breaks = 10, right = T) +
  labs(title = 'n.breaks = 10')

p_histo +
  geom_bar(aes(x = 취업률_계)) +
  scale_x_binned(n.breaks = 20, right = T) +
  labs(title = 'n.breaks = 20')
```




*실행결과 6-8. geom_bar 를 사용한 히스토그램*

## 1.5. 히스토그램의 축 변환

히스토그램을 시각화할 때 만나는 문제들 중 많이 만나는 문제는 한쪽으로 치우친(Skewed)된 데이터이다. 이럴 경우에는 데이터를 효과적으로 표현하기 위해 축을 수학적 변환 공식에 따라 변형해 주는 방법이 많이 사용되는데 이에 대해 설명하도록 하겠다.

### 1.5.1. 치우친 데이터(Skewed Data)란?

치우친 데이터는 아래의 그림과 같이 데이터의 분포가 한쪽으로 몰려있는 경우를 의미한다. 아래의 그림처럼 데이터가 왼쪽이나 오른쪽으로 치우쳐 있고 반대쪽으로 꼬리가 길게 늘어뜨려진 데이터의 형태이다. 이런 치우친 데이터는 앞에서 언급한 바와 같이 주로 데이터의 사례수를 표현하는 히스토그램 시각화에서 많이 나타나게 된다. 다음 그림은 치우침이 비교적 심하지 않아 적절히 데이터의 분포를 확인할 수 있겠지만 데이터의 치우침이 큰 경우에는 히스토그램의 시각화가 큰 의미가 없을 때도 있다. 이와 같이 데이터의 치우침을 계산할 때 '왜도'라는 지수를 사용한다. R 에서는 왜도 함수를 통해 데이터의 왜도를 비교할 수 있다.

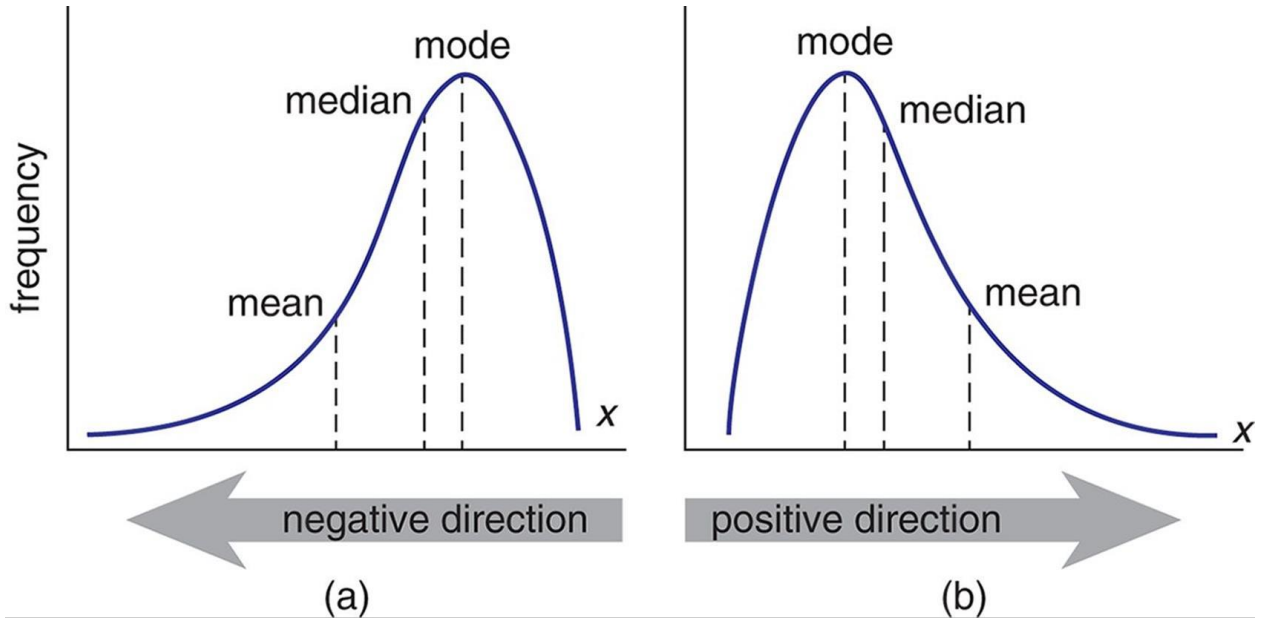
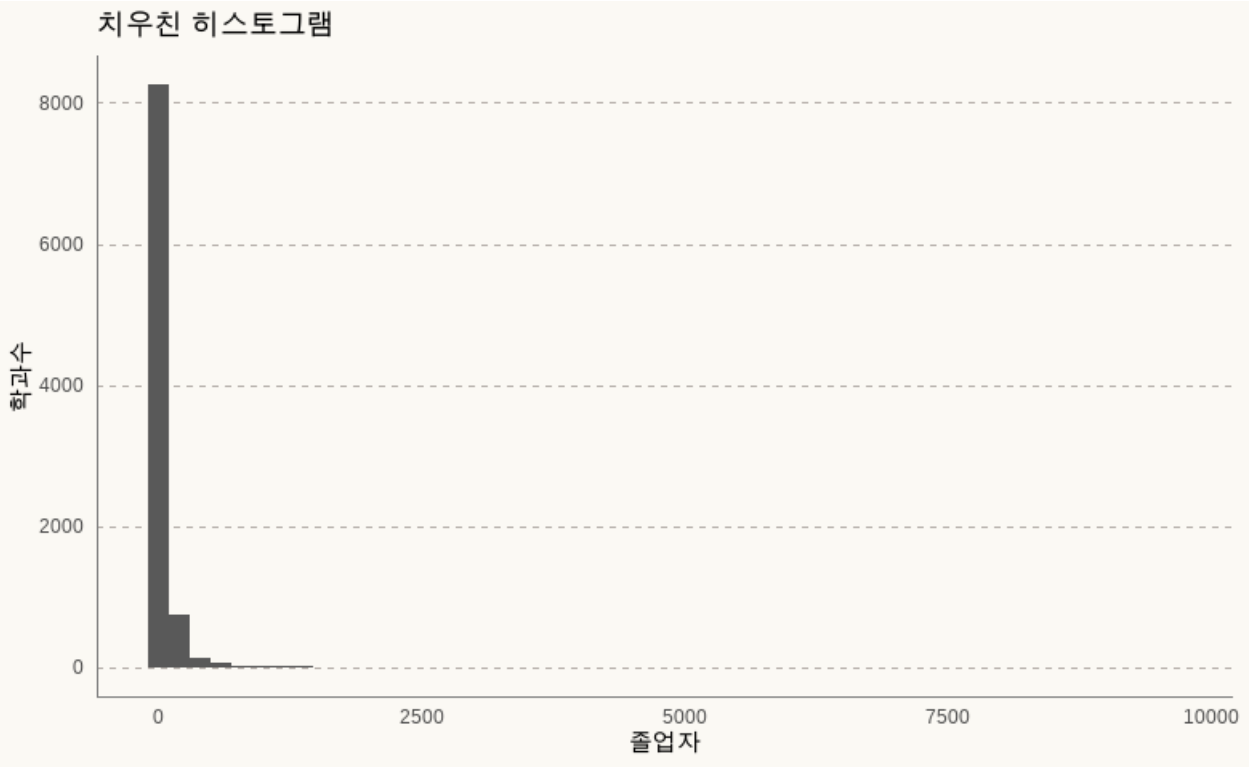



*그림 6-1 치우친 히스토그램의 예 (출처 : https://d1zx6djv3kb1v7.cloudfront.net/wp-content/media/2019/09/What-do-you-mean-by-the-terms-Skewed-Data-Outliers-Missing-Values-and-Null-Values-1-i2tutorials.jpg)*

다음의 코드는 df_취업통계_sample 데이터의 히스토그램이다. 오른쪽으로 매우 치우쳐진 히스토그램이 표현되었고 그 왜도가 매우 심해보인다.

```
p_histo_trans <- df_취업통계 |>
  ggplot() +
  labs(x = '졸업자', y = '학과수')

p_histo_trans1 <- p_histo_trans +
  ## X 축을 졸업자_계에 매핑하고 bins 가 50 인 geom_histogram 레이어 추가
  geom_histogram(aes(x = 졸업자_계), bins = 50) +
```

```
  labs(title = '치우친 히스토그램')

p_histo_trans1
```



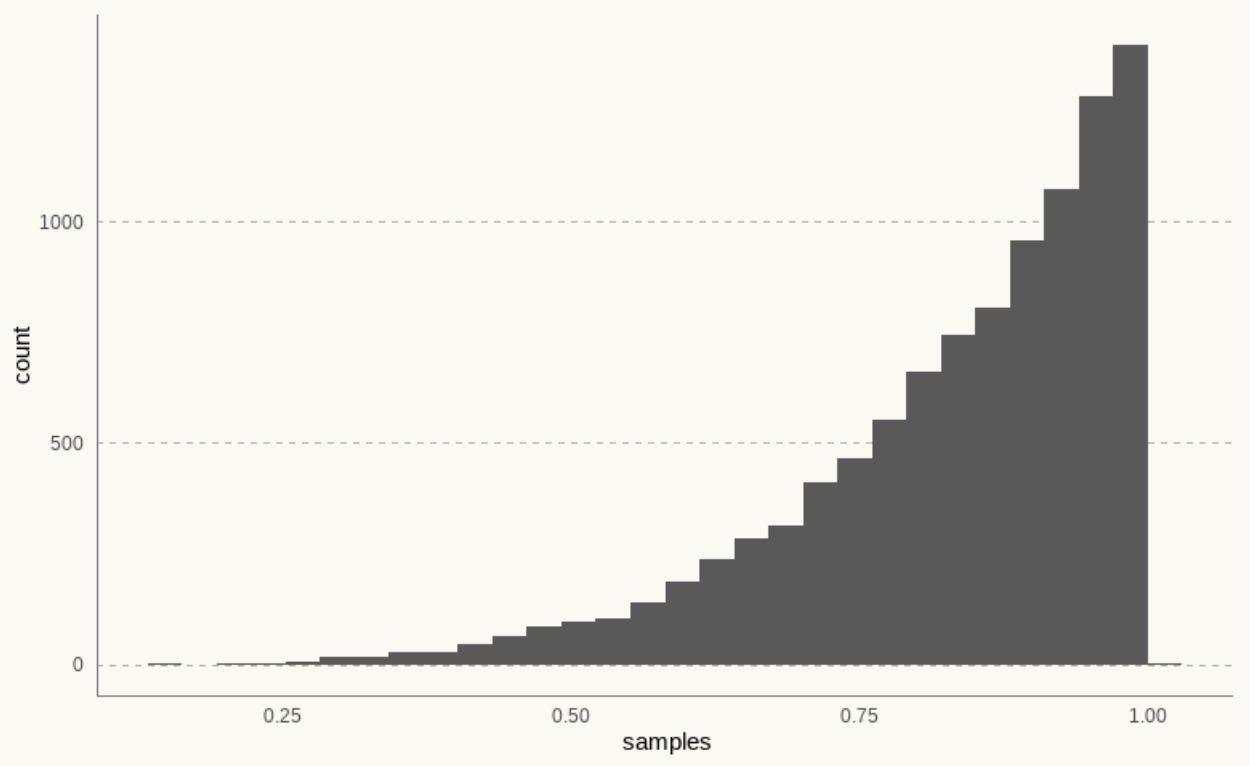

*실행결과 6-9. 치우친 히스토그램*

이 데이터에 대한 왜도는 다음과 같이 산출이 가능하다. 왜도의 정도를 `moments` 패키지의 `skewness()`를 사용하여 측정하면 다음과 같다.

```
## 왜도 측정을 위한 moments 패키지 설치
if(!require(moments)) {
  install.packages('moments')
  library(moments)
}

## df_취업통계$졸업자_계의 왜도를 측정
skewness(df_취업통계$졸업자_계)

## [1] 18.56238
```

왜도 값이 플러스 값이 나왔다. 이는 왼쪽의 봉우리가 높고 오른쪽으로 꼬리가 길게 치우쳐진 히스토그램 의미한다. 그리고 값은 18 이 넘게 나온다. 이 값은 매우 큰 값이다. 값의 정도를 비교해 보기 위해 다음의 예를 살펴보자.

```r
## 랜덤 변수를 위한 시드값 설정
set.seed(123)

##  베타 확률분포 함수를 이용하여 왼쪽으로 치우친(left skewed) 데이터 10000 개를 생성
samples <- rbeta(10000,5,1)

## 시뮬레이션을 위한 데이터 프레임을 생성
sim.data <- data.frame(samples)

## 왜도 산출
skewness(sim.data$samples)

## [1] -1.169109

## 시뮬레이션 데이터에 대한 히스토그램 생성
sim.data |> ggplot(aes(x = samples)) + geom_histogram()
```



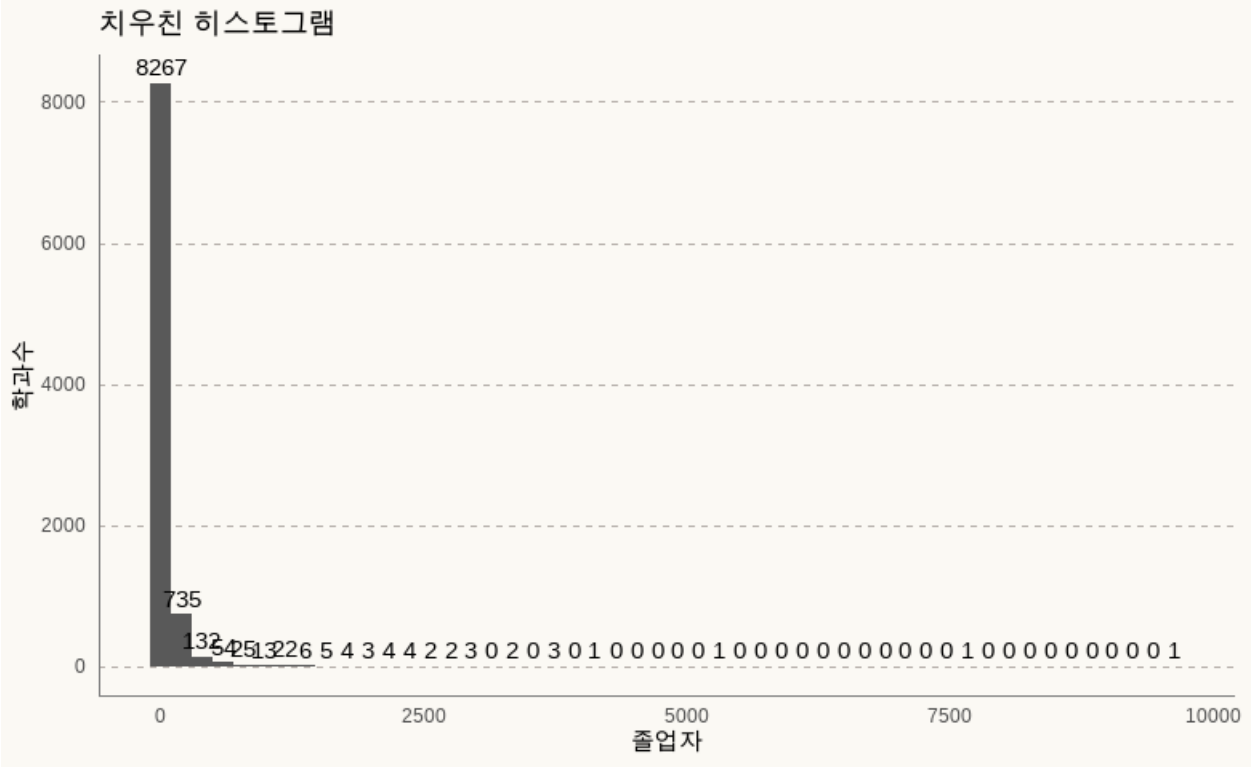

*실행결과 6-10. skew 시뮬레이션*

앞의 히스토그램은 왼쪽으로 크게 치우쳐진 시뮬레이션 데이터에 대한 히스토그램이다. 이 데이터에 대한 왜도값은 -1.16 이 산출되었다. 이에 비해 앞선 p_histo_trans1 에서 산출된 18 이 넘는 왜도 값은 꽤 큰 편이다.

### 1.5.2. 축 스케일 변환

앞서 보았던 p_histo_trans1 를 다시 한번 살펴보자. 히스토그램을 잘 이해하기 위해 아래와 같이 그래프를 수정해보자.

```
p_histo_trans1 +
  ## 히스토그램에 데이터 값이 표시되는 text 레이어 추가
  stat_bin(aes(x = 졸업자_계, y=..count.., label=..count..), geom="text", vjust=
-.5, bins = 50)
```




*실행결과 6-11. 데이터 값이 추가된 치우친 히스토그램*

오른쪽으로 치우쳐진 그래프이기 때문에 전체적인 분포를 알아볼 수 없다. 거의 모든 사례가 첫번째 막대에 분포되어 있기 때문에 히스토그램의 의미가 거의 없다. 이럴 경우 연속형 선형적 스케일을 변환하여 0 에 가까운 구간은 한 유닛(칸)의 간격을 좁게 주고 0 과 먼 구간은 한 유닛(칸)의 간격의 범위를 넓게 줄 수 있다. 이때 사용하는 변환이 log10 변환이다. log10 변환은 다음과 같이 코딩한다.

```
p_histo_trans1 +
  ## X 축 스케일을 log10 으로 변환
  scale_x_log10() +
  labs(title = 'log10 변환')
```

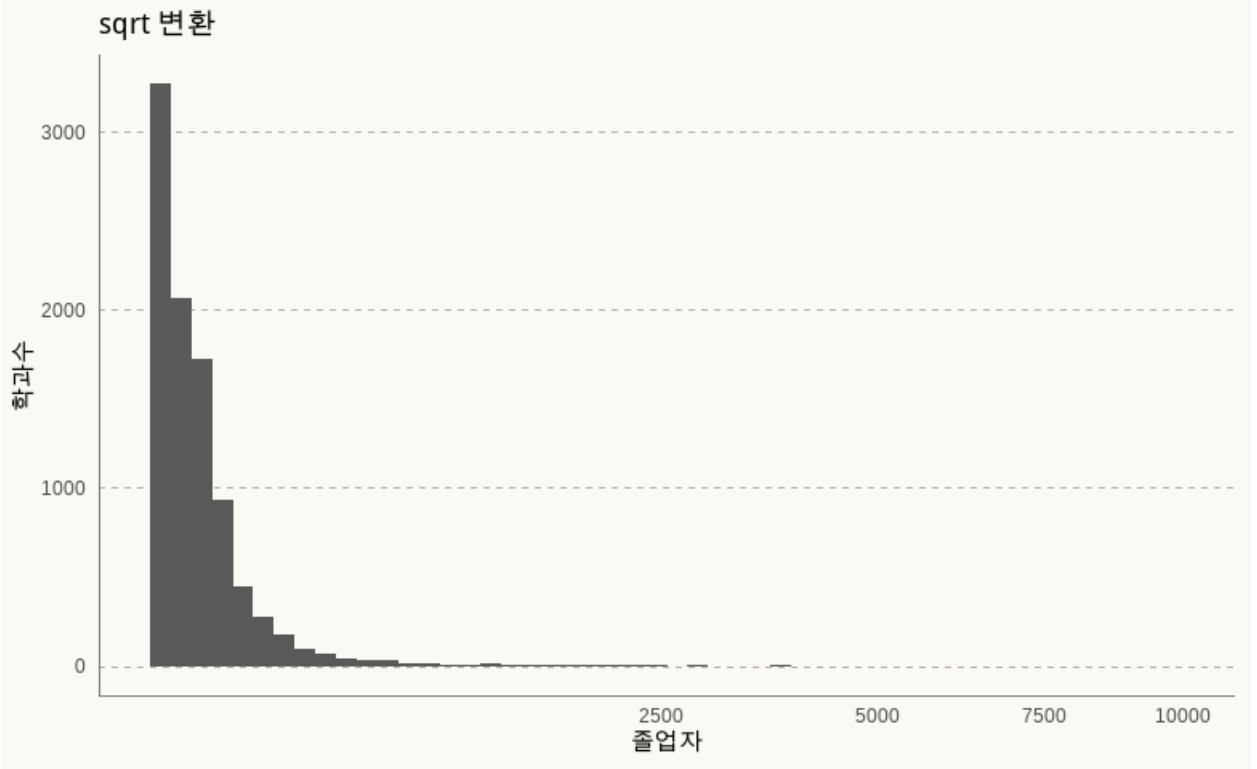
X 축의 스케일을 log10 변환하기 위해서는 `scale_x_log10()`을 사용한다. 반면 Y 축의 스케일을 log10 변환을 위해서는 `scale_y_log10()`을 사용하면 된다. 앞의 히스토그램과 같이 X 축 스케일을 log10 변환하면 히스토그램의 분포가 훨씬 의미있게 나타난다. 이 히스토그램의 X 축을 잘 살펴보면 몇가지 특징이 보일것이다.

첫번째는 X 축에 표현된 라벨이 모두 10 의 제곱수이다. $10^0$, $10^1$, $10^2$, $10^3$ 와 같다. 이 수치를 log10() 변환 결과는 0, 1, 2, 3 이다. 결국 10 진수의 표현은 1, 10, 100 이지만 log10 변환 결과는 1, 2, 3 이다.

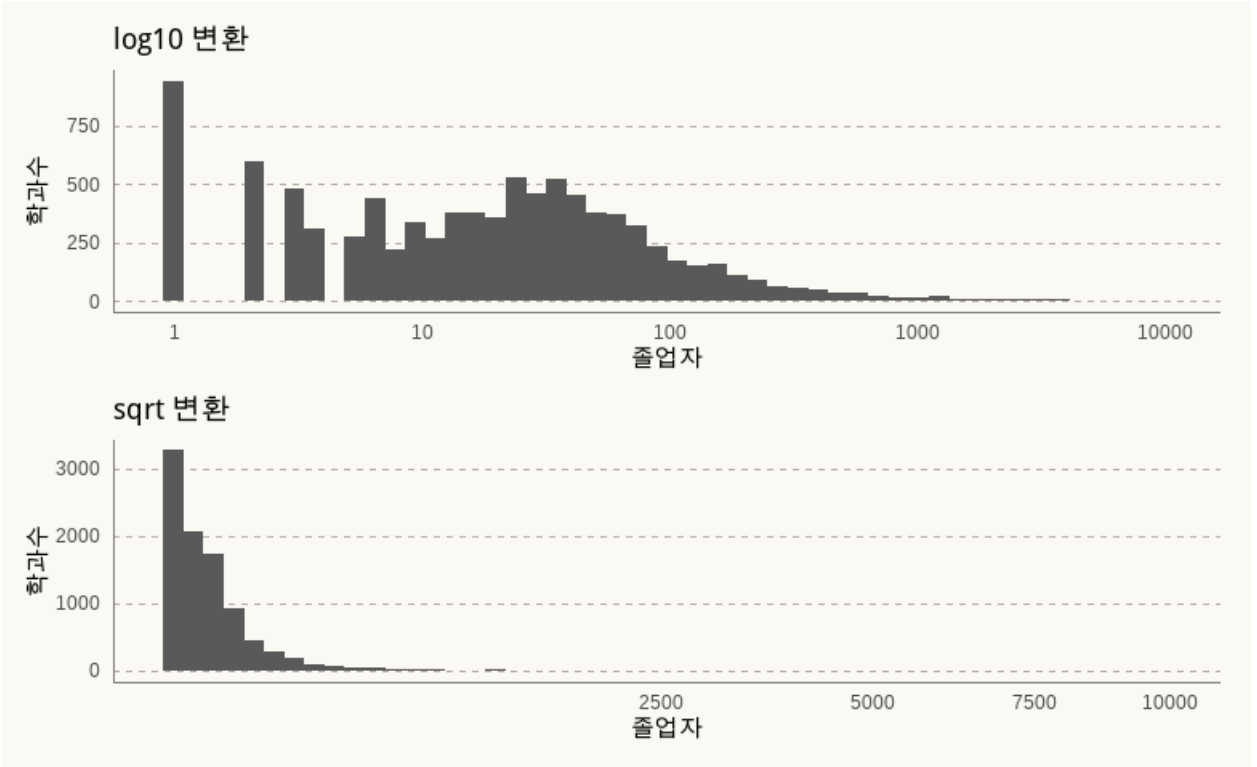
두번째는 각각의 칸의 범위가 다르다는 것이다. 1 다음 X 축의 2 칸은 10 이 표현된다. 그런데 그 다음 2 칸 이후는 100 이다. 앞의 2 칸의 범위는 10 이었지만 다음 2 칸의 범위는 90 이다. 그 다음 2 칸의 범위는 900 이다. 결국 log10 의 값이 커지면 커질수록 한칸에 표현되는 값이 범위가 넓어진다는 것이다.

반면 거꾸로 0 에 가까운 유닛의 범위를 넓게하고 0 과 면 유닛의 범위를 좁게하려면 `scale_x_sqrt()`를 사용한다.

```
p_histo_trans1 +
  ## X 축 스케일을 sqrt 로 변환
  scale_x_sqrt() +
  labs(title = 'sqrt 변환')
```




*실행결과 6-12. sqrt 변환*



## 1.6. 사용자 정의 히스토그램

지금까지 생성했던 히스토그램은 변수의 급간을 전체적으로 동일하게 분할하는 방식으로 나누었다. 하지만 실무에서 많이 사용하는 급간은 사회적으로 많이 통용되는 급간을 사용하거나 특정 급간 이하나 이상을 묶어주는 방식을 많이 사용한다. 예를 들어 연령대를 히스토그램으로 표현할 때 10 대, 20 대, 30 대와 같이 10 세 구분으로 설정한다. 또 보통 80 세 이상의 경우는 80 세, 90 세와 같이 세분화하는 대신 80 세 이상으로 묶어준다. 이와 같이 구간을 사용자가 직접 설정할 경우가 꽤 많이 생긴다. 이러한 경우 어떻게 히스토그램을 만들어 줄 것인지를 살펴보자.

사용자 정의 히스토그램을 만들기 위해서는 변수의 급간을 사용자가 원하는 대로 분할하는 것이 우선이다. 구간을 분할하는 방법은 여러가지가 있지만 여기서는 `cut()`과 `case_when()`을 사용하여 분할하는 방법을 소개한다.

### 1.6.1. cut 을 사용한 구간 분할

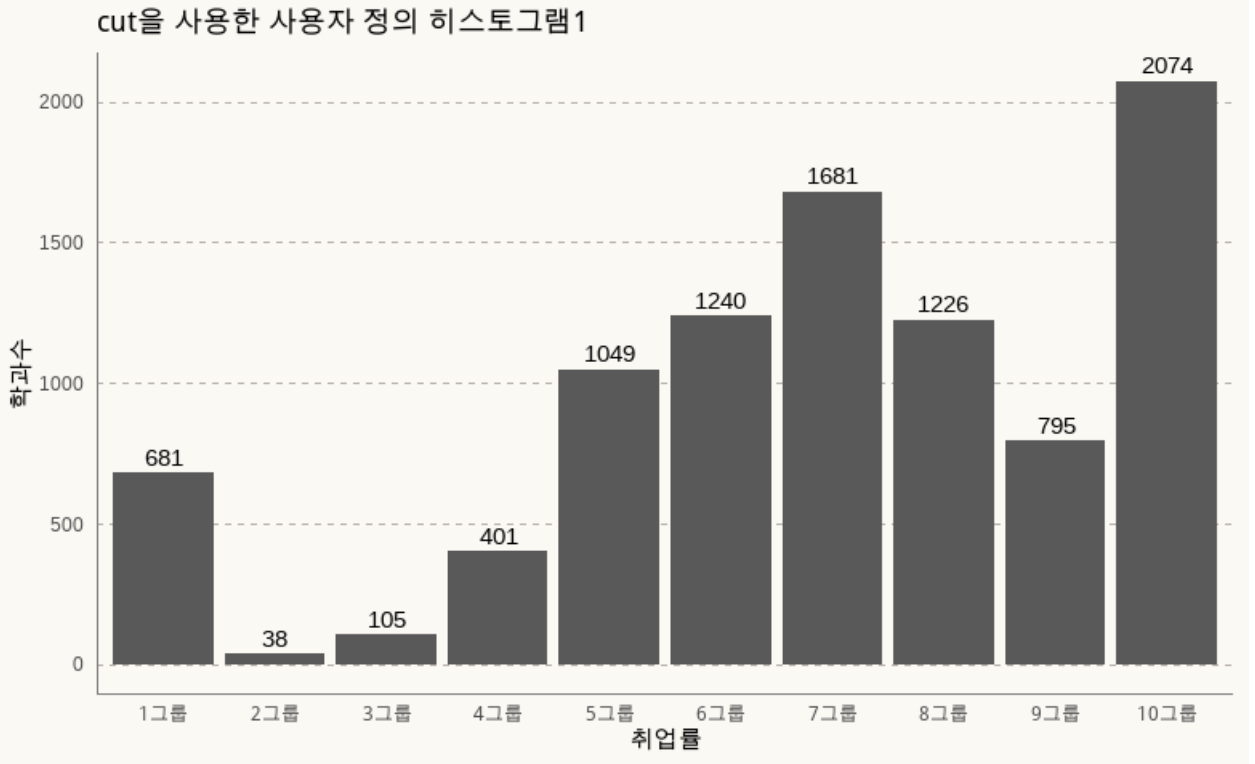
`cut()`은 매개변수롤 전달되는 연속된 숫자 벡터를 n 등분으로 나눈 구간을 설정하거나 구분점별로 나눈 구간을 설정할 때 사용하는 함수이다. 이 함수의 사용법과 주요 매개변수는 다음과 같다.

```
cut(x, breaks, labels = NULL, include.lowest = FALSE, right = TRUE, ...)
  - x : 구간을 설정하기 위해 사용하는 연속된 숫자 벡터
  - breaks : 구간을 어떻게 설정할지에 대한 정보를 전달. 단일 숫자가 전달되면 x 벡터의 최소와 최대값의 구간을 단일 숫자만큼의 구간으로 분할하고, c()를 사용하여 하나 이상의 숫자로 구성된 벡터가 전달 되면 각 숫자를 구분점으로 한 구간을 설정
  - labels : breaks 로 구분된 각 구간을 대표할 수 있는 라벨을 설정
  - right : 각 구간을 이하로 설정할지, 이상으로 설정할지를 결정.
  - include.lowest : x 의 최소값을 구간에 포함할지를 결정.
```

df_취업통계 데이터에서 취업률 구간의 길이가 균등하게 10 개의 구간으로 나누고 히스토그램을 코드는 다음과 같다

```
## cut()을 사용하여 df_취업통계의 취업률_계를 10 구간으로 나누고 각각의 이름을 붙여줌
df_취업통계$취업률_그룹 <- cut(df_취업통계$취업률_계, breaks = 10, label = paste0(1:10, '그룹'))
```

```r
df_취업통계 |>
  ggplot() +
  ## cut()으로 구분된 취업률 그룹을 X 축으로 매핑하고 Y 축에 사례수를 매핑한 geom_bar 레이어 생성
  geom_bar(aes(x = 취업률_그룹, y = ..count..)) +
  ## 사례수를 표시하는 text 레이어 추가
  stat_count(aes(x = 취업률_그룹, y=..count.., label=..count..), geom="text", vjust=-.5) +
  labs(title = 'cut 을 사용한 사용자 정의 히스토그램 1', x = '취업률', y = '학과수')
```

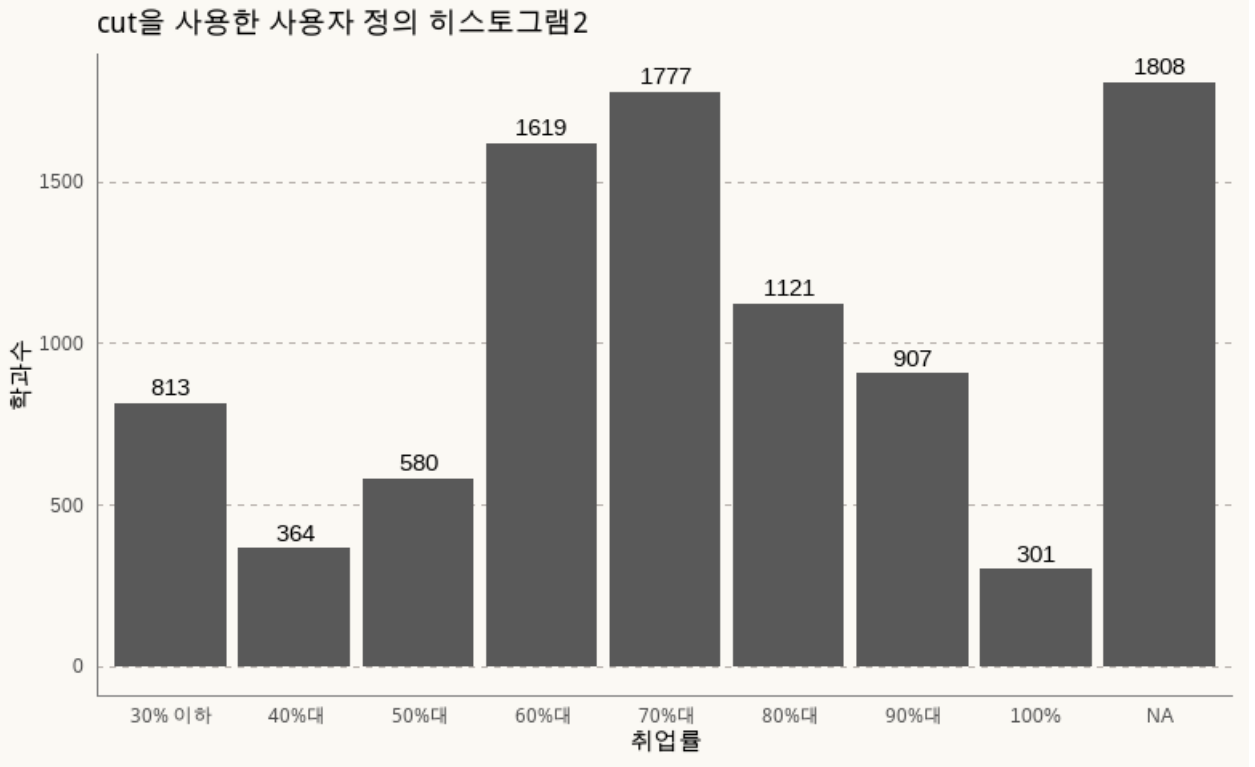



*실행결과 6-13. cut 을 사용한 사용자 정의 히스토그램 1*

이번에는 취업률 30%이하는 묶어 '30% 이하'로 표현되는 분할을 만들어보겠다.

```r
## cut()을 사용하여 df_취업통계의 취업률_계를 나눌 구간을 지정하고 각각의 이름을 붙여줌
df_취업통계$취업률_그룹 <- cut(df_취업통계$취업률_계, breaks = c(-Inf, 30, 40, 50, 60, 70, 80, 90, 100), label = c('30% 이하', '40%대', '50%대', '60%대', '70%대', '80%대', '90%대', '100%'), right = FALSE)

df_취업통계 |>
```

```
  ggplot() +
  ## cut()으로 구분된 취업률 그룹을 X 축으로 매핑하고 Y 축에 사례수를 매핑한 geom_bar 레이어 생성
  geom_bar(aes(x = 취업률_그룹, y = ..count..)) +
  ## 사례수를 표시하는 text 레이어 추가
  stat_count(aes(x = 취업률_그룹, y=..count.., label=..count..), geom="text", vjust=-.5) +
  labs(title = 'cut 을 사용한 사용자 정의 히스토그램 2', x = '취업률', y = '학과수')
```




*실행결과 6-14. cut 을 사용한 사용자 정의 히스토그램 2*

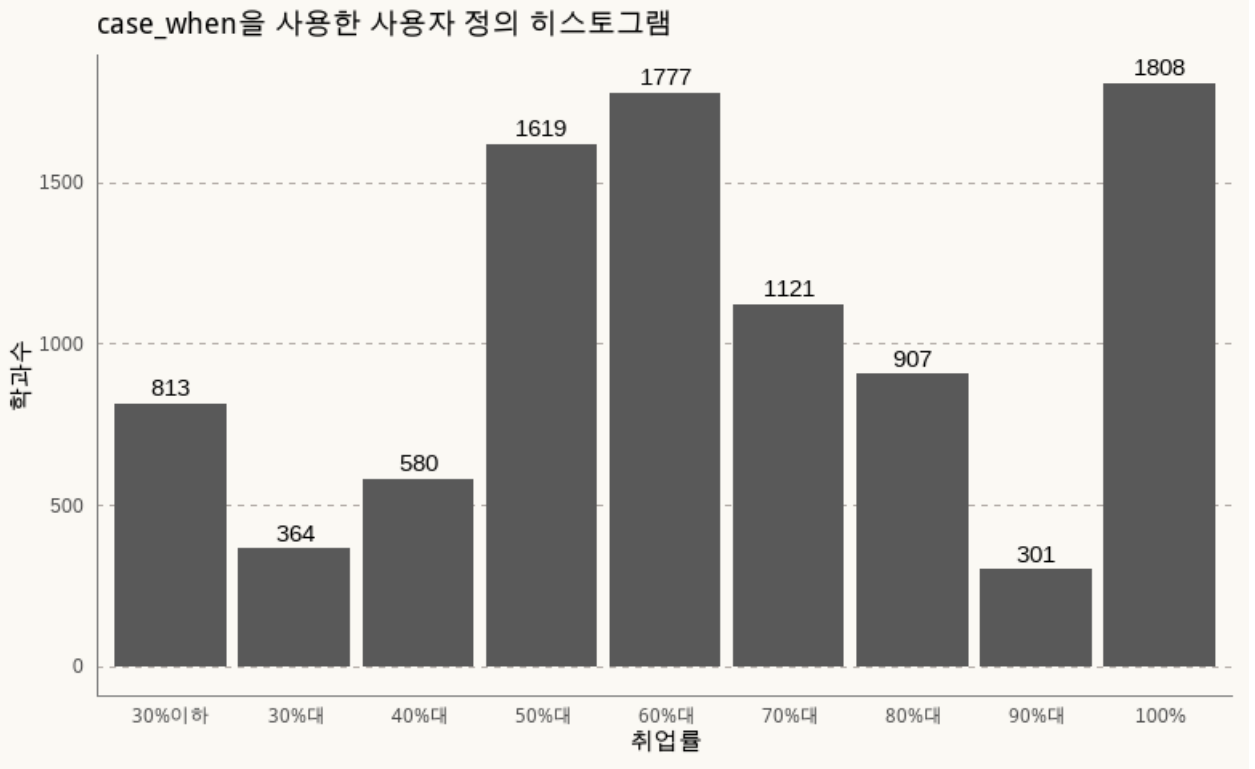
### 1.6.2. case_when 을 사용한 구간 분할

구간을 분할하는 방식으로 `case_when()`을 사용할 수 있다. 이 함수는 `tidyverse` 패키지에 포함된 `dplyr` 패키지에서 제공하는 함수이므로 다음과 같이 `mutate()`와 함께 사용해주는 것이 가장 효과적이다.

`case_when()`은 ~로 표현되는 연속된 양 사이드 공식(Two side formulas)으로 표현된다. ~의 왼쪽편(LHS, Left Hand Side)에는 구간을 나눌 조건식이 오고 ~의 오른쪽편(RHS, Right Hand

Side)에는 대체할 값이 온다. 모든 조건에 `FALSE`를 얻은 데이터는 마지막에 도달하면 `TRUE`를 사용하여 최종 구간을 설정하도록 할당된다.

```
## case_when()을 사용하여 각각의 그룹으로 분할
df_취업통계 <- df_취업통계 |>
  mutate(취업률_그룹 = case_when(
    취업률_계 < 30 ~ '30%이하',
    취업률_계 >= 30 & 취업률_계 < 40 ~ '30%대',
    취업률_계 >= 40 & 취업률_계 < 50 ~ '40%대',
    취업률_계 >= 50 & 취업률_계 < 60 ~ '50%대',
    취업률_계 >= 60 & 취업률_계 < 70 ~ '60%대',
    취업률_계 >= 70 & 취업률_계 < 80 ~ '70%대',
    취업률_계 >= 80 & 취업률_계 < 90 ~ '80%대',
    취업률_계 >= 90 & 취업률_계 < 100 ~ '90%대',
    TRUE ~ '100%'
  ))

## 그룹순서를 설정
df_취업통계$취업률_그룹 <-  fct_relevel(df_취업통계$취업률_그룹, '30%이하', '30%대',
 '40%대', '50%대', '60%대', '70%대', '80%대', '90%대', '100%')

df_취업통계 |>
  ggplot() +
  ## case_when()으로 구분된 취업률 그룹을 X 축으로 매핑하고 Y 축에 사례수를 매핑한 geo
m_bar 레이어 생성
  geom_bar(aes(x = 취업률_그룹, y = ..count..)) +
  ## 사례수를 표시하는 text 레이어 추가
  stat_count(aes(x = 취업률_그룹, y=..count.., label=..count..), geom="text", vju
st=-.5) +
  labs(title = 'case_when 을 사용한 사용자 정의 히스토그램', x = '취업률', y = '학과
수')
```




*실행결과 6-15. case_when 을 사용한 사용자 정의 히스토그램*

## 2. 밀도 분포 플롯

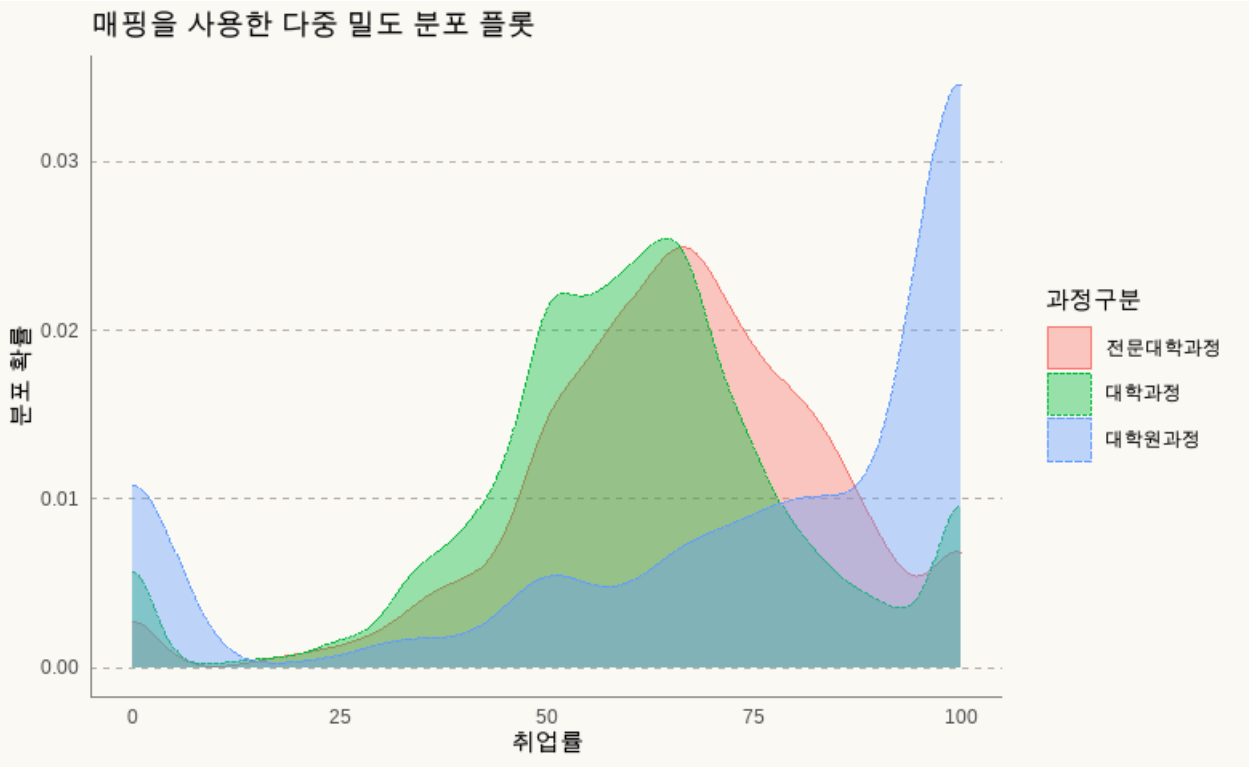
밀도 분포 플롯는 히스토그램의 또 다른 표현 방법이다. 히스토그램은 막대로 도수분포가 표현되기 때문에 다소 딱딱한 감이 있지만 밀도 분포 플롯은 전체 분포가 곡선으로 표현되어 히스토그램보다 부드럽게 표현된다는 장점이 있다. 또 도수분포에 대한 사례수를 직접적으로 사용하는 것이 아니고 확률 밀도 함수에 따른 확률값을 사용하기 때문에 0 부터 1 사이의 값으로 표현된다는 것이 히스토그램과의 차이다.

`ggplot2` 에서는 밀도 분포 그래프를 그리기 위해 `geom_density()`를 제공한다. `geom_density()`의 기초 사용법은 3 장에서 살펴보았기 때문에 밀도 분포 플롯의 다양한 표현 방법에 대해 알아보겠다.

### 2.1. 다중 밀도 분포 플롯

다중 밀도 분포 플롯는 변수 변량에 따라 여러개의 밀도 분포 플롯을 그리는 방법을 말한다. 다중 밀도 분포 플롯은 다중화할 변수를 color 나 fill 에 매핑함으로써 생성할 수 있다.

```r
p_density <- df_취업통계 |>
  ggplot() +
  labs(x = '취업률', y = '분포 확률')

p_density_multi1 <- p_density +
  ## X 축을 취업률_계, color, fill, linetype 을 과정구분으로 매핑하고 미적요소를 설정한 geom_density 레이어 추가
  geom_density(aes(x = 취업률_계, color = 과정구분, fill = 과정구분, linetype = 과정구분), alpha=0.4, position = 'identity') +
  labs(title = '매핑을 사용한 다중 밀도 분포 플롯')

p_density_multi1
```

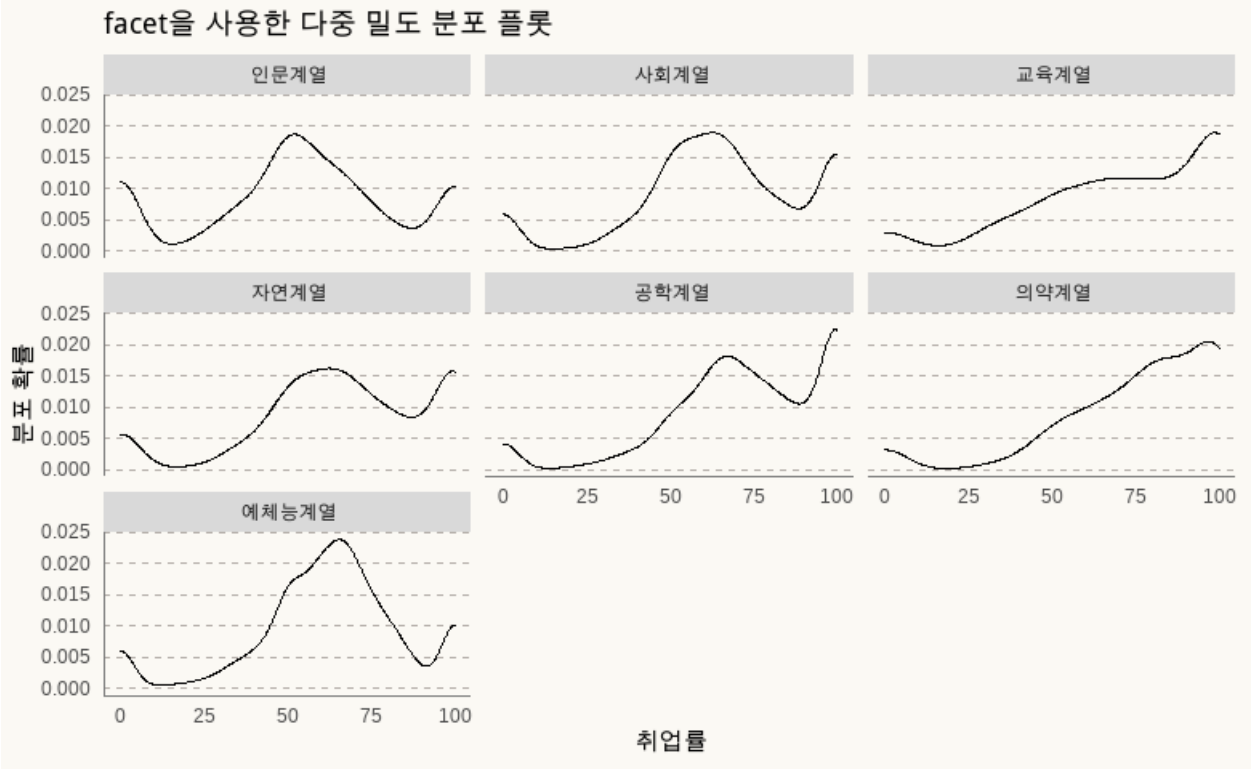



*실행결과 6-16. 매핑을 사용한 다중 밀도 분포 플롯*

앞의 다중 밀도 플롯은 color 와 fill 로 매핑하여 생성하였지만 `facet_wrap()`을 사용하여 각각의 밀도 플롯를 분리할 수도 있다.

```r
p_density +
  ## X 축을 취업률_계로 매핑하고 미적요소를 설정한 geom_density 레이어 추가
  geom_density(aes(x = 취업률_계), alpha=0.4, position = 'identity') +
  ## 대계열별로 다중 밀도 플롯 생성
```

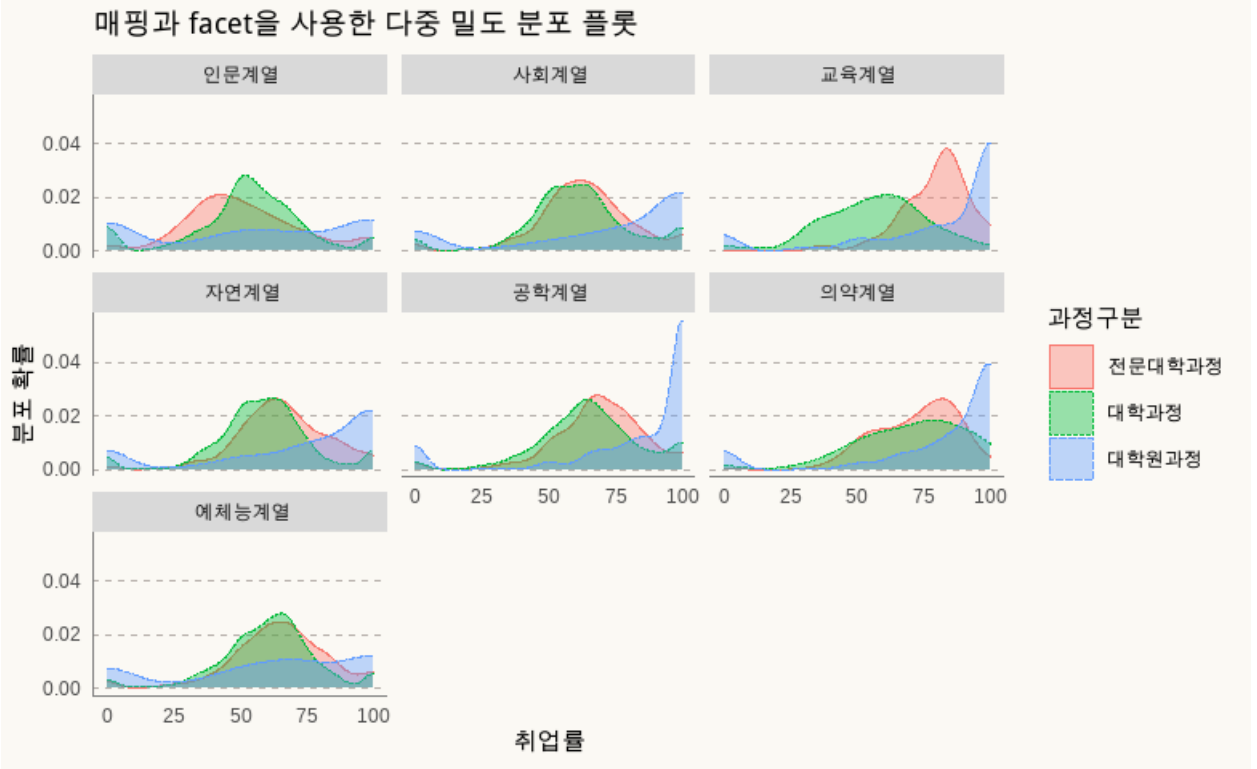
```
  facet_wrap(~대계열) +
  labs(title = 'facet 을 사용한 다중 밀도 분포 플롯')
```




*실행결과 6-17. facet 을 사용한 다중 밀도 분포 플롯*

앞의 두가지 다중 밀도 확률 플롯를 모두 사용하면 다음과 같다.

```
p_density +
  ## X 축을 취업률_계, color, fill, linetype 을 과정구분으로 매핑하고 미적요소를 설정한 geom_density 레이어 추가
  geom_density(aes(x = 취업률_계, color = 과정구분, fill = 과정구분, linetype = 과정구분), alpha=0.4, position = 'identity') +
  ## 대계열별로 다중 밀도 플롯 생성
  facet_wrap(~대계열) +
  labs(title = '매핑과 facet 을 사용한 다중 밀도 분포 플롯')
```




*실행결과 6-18. 매핑과 facet 을 사용한 다중 밀도 분포 플롯*

## 2.2. 등선(ridge) 플롯

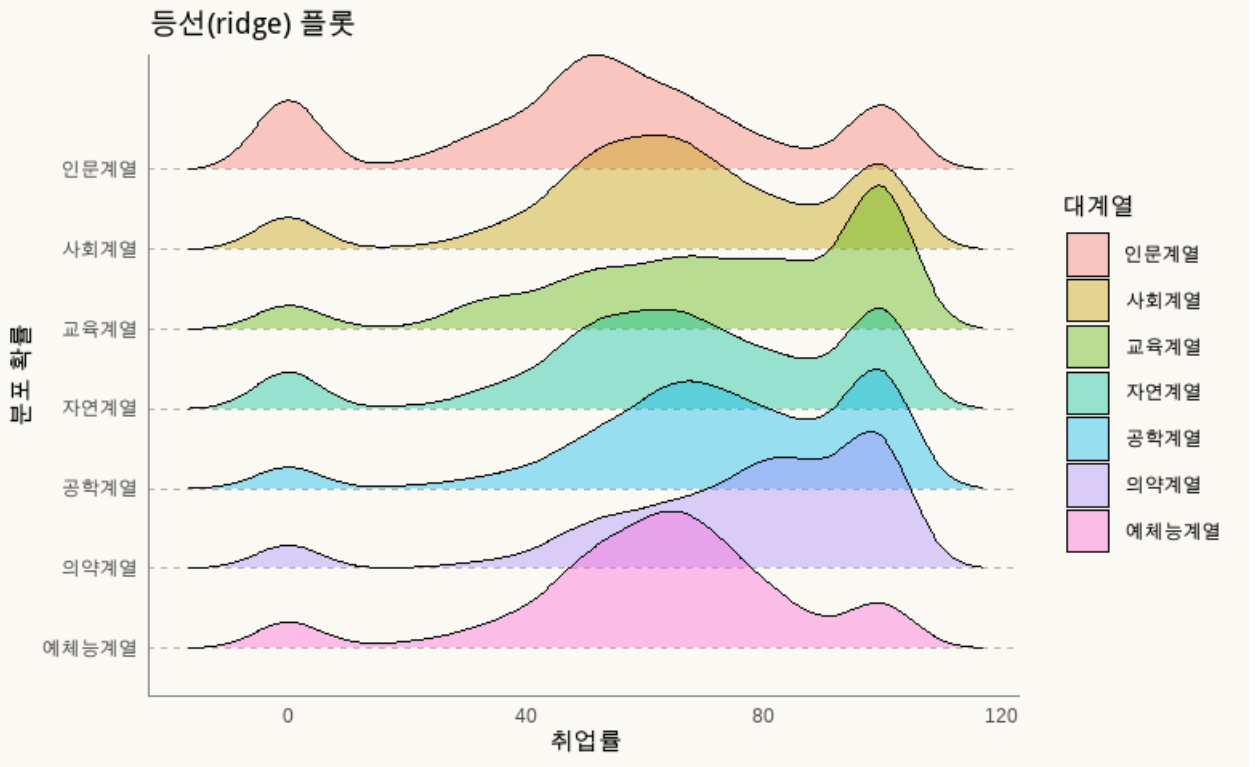
등선(ridge) 플롯(Joyplot 이라고도 함)은 여러 그룹에 대한 데이터의 분포를 밀도 함수로 보여주는 플롯이다. 등선 플롯은 모두 동일한 수평 스케일에 정렬되어 각각의 밀도 분포 플롯들이 겹쳐서 표현되기도 한다. 이와 같은 겹침은 `facet_*()`을 이용한 분할보다는 공간을 절약할 수 있다는 장점이 있다. 따라서 분할해야할 그룹이 많은 경우 등선 플롯이 효과적으로 사용될 수 있다. 보통 6 개 이상의 분할이 발생하면 등선 플롯이 더 효과적이라고 한다.[1]

등선 플롯은 `ggridges` 패키지에서 제공하는 `geom_density_ridge()`를 사용하면 생성할 수 있다.

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

[1] https://www.data-to-viz.com/graph/ridgeline.html

```
p_density +
## X 축을 취업률_계, Y 축, fill 을 대계열로 매핑한 geom_density_ridge 레이어 생성
  geom_density_ridges(aes(x = 취업률_계, y = 대계열, fill = 대계열), alpha=0.4, po
sition = 'identity') +
  ## 범례 순서와 맞추기 위해 Y 축 순서를 반전
  scale_y_discrete(limits = rev) +
  labs(title = '등선(ridge) 플롯')
```




*실행결과 6-19. 등선(ridge) 플롯*

## 2.3. 가장자리(marginal) 플롯

이번 장에서 설명하고 있는 분포의 시각화는 앞 장에서 설명한 상관의 시각화와 상당히 유사한 부분이 있다. 관계의 시각화는 데이터를 통계처리하지 않고 그 자체를 시각화하는 반면 분포의 시각화는 전체 데이터를 통계처리한 결과를 시각화한다는 차이가 있다. 따라서 이 두 시각화들은 서로 상호보완적으로 사용이 가능하다. 이렇게 분포의 시각화와 상관의 시각화를 상호보완적으로 보여주는 시각화 방법이 가장자리(Marginal) 플롯이다.

가장자리 플롯은 앞 장에서 설명한 관계의 시각화를 기본으로 한다. 이 시각화의 X 축과 Y 축의 가장자리에 히스토그램, 밀도 분포 플롯, 다음 절에서 설명할 박스플롯 등을 표현함으로서 전체 분포를 상관과 같이 보여준다.

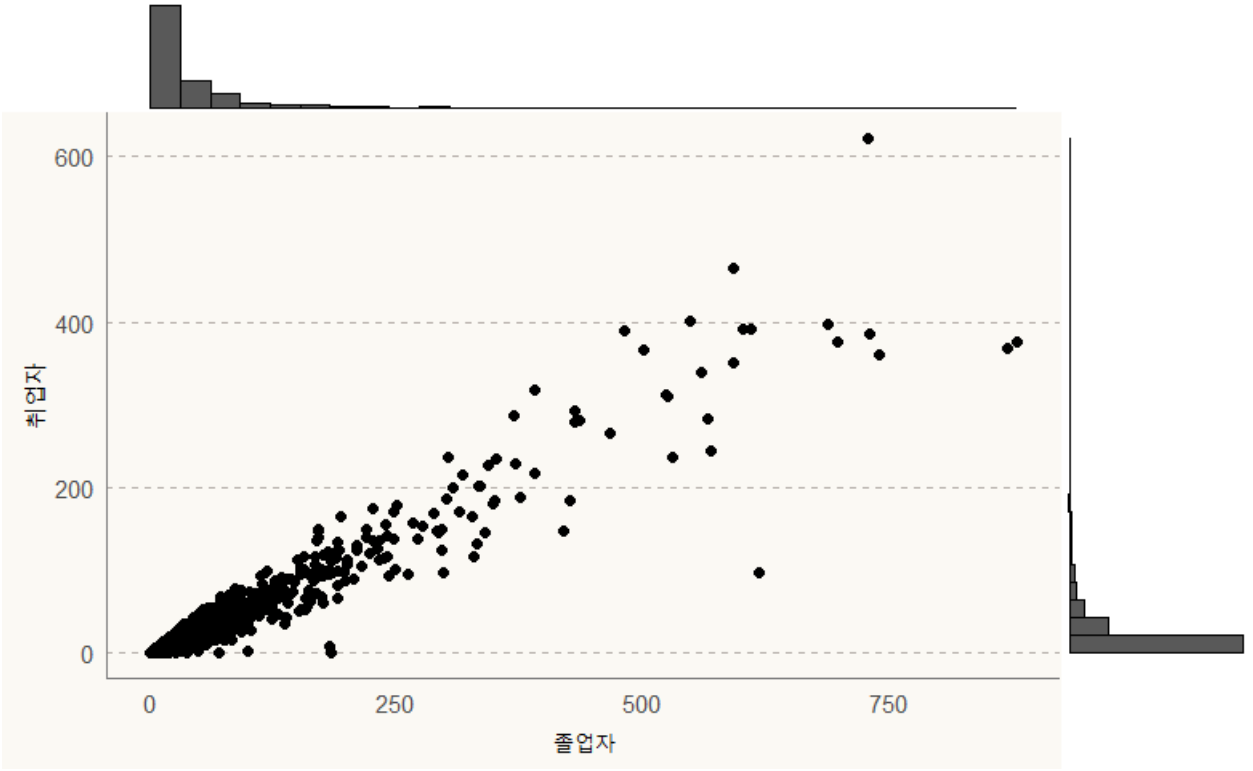
이 가장자리 플롯은 `ggplot2` 에서 직접적으로 사용하지는 못하고 `ggExtra` 패키지를 통해 사용할 수 있다. `ggExtra` 패키지는 `ggplot2` 의 기능적 향상을 위해 개발된 패키지로 이 패키지의 가장 주된 기능은 `ggMarginal` 이다. `ggMarginal` 은 `ggplot2` 산점도의 X 축과 Y 축의 가장자리에 히스토그램, 박스플롯, 밀도 플롯을 추가시키는 함수이다.

```
ggMarginal(p, data, x, y, type = c("density", "histogram", "boxplot", "violin",
"densigram"), margins = c("both", "x", "y"), xparams = list(), yparams = list(),
 groupColour = FALSE, groupFill = FALSE, size = 5)
  - p : 가장자리 플롯을 추가할 산점도 ggplot 객체
  - data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
  - x : X 축에 매핑된 변수명
  - y : Y 축에 매핑된 변수명
  - type : 가장자리에 추가될 플롯 설정
  - margins : 가장자리 플롯이 추가될 위치 설정
  - xparams : X 축에 표현되는 플롯에 추가적으로 사용될 매개변수
  - yparams : Y 축에 표현되는 플롯에 추가적으로 사용될 매개변수
  - groupColor : 산점도에 사용된 그룹 색을 사용할지 설정하는 논리값
  - groupFll : 산점도에 사용된 그룹 채움 색을 사용할지 설정하는 논리값
  - size : 가장자리 플롯의 크기 설정
```

다음의 코드는 앞서 생성했던 df_취업통계_sample 의 산점도에 히스토그램을 가장자리에 추가한 플롯을 생성한다.

```r
### ggExtra 패키지 설치
if(!require('ggExtra')) {
  install.packages('ggExtra')
  library(ggExtra)
}

## 산점도 생성
p_marginal <- df_취업통계_sample |>
  ggplot() +
  geom_point(aes(x = 졸업자_계, y = 취업자_합계_계)) +
  labs(x = '졸업자', y = '취업자')
```

```
##  가장자리 플롯으로 히스토그램을 추가
ggMarginal(p_marginal, x = 졸업자_계, y = 취업자_합계_계, type = 'histogram')
```




*실행결과 6-20. 타입별 가장자리 플롯*

## 3. 박스 플롯과 바이올린 플롯

박스플롯은 연속된 수치형 데이터의 전체 분포를 완벽하게 보여줄 수 있는 플롯이다. 이 박스플롯은 중앙값, 데이터의 25%, 75%의 위치, IQR 의 1.5 배, -1.5 배 위치의 다섯가지의 요약통계를 사용하여 데이터의 전체 분포를 시각화한다. 이러한 박스를 사용하여 그룹된 이산형 변수들의 분포를 비교할 수 있다.

박스플롯은 변수 변량 그룹간의 데이터의 분포와 여러 통계값을 동시에 비교할 수 있다는 점에 매우 유용하지만 주의해야할 점도 있다. 박스플롯에는 데이터가 요약되기 때문에 이 요약으로 인해 잃게되는 정보도 있다는 점이다. 가장 큰 점은 데이터의 사례수가 표현되지 않는다는 점이다. 박스 플롯상으로 중간값, 25%, 75% 값이 다른 그룹보다 높게 있더라도 그 그룹에 포함된 사례가 다른 그룹보다 현저하게 적다면 이 부분을 다시 한번 검증해봐야하지만

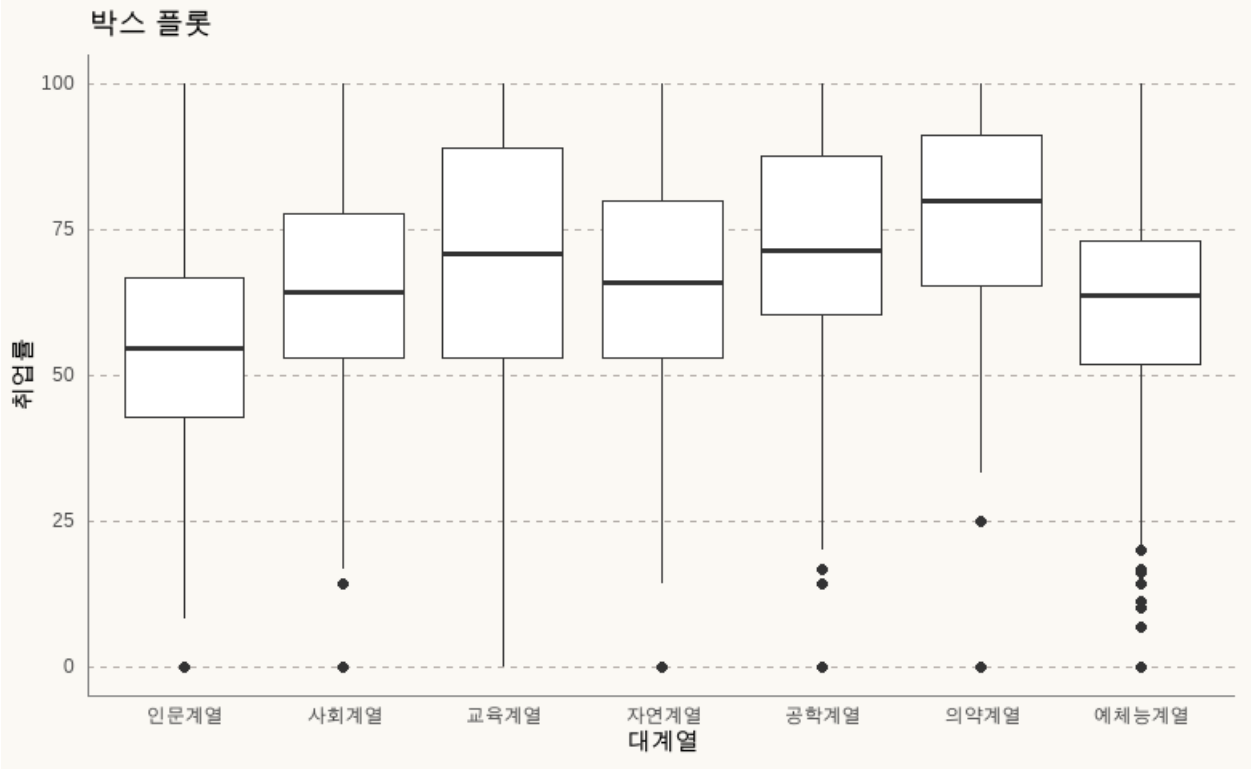
박스플롯만으로는 이 부분을 알아내기가 어렵다. 이러한 단점을 보완하는 방법으로 사용할 수 있는 것이 박스플롯에 산점도를 병합하는 방법과 바이올린 플롯이다.

박스플롯에 산점도를 병합하는 방식은 여전히 산점도가 가지고 있는 오버플로팅 문제가 존재한다. 이를 피하기 위해 설명한 여러 방법(샘플수 조정, 투명도 조정 등)을 사용할 수 있지만 가장 많이 사용되는 방법은 데이터를 좌우로 흩뿌려 주는(Jitter) 방법이다. 하지만 이 방법도 사례수가 많아지면 여전히 오버플로팅의 문제가 발생한다. 바이올린 플롯은 박스플롯에 산점도를 병합한 방식의 단점을 보완할 수 있다. 따라서 바이올린 플롯은 데이터의 양이 많아져서 산점도를 겹쳐서 보여주기 어려울 경우 매우 효과적인 플롯이다.

## 3.1. 기본 플롯

앞서 설명한 바와 같이 박스플롯과 바이올린 플롯의 기초 컨셉은 동일하다. 따라서 다음과 같이 기본 플롯을 만들어 사례를 보이겠다.

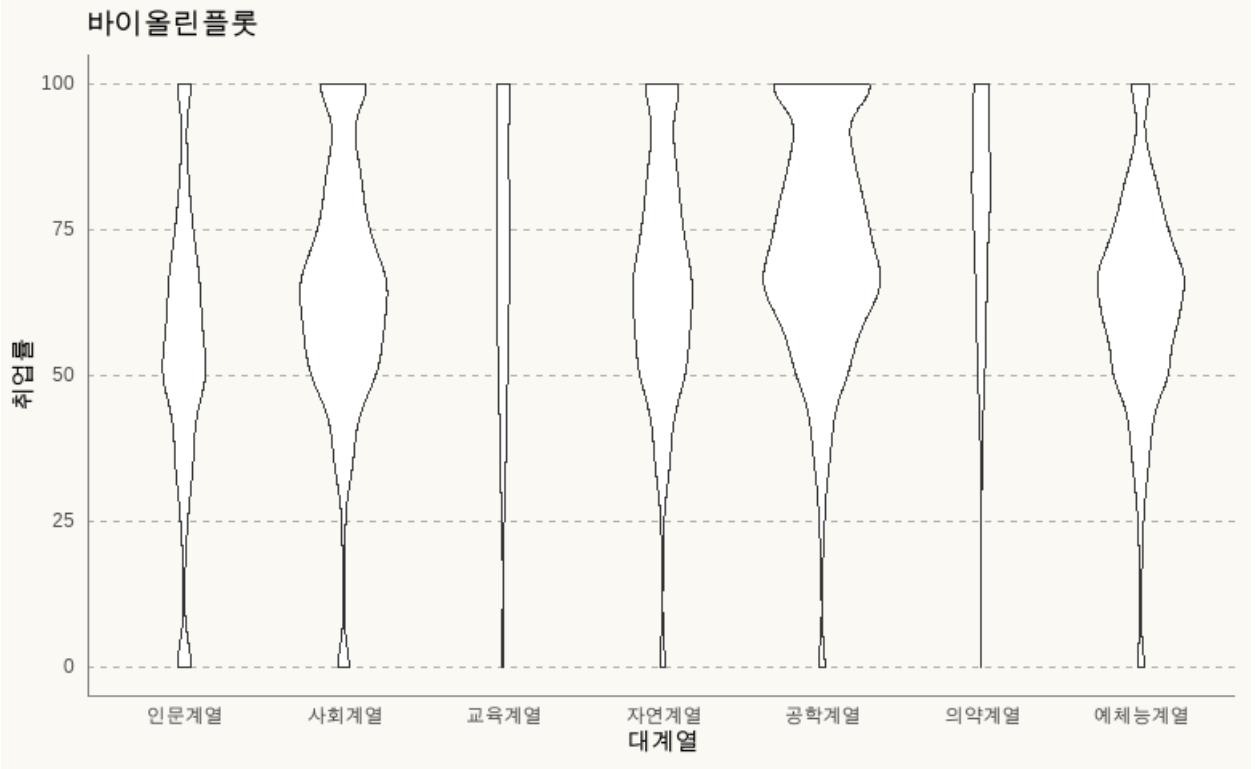
```
## df_취업통계에서 다소간의 왜곡을 방지하기 위해 졸업자가 3 명이상인 학과만 필터링해서 ggplot 객체 생성
p_boxplot <- df_취업통계 |> filter(졸업자_계 >= 3) |>
  ggplot() +
  labs(x = '대계열', y = '취업률')

p_boxplot1 <- p_boxplot +
  ## X 축을 대계열, Y 축을 취업률_계로 매핑한 geom_boxplot 레이어 생성
  geom_boxplot(aes(x = 대계열, y = 취업률_계)) +
  labs(title = '박스 플롯')

p_boxplot1
```




*실행결과 6-21. 기본 박스 플롯*

이를 바이올린 플롯으로 바꾸려면 `geom_boxplot()`을 `geom_violin()`으로 바꾸면 된다. 다만 앞에서 언급한 바와 같이 박스플롯의 단점인 사례수의 표현을 위해서는 `scale` 매개변수를 'count'로 맞춰준다.

```
## 바이올린플롯 생성
p_violin1 <- p_boxplot +
  ## X 축을 대계열, Y 축을 취업률_계로 매핑한 geom_violin 레이어 생성
  geom_violin(aes(x = 대계열, y = 취업률_계), scale = 'count') +
  labs(title = '바이올린플롯')

p_violin1
```



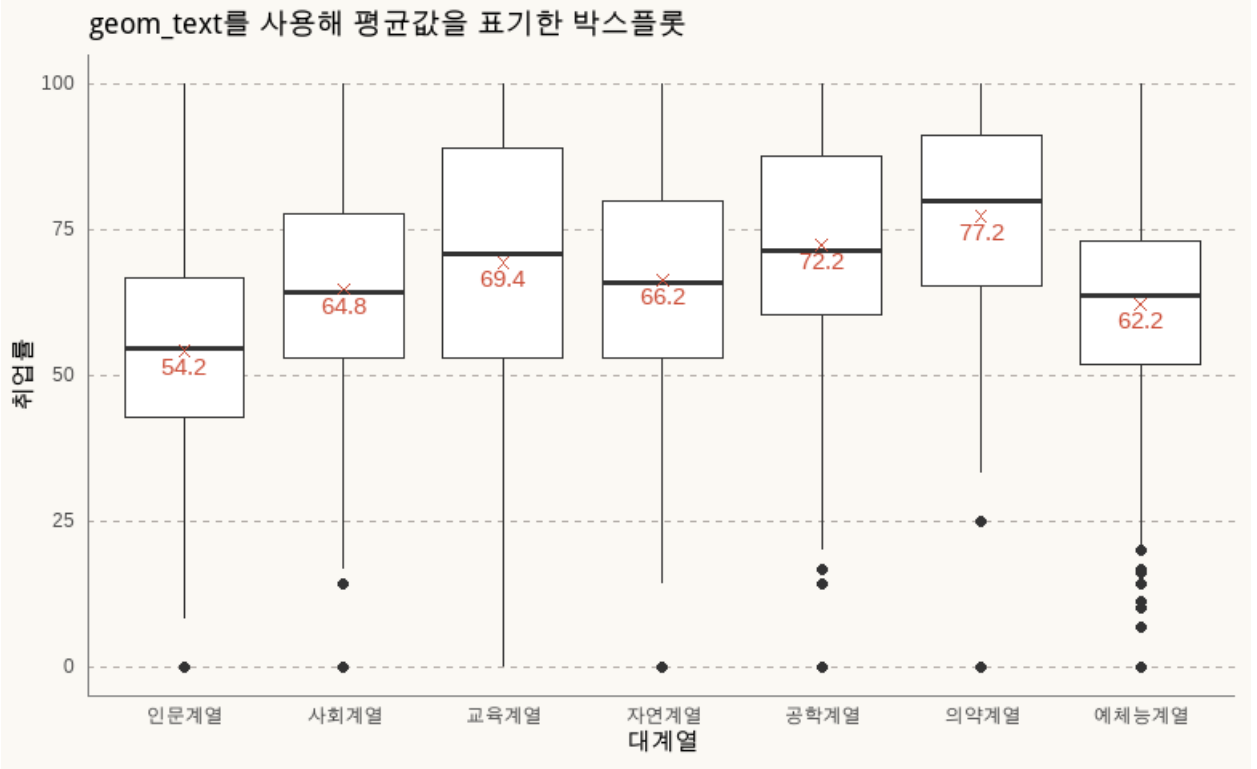

*실행결과 6-22. 기본 바이올린플롯*

앞의 바이올린플롯을 보면 상대적으로 좌우폭이 좁은 계열이 교육계열과 의약계열이다. 이 두 계열은 타 계열보다 사례수가 적다는 것을 알 수 있다.

## 3.2. 평균값 표현

박스플롯을 사용하다 보면 중간값을 평균값으로 오해하는 경우가 많다. 사실 우리는 중간값보다는 평균값에 더 익숙하다. 그렇기 때문에 박스플롯에 평균값을 표기하는 경우가 매우 흔히 발생한다. 다음은 앞서 생성한 기본 박스 플롯에 평균값을 표기하는 코드이다.

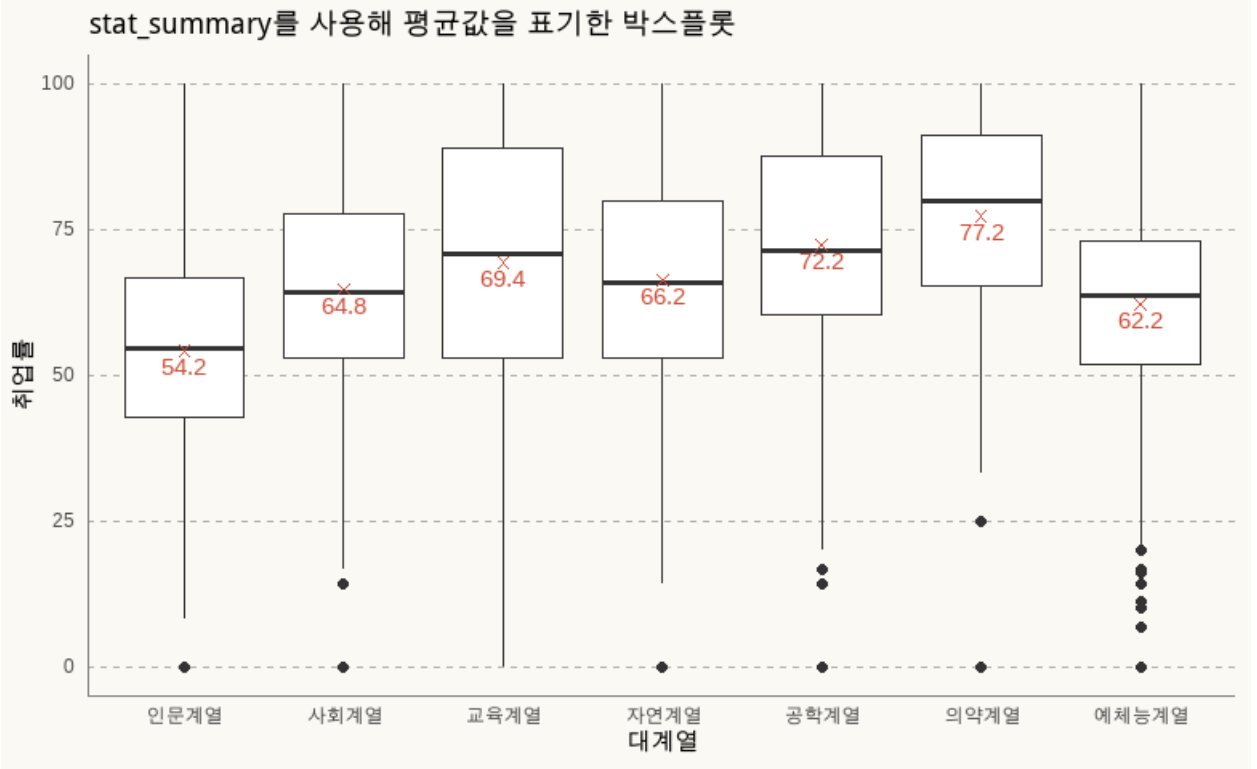
```
p_boxplot1 +
  ## stat = summary, fun.y = mean 으로 설정한 geom_point 레이어 추가
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', col
or = 'tomato3', shape = 4) +
  ## stat = summary, fun.y = mean 으로 설정한 geom_text 레이어 추가
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summa
ry', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  labs(title = 'geom_text 를 사용해 평균값을 표기한 박스플롯')
```




*실행결과 6-23. geom_text 를 사용해 평균값을 표기한 박스플롯*

앞의 코드는 아래의 코드로 같은 결과를 얻을 수 있다.

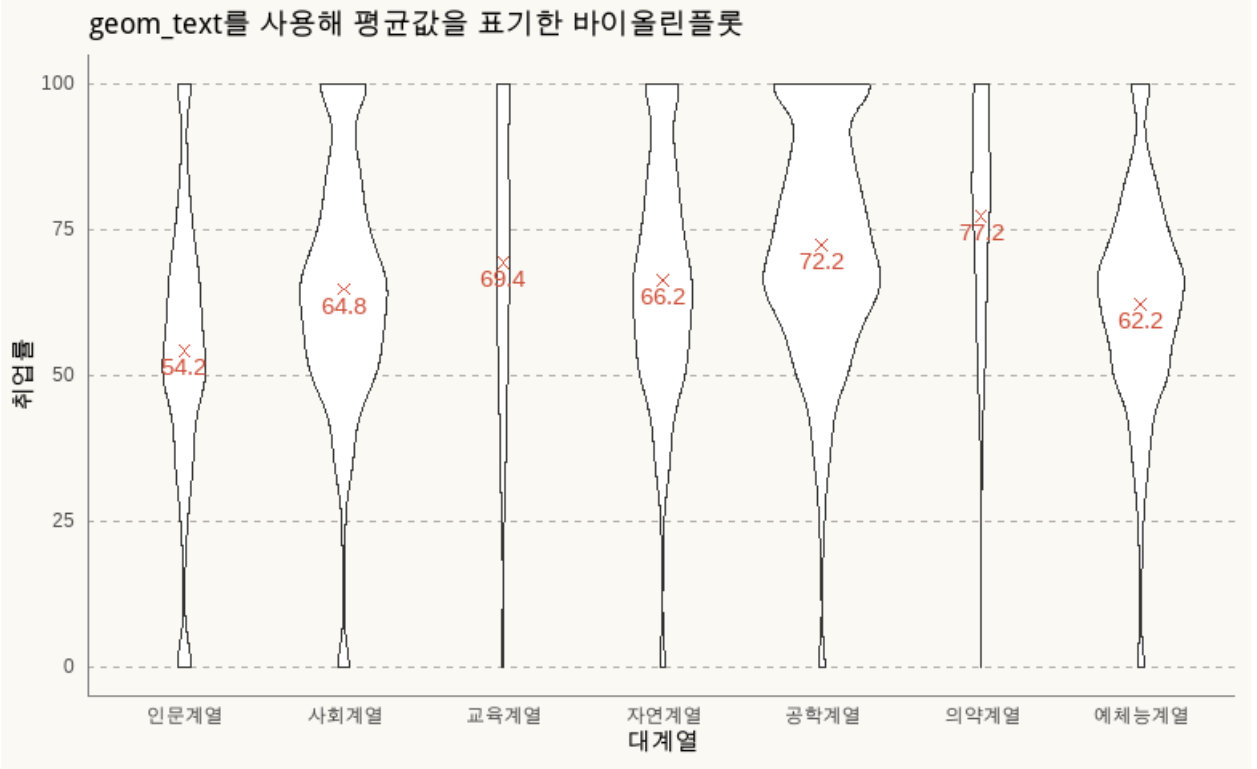
```
p_boxplot1 +
  ## stat = summary, fun.y = mean 으로 설정한 geom_point 레이어 추가
  stat_summary(aes(x = 대계열, y = 취업률_계), geom = 'point', fun.y = 'mean', col
or = 'tomato3', shape = 4) +
  ## stat = summary, fun.y = mean 으로 설정한 geom_text 레이어 추가
  stat_summary(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), geom = 'te
xt', fun.y = 'mean', color = 'tomato3', shape = 4, vjust = 1.5) +
  labs(title = 'stat_summary 를 사용해 평균값을 표기한 박스플롯')
```




*실행결과 6-24. stat_summary 를 사용해 평균값을 표기한 박스플롯*

이를 바이올린 플롯으로 바꾸면 다음과 같다.

```
p_violin1 +
  ## stat = summary, fun.y = mean 으로 설정한 geom_point 레이어 추가
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', col
or = 'tomato3', shape = 4) +
  ## stat = summary, fun.y = mean 으로 설정한 geom_text 레이어 추가
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summa
ry', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  labs(title = 'geom_text 를 사용해 평균값을 표기한 바이올린플롯')
```




*실행결과 6-25. geom_text 를 사용해 평균값을 표기한 바이올린플롯*

## 3.3. 박스플롯의 강조

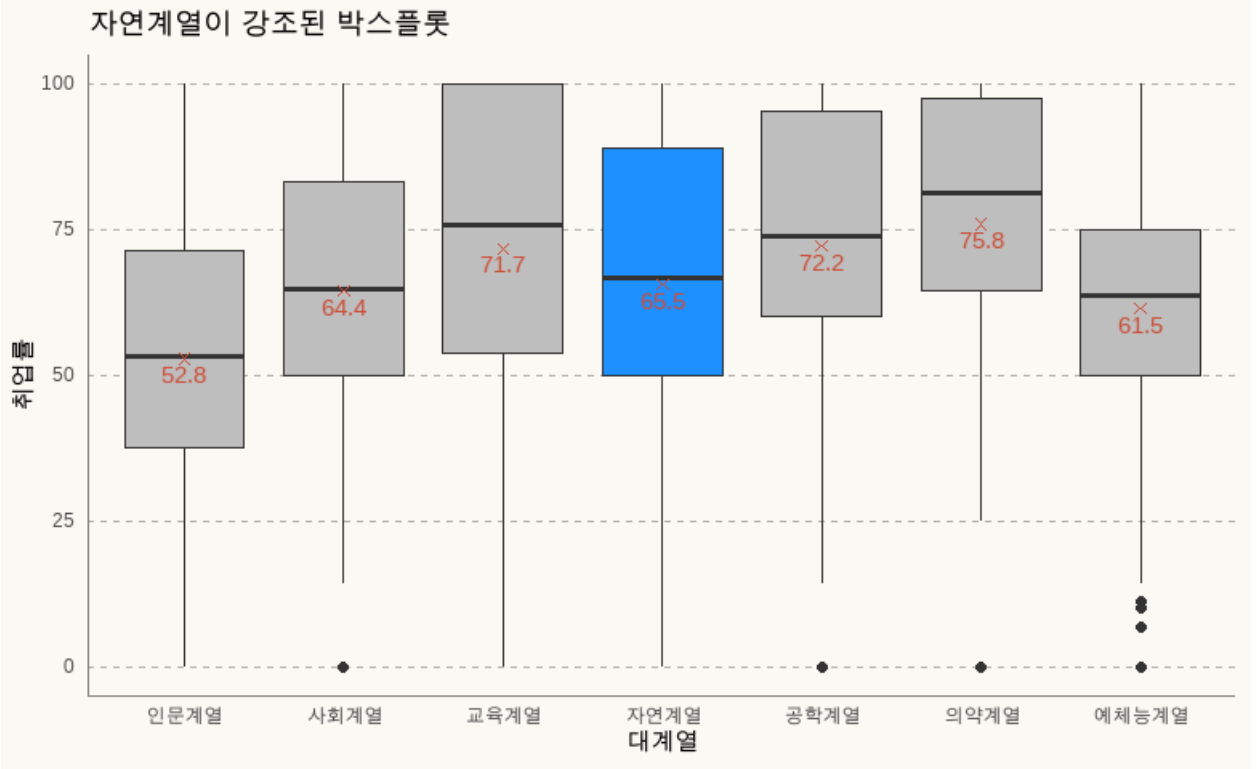
산점도에서도 사용자가 강조하고자 하는 데이터를 강조하는 방법을 설명했다.
박스플롯에서도 사용자가 강조해야할 부분을 강조하여 시각화하는 방법을 많이 사용한다.
주로 사용하는 방법이 특정 부분의 색을 짙게하고 나머지 부분의 색을 옅게 하는 방법이다.
다음의 코드는 앞서 생성한 박스플롯에 '자연계열'을 강조하는 코드이다.

다음의 코드에서는 우선 그룹의 강조를 위한 변수를 생성했다. 여기서는 highlight 변수를 생성하는데 이 변수는 대계열이 강조해야할 '자연계열'이면 1 을 아니면 0 을 가지는 변수이다. 이를 설정하기 위해 `ifelse()`를 사용했다. `ifelse()`는 `case_when()`과 유사하지만 하나 두개 정도의 변수를 설정하는데 유용하다. 더 많아지면 헤깔려서 사용하기 어렵다.

```
ifelse(test, yes, no)
  - test : 데이터 조건절
  - yes : 데이터 조건이 참이라면 설정할 값
  - no : 데이터 조건이 거짓이라면 설정할 값
```

그 이후에 박스 플롯을 생성하고 `color` 를 앞서 생성한 highlignt 변수에 매핑함으로써 해당 그룹을 강조할 수 있다.

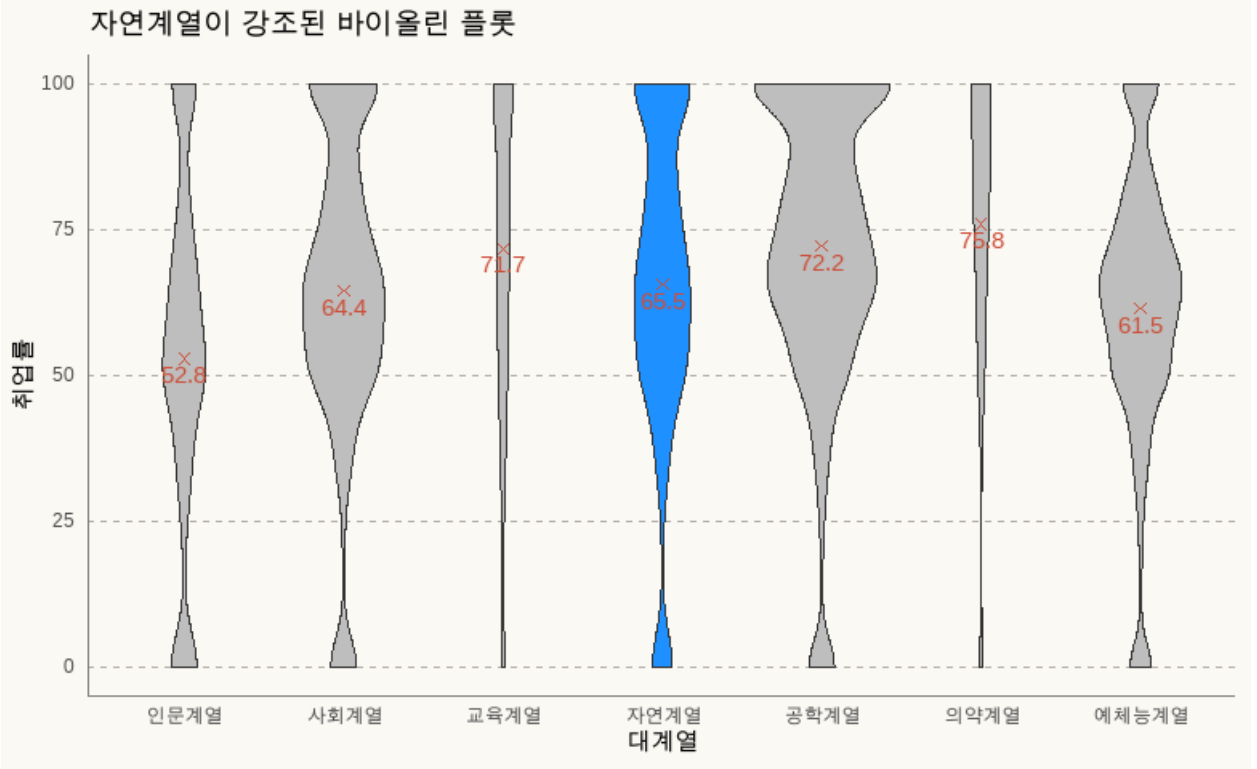
```
## ifelse()를 사용해 highlignt 변수 생성
df_취업통계$highlight <- ifelse(df_취업통계$대계열 == '자연계열', 1, 0)

## ggplot 객체 생성
p_boxplot_highlight <- df_취업통계 |>
  ggplot() +
  labs(x = '대계열', y = '취업률')

p_boxplot_highlight +
  ## highlignt 를 fill 에 매핑한 geom_boxplot 레이어 생성, highling 를 팩터로 바꾼이유는 이산형 색값을 사용하기 위해서임
  geom_boxplot(aes(x = 대계열, y = 취업률_계, fill = as.factor(highlight)), show.legend = FALSE) +
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', color = 'tomato3', shape = 4) +
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summary', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  scale_fill_manual(values=c("grey", "dodgerblue1")) +
  labs(title = '자연계열이 강조된 박스플롯')
```




*실행결과 6-26. 자연계열이 강조된 박스플롯*

이를 바이올린 플롯으로 바꾸면 다음과 같다.

```
## ggplot 객체 생성
p_violin_highlight <- df_취업통계 |>
  ggplot() +
  labs(x = '대계열', y = '취업률')

p_violin_highlight +
  ## highlignt 를 fill 에 매핑한 geom_boxplot 레이어 생성, highling 를 팩터로 바꾼이유는 이산형 색값을 사용하기 위해서임
  geom_violin(aes(x = 대계열, y = 취업률_계, fill = as.factor(highlight)), show.legend = FALSE, scale = 'count') +
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', color = 'tomato3', shape = 4) +
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summary', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  scale_fill_manual(values=c("grey", "dodgerblue1")) +
  labs(title = '자연계열이 강조된 바이올린 플롯')
```

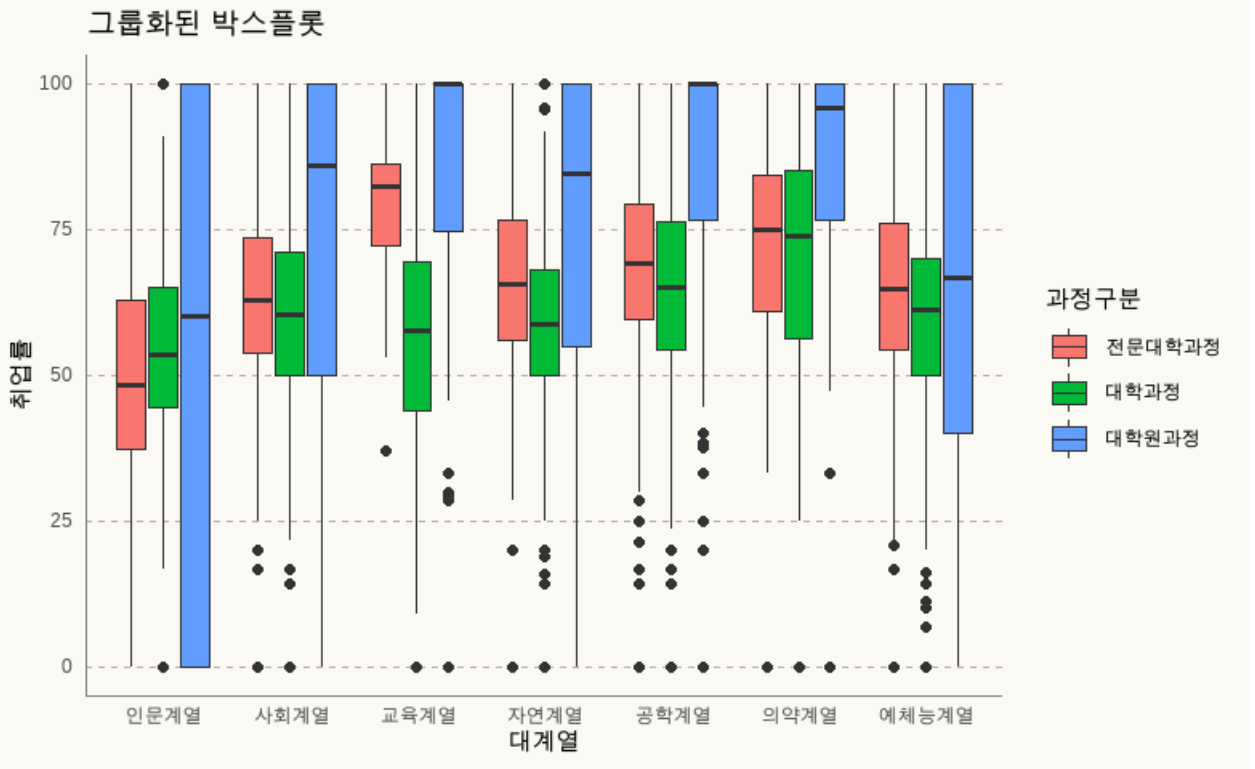



*실행결과 6-27. 자연계열이 강조된 바이올린 플롯*

## 3.4. 그룹화된 박스플롯

박스플롯은 대부분이 X 축은 이산형 데이터, Y 축은 연속형 수치 데이터로 구성된다. X 축이 이산형 데이터이기 때문에 추가적인 변수를 매핑함으로써 이산형 데이터를 다시 세분화할 수 있다. 이렇게 단계적으로 구성된 이산형 데이터는 하위 단계까지 그룹화해서 박스플롯을 만들수 있다. 이런 그룹화 박스플롯은 특별한 함수를 사용하는 것이 아니고 color 나 fill 로 추가 변수를 매핑함으로써 만들수 있다. 다음은 앞서 만든 박스플롯을 '과정구분' 변수로 세부 그룹화하여 만든 박스플롯이다.

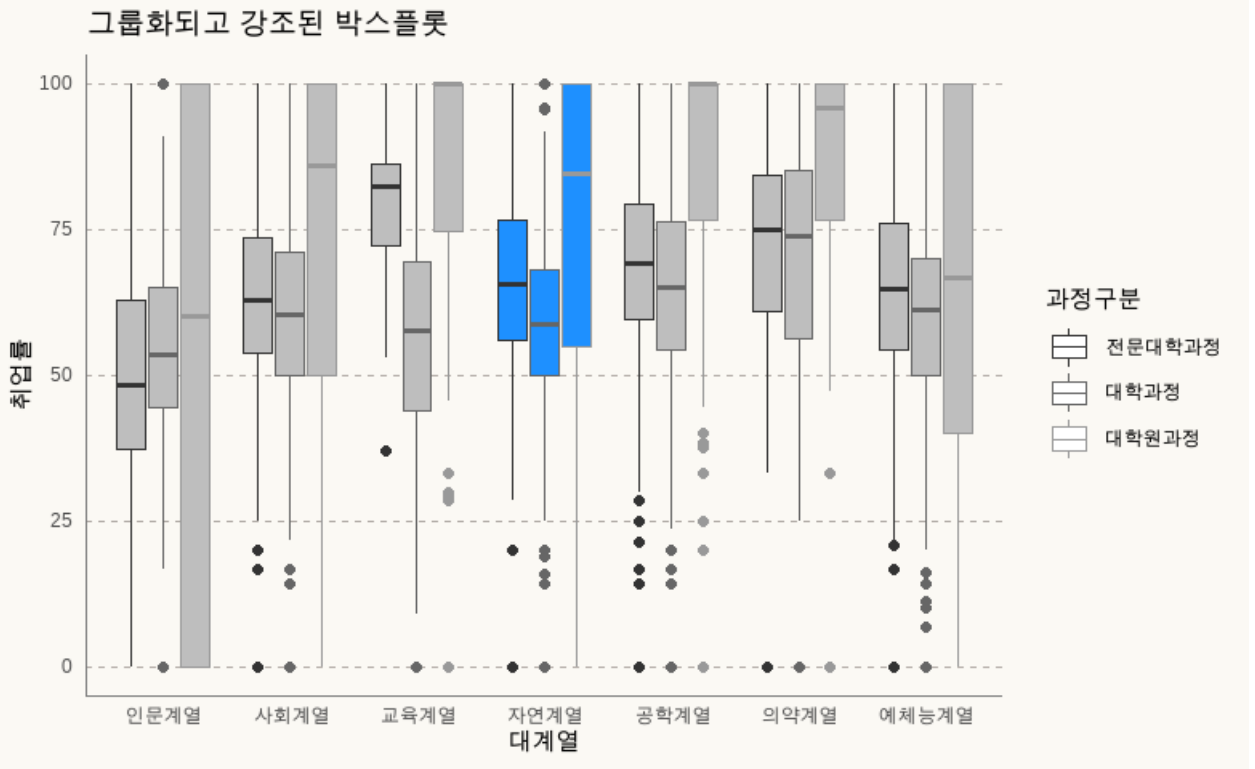
```
p_boxplot_highlight +
  geom_boxplot(aes(x = 대계열, y = 취업률_계, fill = 과정구분)) +
  labs(title = '그룹화된 박스플롯')
```




*실행결과 6-28. 그룹화된 박스플롯*

앞의 박스플롯은 색이 전반적으로 각 그룹별로 설정되어 있기 때문에 특정 계열의 강조가 사라졌다. 그룹화한 박스플롯에 특정 계열을 강조하기 위해서는 다음과 같이 사용할 수 있다.

```
p_boxplot_highlight +
  ## color 매핑을 대계열로 추가한 geom_box 레이어 생성
  geom_boxplot(aes(x = 대계열, y = 취업률_계, color = 과정구분, fill = as.factor(highlight))) +
  ## fill 값을 직접 정의
  scale_fill_manual(values = c('0' = 'grey', '1' = 'dodgerblue1'), guide = FALSE) +
  ## color 값을 직접 정의
  scale_color_manual(values = c('전문대학과정' = 'grey20', '대학과정' = 'grey40', '대학원과정' = 'grey60')) +
  labs(title = '그룹화되고 강조된 박스플롯')
```

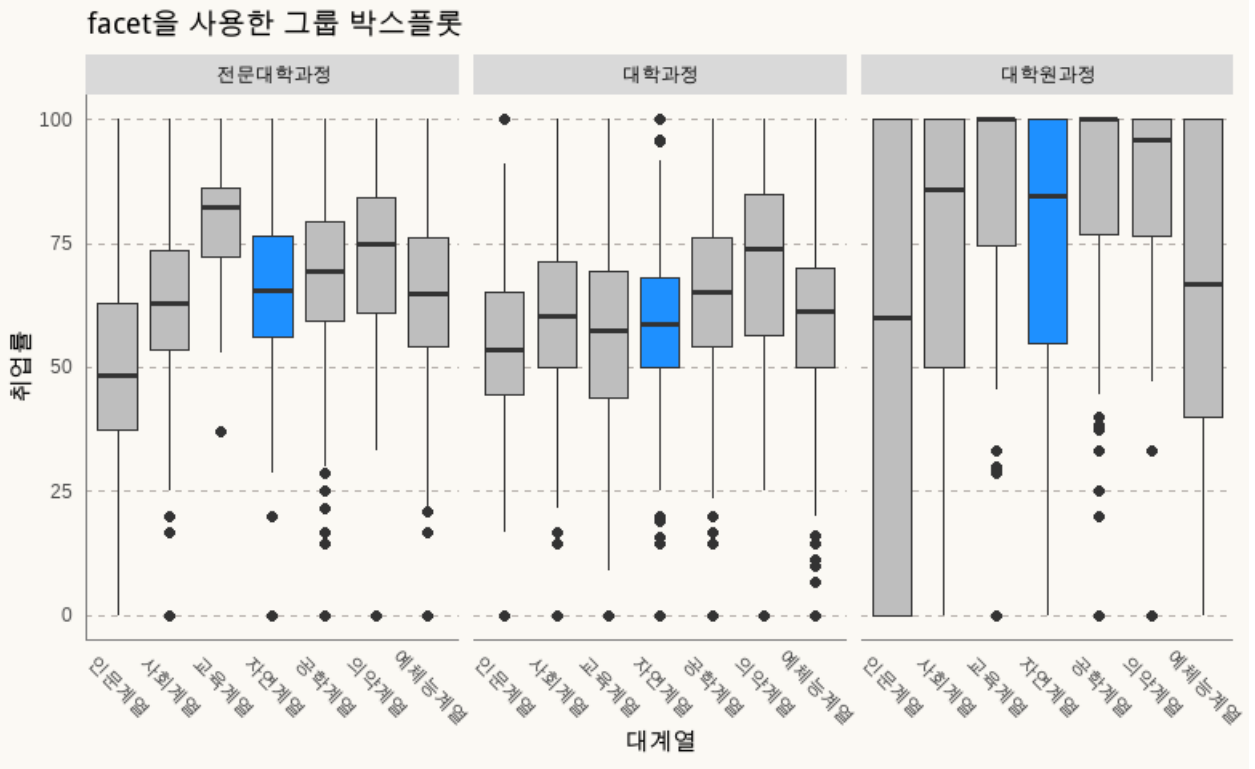



*실행결과 6-29. 그룹화되고 강조된 박스플롯*

## 3.5. 분할된 박스 플롯(facet boxplot)

앞의 그룹화된 박스플롯은 그룹화된 변수의 변량이 적을때는 효과적으로 보이지만 변량이 많아져서 박스플롯이 많아지면 효과적인 시각화가 되지 않는다. 이럴 경우는 `facet_*()`을 사용하여 플롯을 분리시켜주는게 효과적이다.

```
p_boxplot_highlight +
  geom_boxplot(aes(x = 대계열, y = 취업률_계, fill = as.factor(highlight)), show.l
egend = FALSE) +
  ## 과정구분 열을 사용해 박스플롯을 분할
  facet_wrap(~과정구분) +
  ## X 축의 라벨이 겹쳐보기가 어려우니 45 도 기울임
  theme(axis.text.x = element_text(angle = -45, vjust = 0.5)) +
  scale_fill_manual(values=c("grey", "dodgerblue1")) +
  labs(title = 'facet 을 사용한 그룹 박스플롯')
```



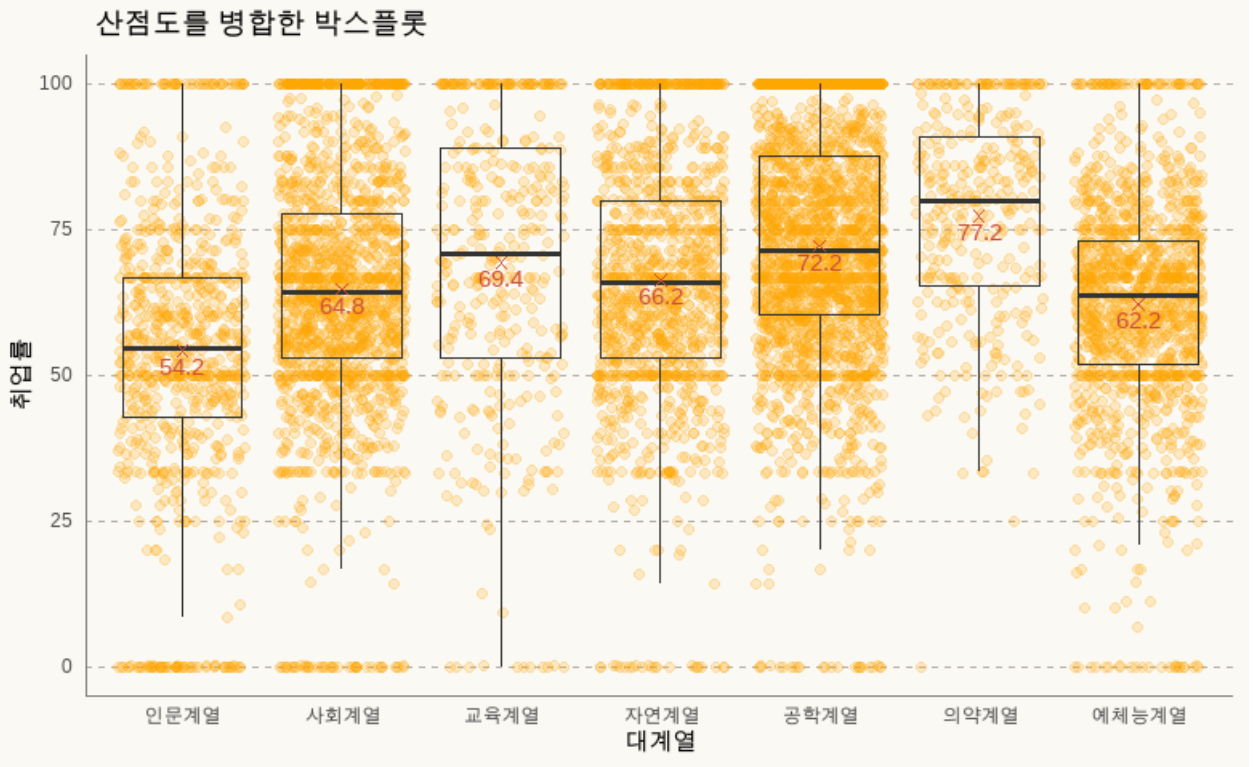

*실행결과 6-30. facet 을 사용한 그룹 박스플롯*

## 3.6. 박스플롯과 산점도의 병합

각 박스플롯의 데이터 분포를 살펴보기 위해 바이올린플롯을 사용했다. 바이올린플롯을 사용하는 것 외에 산점도를 사용하여 표현하는 방법도 있다. 산점도 겹쳐서 사용할 때 하나 주의해야할 것이 일반적인 산점도는 X, Y 축 모두 연속형 변수를 사용했지만 박스플롯에서는 X 축이 이산형 변수라는 것이다. 따라서 X 축이 이산형 변수가 사용된 산점도는 X 축의 각각의 값에 따라 하나의 직선위에 표시되는 산점도가 나타난다. 이렇게 표현된 산점도는 데이터의 분포를 알아보기가 어렵기 때문에 겹치는 데이터들은 X 축의 값을 약간씩 차이를 두어 플로팅하는 산점도를 사용한다. 이는 `geom_jitter()`를 사용하여 표현이 가능하다.

```
p_boxplot +
  ## geom_jitter 레이어를 먼저 생성
  geom_jitter(aes(x = 대계열, y = 취업률_계), alpha = 0.2, color = 'orange') +
  ## geom_jitter 레이어 위에 geom_boxplot 레이어 추가
  geom_boxplot(aes(x = 대계열, y = 취업률_계), alpha = 0) +
  ## geom_jitter, geom_boxplot 레이어 위에 geom_point 레이어 추가
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', col
or = 'tomato3', shape = 4) +
```

```
  ## 마지막으로 geom_text 레이어 추가
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summary', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  labs(title = ' 산점도를 병합한 박스플롯')
```

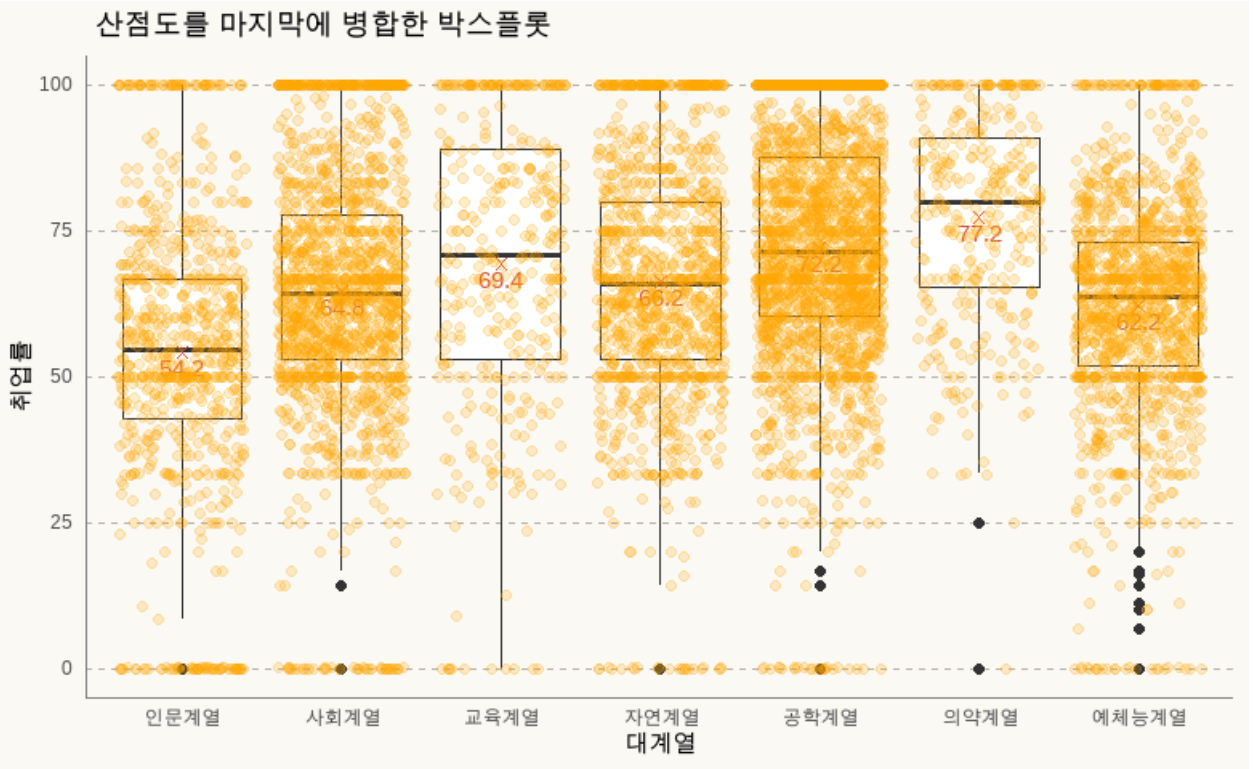



*실행결과 6-31. 산점도를 병합한 박스플롯*

여기서 하나 주의해야 할 것은 각각의 미적요소 레이어의 순서이다. 앞의 코드에서 가장 먼저 `geom_jitter()` 레이어를 그렸고, 그 위에 `geom_boxplot()` 레이어, 평균값 레이어를 겹쳤다. 만약 다음과 같이 `geom_jitter()`를 맨 마지막에 겹치면 앞선 박스플롯과 평균값이 가려져서 잘 보이지 않게 된다.

```
p_boxplot +
  ## geom_boxplot 레이어를 맨 처음 생성
  geom_boxplot(aes(x = 대계열, y = 취업률_계)) +
  ## geom_boxplot 레이어 위에 geom_point 레이어 추가
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', color = 'tomato3', shape = 4) +
  ## geom_boxplot, geom_point 레이어 위에 geom_text 레이어 추가
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summary', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
```

```
  ## 마지막으로 geom_jitter 레이어 추가
  geom_jitter(aes(x = 대계열, y = 취업률_계), alpha = 0.2, color = 'orange') +
  labs(title = ' 산점도를 마지막에 병합한 박스플롯')
```



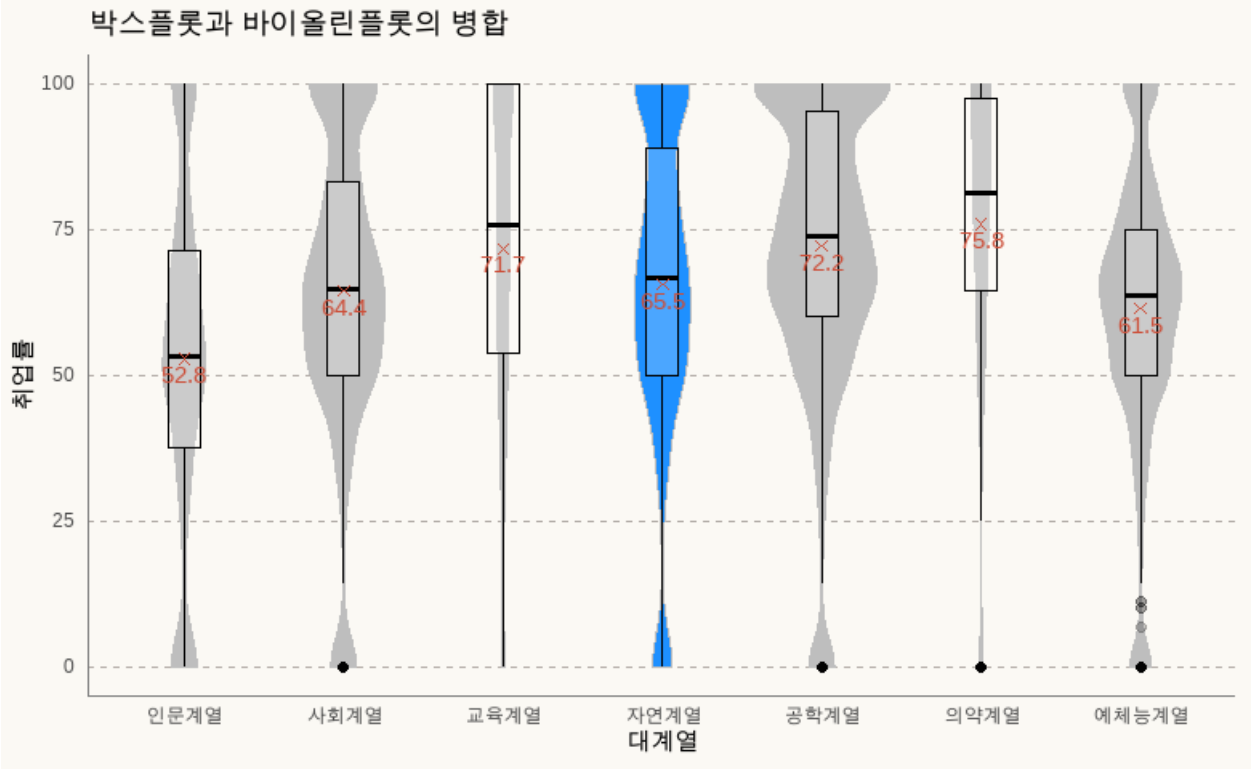

*실행결과6-32. 산점도를 마지막에 병합한 박스플롯*

## 3.7. 박스플롯과 바이올린 플롯의 병합

앞에서 살펴본 박스플롯과 산점도를 병합한 것과 같이 박스플롯과 바이올린플롯도 병합이 가능하다. 사실 산점도를 병합하는 것보다 이 방법이 시각화를 보는 사람들에게 보다 간결하게 정보를 전달한다는 점에서는 강점이 있다.

```
p_violin_merge <- df_취업통계 |>
  ggplot() +
  labs(x = '대계열', y = '취업률')

p_violin_merge +
  ## geom_violin 레이어 생성
  geom_violin(aes(x = 대계열, y = 취업률_계, fill = as.factor(highlight)), color =
 'grey', scale = 'count', show.legend = FALSE) +
  ## geom_violin 레이어 위에 geom_boxplot 레이어 추가
```

```
  geom_boxplot(aes(x = 대계열, y = 취업률_계), width=0.2, color="black", alpha=0.
2) +
  ## 평균점을 찍기 위한 geom_point 레이어 추가
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun.y = 'mean', col
or = 'tomato3', shape = 4) +
  ## 평균값을 표기하기 위한 geom_text 레이어 추가
  geom_text(aes(x = 대계열, y = 취업률_계, label = round(..y.., 1)), stat = 'summa
ry', fun.y = 'mean', color = 'tomato3', vjust = 1.5) +
  ## fill 의 스케일 설정
  scale_fill_manual(values=c("grey", "dodgerblue1")) +
  labs(title = '박스플롯과 바이올린플롯의 병합')
```




*실행결과 6-33. 박스플롯과 바이올린플롯의 병합*

## 4. 피라미드 플롯

피라미드 플롯은 주로 인구의 분포를 시각화는데 많이 사용된다. 아마도 아래의 그림과 같은 플롯을 한번쯤은 본적이 있을 것이다.

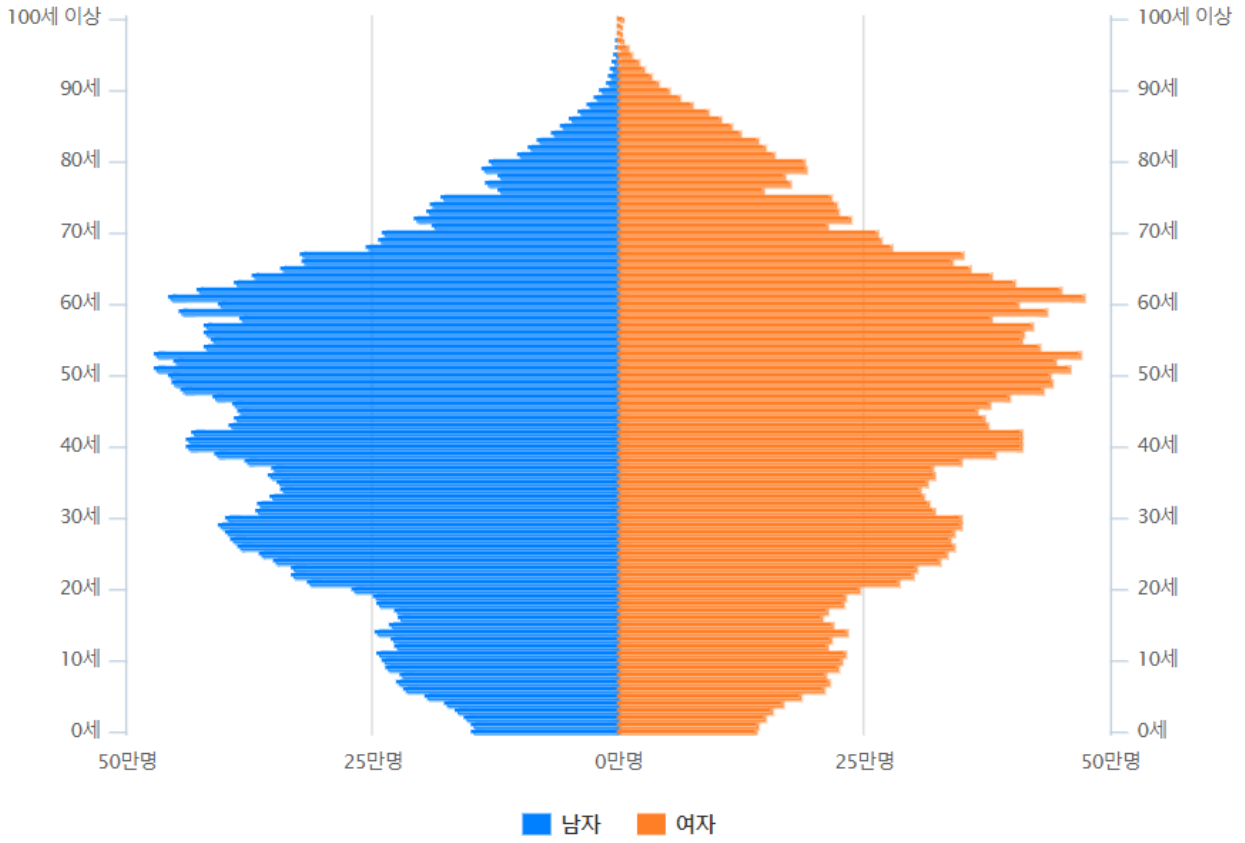
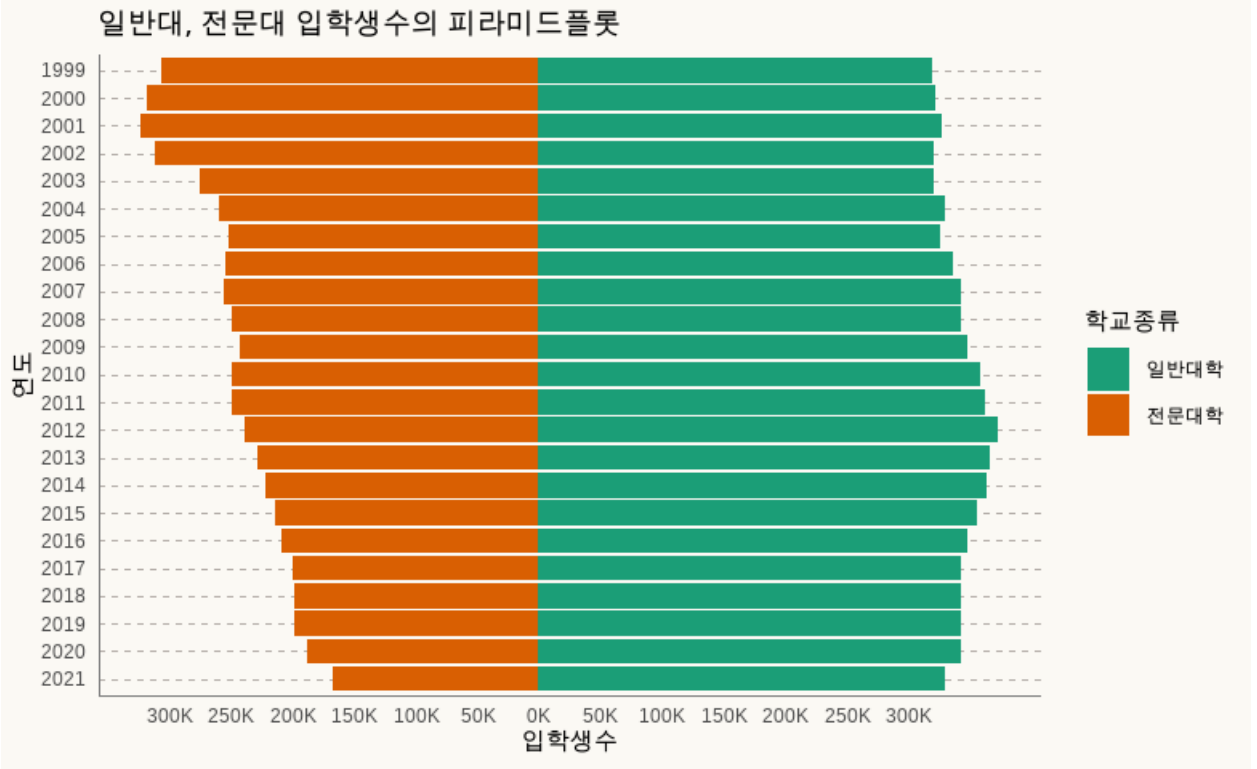



*그림 6-1 인구 피라미드 플롯 ( 출처 : https://sgis.kostat.go.kr/jsp/pyramid/pyramid1.jsp)*

`ggplot2` 에서는 피라미드 플롯을 구현하는 함수를 제공하지는 않는다. 이 피라미드 플롯을 생성하려면 대비되는 두개의 그룹을 특정 변수로 분리하고 한쪽에는 양의 값으로 나머지 한쪽은 음의 값을 가지는 값으로 설정하고 축을 돌려줌으로써 생성할 수 있다. 다음은 입학자의 연도별 변화를 전문대학과 일반대학의 그룹으로 분리하고 피라미드 플롯을 그린 코드이다.

```
## 피라미드 플롯에서 사용할 축의 눈금과 라벨을 설정
brks <- seq(from = -300000, to = 300000, by = 50000)
lbls <- paste0(as.character(c(seq(300, 0, -50), seq(50, 300, 50))), "K")

## df_입학자_long 데이터에서
df_입학자_pyramid<- df_입학자_long |>
  ## 대비할 두개의 그룹(전문대학, 일반대학)을 필터링하고
  filter(학교종류 %in% c('전문대학', '일반대학'), 지역 == '전체') |>
  ## 전문대학 입학생수를 음수로 변환
  mutate(입학생수 = ifelse(학교종류 == '전문대학', 입학생수 * -1, 입학생수))
```

```r
## ggplot 객체 생성
p_pyramid <- df_입학자_pyramid |>
  ggplot() +
  labs(title = '일반대, 전문대 입학생수의 피라미드플롯')


p_pyramid +
  ## X 축은 연도, Y 축은 입학생수, fill 은 학교종류로 매핑한 geom_col 레이어 생성
  geom_col(aes(x = 연도, y = 입학생수, fill = 학교종류)) +
  ## Y 축의 눈금과 라벨을 미리 생성해 둔 값으로 설정
  scale_y_continuous(breaks = brks, labels = lbls) +
  ## X 축의 순서를 반전
  scale_x_discrete(limits=rev) +
  ## X 축과 Y 축의 위치를 전환
  coord_flip() +
  ## 색 팔레트를 설정
  scale_fill_brewer(palette = "Dark2")
```




*실행결과 6-34. 일반대, 전문대 입학생수의 피라미드플롯*