

III. 데이터 시각화를 위한 **ggplot2**

지금까지 데이터 시각화를 위한 R의 기초 사용법에 대해 알아보았다. 이제 본격적으로 **ggplot2** 패키지를 위주로 데이터 시각화 방법을 알아보도록 하자.

먼저 데이터 시각화를 실습하기 위한 데이터를 설정하도록 하겠다. 두 개의 데이터를 사용할 것인데 하나는 2장에서 사용했던 df_입학자 데이터이고 나머지 하나는 교육통계 서비스 홈페이지에서 제공하는 학교/학과별 데이터셋의 2020년 취업통계 학과별 데이터 셋¹을 사용하겠다. 취업통계 데이터 셋을 불러 들이는 코드는 다음과 같다.²

```
df_취업통계 <- read_excel('2020년 학과별 고등교육기관 취업통계.xlsx',
                           ## '학과별' 시트의 데이터를 불러오는데,
                           sheet = '학과별',
                           ## 앞의 13 행을 제외하고
                           skip = 13,
                           ## 첫번째 행은 열 이름으로 설정
                           col_names = TRUE,
                           ## 열의 타입을 설정, 처음 9개는 문자형으로 다음 79개는 수치형으로 설정
                           col_types = c(rep('text', 9), rep('numeric', 79)))

## df_취업통계에서 첫번째부터 9번째까지의 열과 '계'로 끝나는 열을 선택하여 다시 df_취업통계에 저장
df_취업통계 <- df_취업통계 |> select(1:9, ends_with('계'))

## df_취업통계 정보 확인
str(df_취업통계)

## tibble [9,290 x 35] (S3: tbl_df/tbl/data.frame)
## $ 조사기준일           : chr [1:9290] "2020.12.31" "2020.12.31" "2020.12.31" "2020.12.31" ...
## $ 학제                 : chr [1:9290] "전문대학(4년제)" "전문대학(4년제")
```

¹ https://kess.kedi.re.kr/contents/dataset?itemCode=04&menuId=m_02_04_03_02&tabId=m3

² 해당 데이터는 필자의 블로그(2stndard.tistory.com)에서 다운로드 받을 수 있다.

```
제)" "전문대학(3 년제)" "전문대학(3 년제)" ...
## $ 과정구분 : chr [1:9290] "전문대학과정" "전문대학과정" "전문
대학과정" "전문대학과정" ...
## $ 대계열 : chr [1:9290] "의약계열" "의약계열" "인문계열" "
인문계열" ...
## $ 중계열 : chr [1:9290] "간호" "간호" "언어 · 문학" "언어 ·
문학" ...
## $ 소계열 : chr [1:9290] "간호" "간호" "일본어" "일본어"
...
## $ 학과코드 : chr [1:9290] "C06010100004" "C06010100008" "C
01010100002" "C01010100003" ...
## $ 학과명 : chr [1:9290] "간호학과" "간호학과(4 년제)" "관광
일본어과" "관광일본어전공" ...
## $ 학위구분 : chr [1:9290] NA NA NA NA ...
## $ 졸업자_계 : num [1:9290] 391 482 31 1 74 27 46 32 54 172
...
## $ 취업률_계 : num [1:9290] 81.7 81.6 40.9 0 29.9 37.5 36.8
41.4 39.6 46.4 ...
## $ 취업자_합계_계 : num [1:9290] 317 390 9 0 20 9 14 12 19 70 ...
## $ 취업자_교외취업자_계 : num [1:9290] 313 388 8 0 18 6 12 9 15 58 ...
## $ 취업자_교내취업자_계 : num [1:9290] 0 0 0 0 0 1 0 1 1 1 ...
## $ 취업자_해외취업자_계 : num [1:9290] 0 0 0 0 0 0 0 0 0 0 ...
## $ 취업자_농림어업종사자_계 : num [1:9290] 0 0 0 0 0 0 0 0 0 0 ...
## $ 취업자_개인창작활동종사자_계 : num [1:9290] 0 0 0 0 0 0 0 0 0 0 ...
## $ 취업자_1 인창(사)업자_계 : num [1:9290] 0 0 0 0 0 0 0 0 0 3 ...
## $ 취업자_프리랜서_계 : num [1:9290] 4 2 1 0 2 2 2 2 3 8 ...
## $ 진학률_계 : num [1:9290] 0 0 16.1 0 5.4 11.1 10.9 6.3 3.7
6.4 ...
## $ 진학자_계 : num [1:9290] 0 0 5 0 4 3 5 2 2 11 ...
## $ 취업불가능자_계 : num [1:9290] 0 0 0 0 0 0 0 0 0 0 ...
## $ 외국인유학생_계 : num [1:9290] 1 0 0 0 0 0 0 0 0 1 ...
## $ 제외인정자_계 : num [1:9290] 1 4 0 1 2 0 3 1 0 4 ...
## $ 기타_계 : num [1:9290] 70 88 13 0 47 14 24 15 26 81 ...
## $ 미상_계 : num [1:9290] 1 0 0 0 0 1 0 2 3 0 ...
## $ 1차 유지취업자_계 : num [1:9290] 298 377 7 0 16 6 12 10 13 51 ...
## $ 1차 유지취업률_계 : num [1:9290] 95.2 97.2 87.5 0 88.9 85.7 100 1
```

```

00 81.3 86.4 ...
## $ 2 차 유지취업자_계 : num [1:9290] 292 362 7 0 11 6 8 10 12 41 ...
## $ 2 차 유지취업률_계 : num [1:9290] 93.3 93.3 87.5 0 61.1 85.7 66.7
100 75 69.5 ...
## $ 3 차 유지취업자_계 : num [1:9290] 280 347 7 0 8 5 8 7 11 36 ...
## $ 3 차 유지취업률_계 : num [1:9290] 89.5 89.4 87.5 0 44.4 71.4 66.7
70 68.8 61 ...
## $ 4 차 유지취업자_계 : num [1:9290] 277 342 7 0 8 4 7 7 11 33 ...
## $ 4 차 유지취업률_계 : num [1:9290] 88.5 88.1 87.5 0 44.4 57.1 58.3
70 68.8 55.9 ...
## $ 입학당시 기취업자_계 : num [1:9290] 63 38 2 0 5 2 0 0 3 9 ...

```

1. **ggplot2** 란

ggplot2 패키지는 R에서 데이터 시각화를 위해 가장 널리 사용되는 패키지이다. 이 패키지는 R-Studio의 수석 데이터 사이언티스트인 Hadley Wickham이 주도적으로 개발한 패키지로 2005년 발간된 Leland Wilkinson의 The Grammar of Graphics을 기본으로 작성되었다.

The Grammar of Graphics는 데이터를 효과적으로 요소를 다음의 7 가지로 구분하였다.

ggplot2에서는 이 7 가지 요소를 사용하여 데이터를 시각화하도록 각종 함수들을 제공하고 있다.

- **data(데이터)**

시각화에서 표현해야 할 데이터를 지정한다. 하나의 **ggplot2** 시각화에는 하나 이상의 데이터를 사용할 수 있는데 최소 하나 이상의 데이터가 필수적으로 포함되어야 한다. **ggplot2**에서 지원하는 데이터 타입은 데이터프레임이나 **tibble**이다. 내부적으로 데이터프레임은 **tibble**로 변환되어 사용된다.

- **Aesthetics(미적요소)**

Aesthetics는 데이터 시각화에서 시각적 속성과 해당 시각적 속성을 연결시키는 매팅 정보를 표현한다. **Aesthetics**로 매팅될 수 있는 시각적 속성은 X, Y 축, 색깔, 크기 등이다.

- **Geometries(기하요소)**

Geometries 는 데이터 시각화에서 실질적으로 데이터를 표현하는데 사용되는 기하학적 도형을 말한다. 기하학적 도형은 점, 선, 막대 등이 있는데 각각의 기하 요소는 각각의 레이어로 시각화되고 이 기하 요소의 레이어를 여러개 사용하여 동시에 여러개의 기하요소를 사용할 수 있다.

- Statistics(통계요소)

데이터 시각화에 표현될 데이터가 원데이터가 아닌 mean, median 등 통계처리되어 표현할 경우 사용되는 요소이다.

- Facet(분할요소)

데이터 시각화에 표현되는 데이터가 일변량(univariate)이 아닌 다변량(multivariate)인 경우 하나의 시각화에 각 변량의 기하요소를 여러개 표현하면 여러 기하요소가 혼재되어 데이터 시각화를 통한 분석이 어렵게 된다. 이런 경우 다변량 기하 요소를 일변량화하여 일변량 시각화로 분할하여 표현할 수 있다.

- Coordinates(좌표요소)

데이터 시각화에 사용되는 좌표계를 설정한다. 사용되는 좌표계는 우리가 흔히 X, Y 축의 2 차원 좌표계인 데카르트 좌표계(Cartesian Coordinates)나 극좌표계(Polar Coordinates) 등이 제공된다.

- Theme(테마요소)

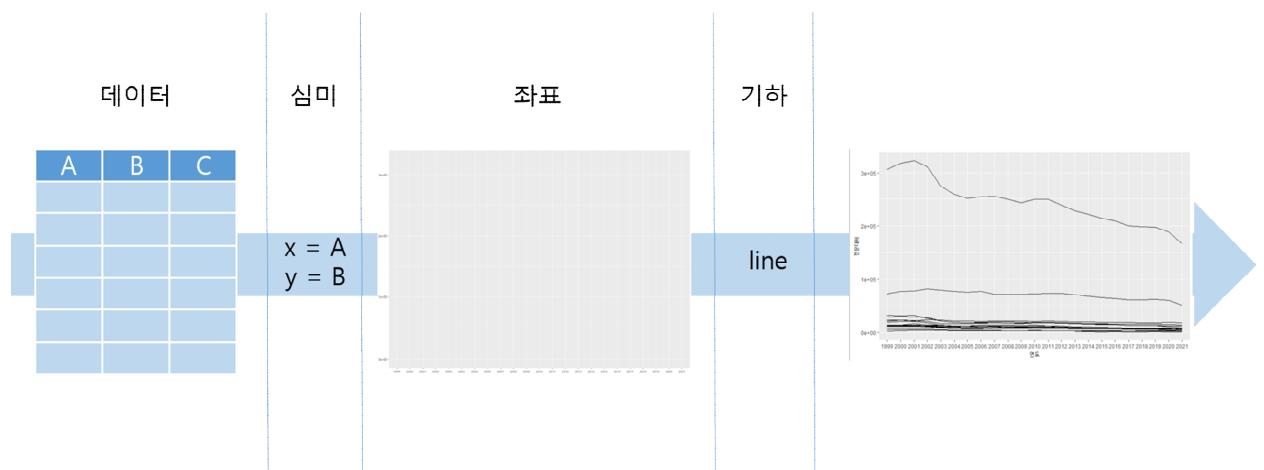
시각화 제목, 축 제목, 축 단위, 범례 등 데이터 시각화의 전반적인 디자인을 꾸며주는 각종 요소를 말한다. 자신만의 Theme 을 만들거나 미리 정의된 Theme 을 적용하여 사용할 수 있다.

ggplot2 는 일정하고 반복적인 코드를 사용하여 일관된 문법을 제공한다는 점이 가장 큰 장점이다. 함수의 사용법이 일관적이어서 편리하고 읽기가 쉽다는 점과 그래픽의 완성도가 또한 큰 장점이기 때문에 R 사용자들이 가장 애용하는 데이터 시각화 패키지이다.

ggplot2 는 레이어를 추가하는 방식으로 여러 타입의 데이터 시각화 요소들을 동시에 표현할 수 있다. 위에서 설명한 7 가지 레이어를 적절히 겹쳐서 시각화를 표현한다.

2. `ggplot2` 요소

`ggplot2`에서 사용하는 7 가지 요소중에 필수적으로 필요한 요소는 데이터, 좌표요소, 기하요소의 세가지이다. 하지만 데이터와 좌표 요소를 미적요소를 통해 연결하게 되기 때문에 결국 데이터, 좌표요소, 기하요소와 미적요소가 가장 기본적인 `ggplot2`의 요소라고 할 수 있겠다. 이 중 좌표 요소는 별다른 함수를 사용하지 않아도 기본적으로 설정되고 나머지 데이터, 기하요소, 미적요소는 `ggplot2`에서 제공하는 각종 함수를 사용하여 지정해야 한다.



`ggplot2`의 문법은 아래의 그림과 같다. 시작은 반드시 `ggplot()` 함수부터이다. 이후는 순서에 큰 영향은 없으나 일반적으로 `geom_`으로 시작하는 기하요소 함수를 사용하며 이 기하요소 함수 안에서 `aes()` 함수를 사용하여 미적 요소를 매핑한다. 각각의 기하요소에 공통적으로 적용될 미적요소의 매핑은 `ggplot()`에 넣어 준다. `ggplot()`를 사용하여 여러개의 레이어를 중첩하여 사용하기 위해서는 문법에서 제공하는 각각의 함수를 파이프가 아닌 `+` 기호 연결하여 사용한다.

필수요소

ggplot(data = 데이터, mapping = aes(심미매핑)) +

기하요소함수 (mapping = aes(심미매핑)),

stat = 통계요소, position = 위치설정 +

좌표요소함수 +

분할요소함수 +

스케일 함수 +

테마요소함수 +

선택요소

2.1. ggplot()

ggplot() 는 ggplot() 객체를 만들기 위해 초기화하는 함수이다. ggplot() 은 시각화를 위한 대상 데이터를 선언하고 전체 레이어에서 공통적으로 사용될 미적 요소들을 지정하는데 사용된다. ggplot() 의 사용법과 주요 매개변수는 다음과 같다.

ggplot(data = NULL, mapping = aes(), ...)

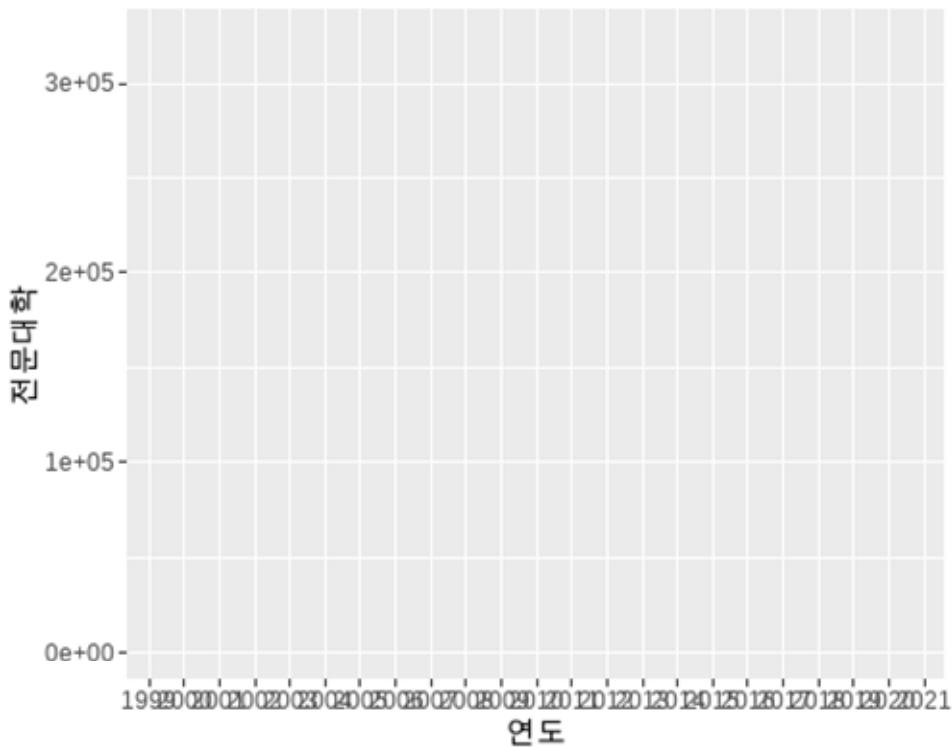
- data : ggplot 객체를 위해 사용할 전체 레이어에 공통으로 사용될 기본 데이터프레임을 선언

- mapping : aes()를 사용하여 전체 레이어에 공통으로 사용될 미적 요소 선언

ggplot() 에는 데이터에 연결된 기하요소가 선언되지 않기 때문에 데이터 시각화가 완전히 완성되지는 않는다. 하지만 데이터 미적요소의 매핑 결과는 볼 수 있다. 아래의 코드를 실행하면 앞 장에서 읽어들인 입학자 데이터를 x 축과 Y 축에 매핑한 결과를 볼 수 있다.

```
## df_입학자 데이터를 사용하여 x 축은 연도열, y 축은 전문대학열을 매핑하는 ggplot 객체 생성
```

```
ggplot(df_입학자, aes(x = 연도, y = 전문대학))
```



위의 코드에서 사용된 `aes()` 는 미적요소을 매핑을 생성하는데 사용되는 함수이다. 미적요소 매핑은 어떤 데이터의 열(변수)가 어떤 시각화 요소들로 연결되는지를 선언하는 방법을 말한다. 위의 코드에서 `aes()` 에 `x` 시각화 요소중에 X 축을 나타내는 매개변수인데 이 `x` 가 `df_입학자` 데이터프레임의 연도열과 연결되었다. 또 `y` 는 시각화 요소 중 Y 축을 나타내는 매개변수로 이 `y` 가 `df_입학자`의 전문대학열과 연결되었다. 이 함수는 `ggplot()` 뿐 아니라 기하요소를 정의하는 `geom_*` 함수에서도 공통적으로 사용된다. `aes()` 의 사용법과 주요 매개변수는 다음과 같다.

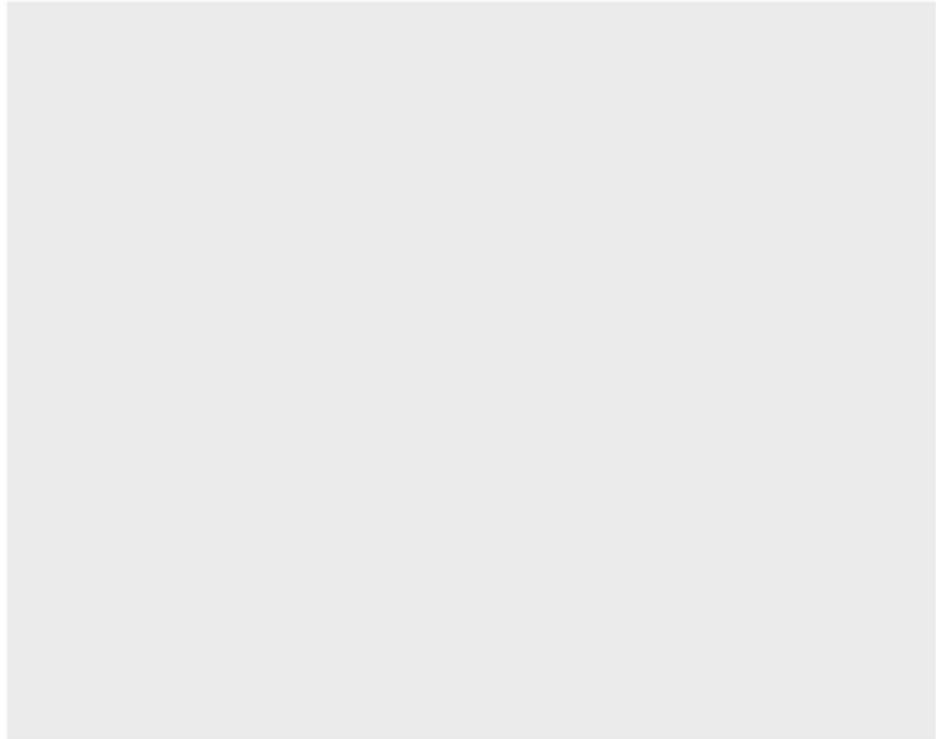
`aes(x, y, ...)`

- `x` : X 축에 매핑될 데이터 열
- `y` : y 축에 매핑될 데이터 열

앞서 설명한 바와 같이 `ggplot()` 에 선언되는 데이터와 미적 요소은 전체 `ggplot` 객체의 레이어에 공통으로 적용되는 데이터와 미적요소이다. `ggplot()` 에 선언되는 미적요소은 X 축과 Y 축의 매핑만 하는 것이 일반적이다. 모든 레이어에 공통적으로 적용되는 미적요소이 많지 않기 때문에 미리 정의해봐야 일부 레이어에서만 사용되기 때문이다. 물론 `ggplot()` 에서 선언된 미적요소들이 각각의 레이어에서 다시 선언되는 경우에는 `ggplot()` 에서 선언된 데이터와 미적 요소보다 각각의 레이어에서 선언되는 데이터와 미적 요소이 해당 레이어에서 우선된다.

만약 모든 레이어에 공통적으로 적용되는 데이터나 미적 요소들이 없다면 매개변수 없이도 사용될 수 있고 데이터만 선언될 수도 있다. 하지만 데이터 선언없이 미적요소의 선언은 불가하다. 데이터가 없거나 미적요소가 없다면 아래와 같이 빈 `ggplot` 객체가 생성된다.

```
ggplot(df_입학자)
```



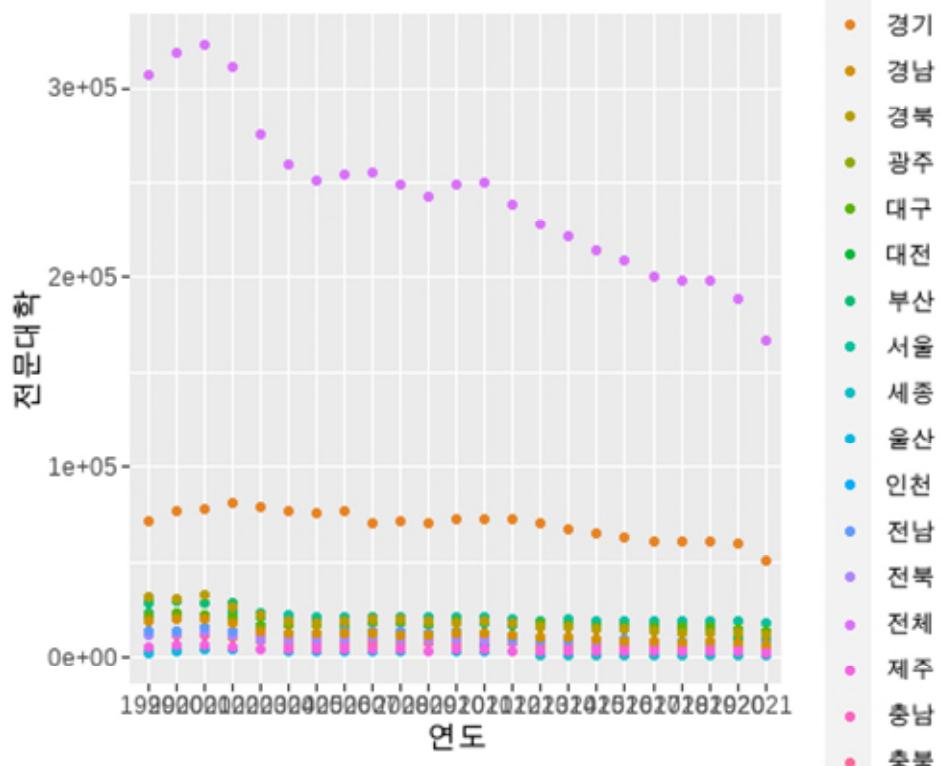
2.2. 미적요소

미적요소는 데이터를 표현하는데 사용되는 필요한 요소들을 통칭한다. 즉 미적요소로 지정할 수 있는 모든 요소은 `ggplot` 객체의 X, Y 좌표내에 표현되는 기하요소의 표현을 위한 요소들로 시각적 요소와 데이터 변수간의 매팅을 통해서 구현된다.

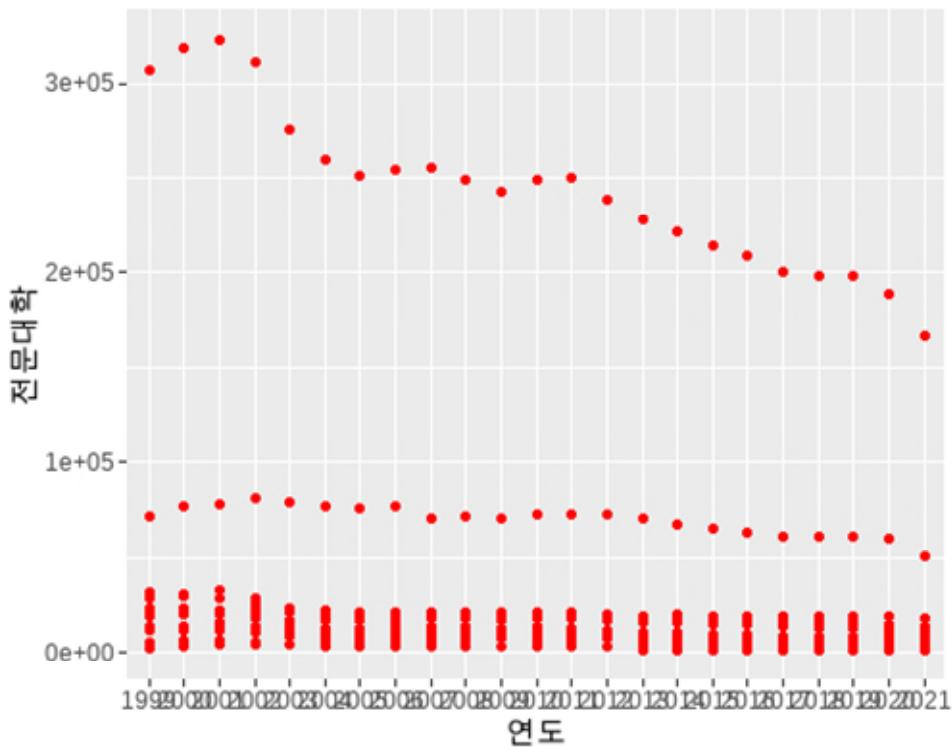
미적요소는 `aes()` 를 사용하여 매팅할 수 있고 고정값을 설정할 수 있다. 이 부분은 미적요소의 사용에 가장 혼동을 유발하는 부분이다. 미적요소를 매팅한다는 것은 미적요소가 데이터 변수에 의해 변경되어야 하는 경우 사용한다. 즉 미적요소가 변수에 대응됨으로써 변수의 변량에 따라 해당 미적요소들이 바뀌어서 표현된다. 그러나 고정값을 설정한다는 것은 변수에 대응되는 것이 아닌 특정값에 고정되도록 설정하기 때문에 고정값으로 설정된 기하요소들은 변수의 변량이 변경되어도 같은 미적요소로 표현된다. 미적요소는 사용하고자 하는 기하요소 함수(`geom_*`()) 내에서 사용되지만 매팅할 때는 반드시 `aes()` 함수를 사용하여 데이터 열이나

매핑 변수를 설정하여야 하고 고정값을 설정할 때는 반드시 `aes()` 함수 바깥에서 선언되어야 한다. 다음의 예를 살펴보자.

```
df_입학자 |> ggplot(aes(x = 연도, y = 전문대학)) +  
  geom_point(aes(color = 지역))
```



```
df_입학자 |> ggplot(aes(x = 연도, y = 전문대학)) +  
  geom_point(color = 'red')
```



위의 코드에서 차이는 `color` 미적요소이 `aes()` 내에 df_입학자의 열이 매핑된 것과 `aes()` 밖에서 고정값(red)로 설정된 것이다. 결과에서 보이듯이 매핑된 `color`는 지역 변수의 변량에 따라 `color` 가 자동적으로 바뀌어 데이터의 구분이 확연히 보인다. 반면 `aes()` 밖에서 고정값인 red 값으로 설정된 코드는 전체 값들이 모두 red 값으로 표현되어 지역적으로 구분되지 않는다.

설정이 가능한 미적요소들은 다음과 같다.

2.2.1. 위치(x , y, xend, yend)

x 와 y 는 기하요소가 표시될 x 축의 위치와 Y 축의 위치 설정에 필요한 데이터 열의 매핑을 설정한다. 일부 선을 그리거나 사각형을 그리는 `geom_segment()` 나 `geom_rect()` 와 같은 기하요소 함수에서는 x, y 부터 시작하여 xend, yend 까지 기하요소를 그린다. 다른 미적요소와는 달리 x, y, xend, yend 는 매핑과 고정값 설정시 모두 반드시 `aes()` 함수안에서 사용되야 한다.

2.2.2. color, colour

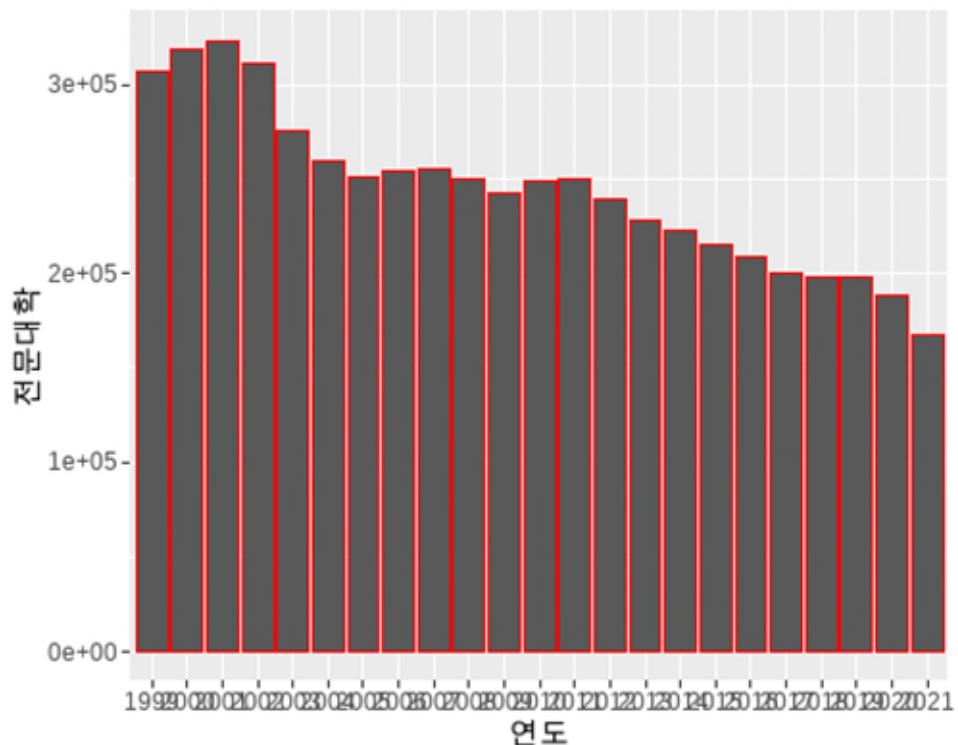
기하요소의 외곽선 색상을 설정한다. R 에서 사용되는 색상은 색상 이름으로 설정하거나 RGB 코드값으로 설정할 수 있다.

R에서 미리 정의된 색상 이름은 총 657개로 `colors()`를 사용하면 확인이 가능하다.

```
## R에서 미리 정의된 색상 이름 출력, 지면 관계상 10개만 출력  
colors() |> head(10)  
  
## [1] "white"      "aliceblue"    "antiquewhite" "antiquewhite1"  
## [5] "antiquewhite2" "antiquewhite3" "antiquewhite4" "aquamarine"  
## [9] "aquamarine1"  "aquamarine2"
```

RGB 색상 코드는 HTML/CSS에서와 같이 RGB 코드를 16진수 값(00에서 FF)을 사용하여 2자리씩 정의하는데 "#" 접두사로 붙은 문자열로 설정한다. 예를 들어 red는 "#FF0000"으로 표현된다.

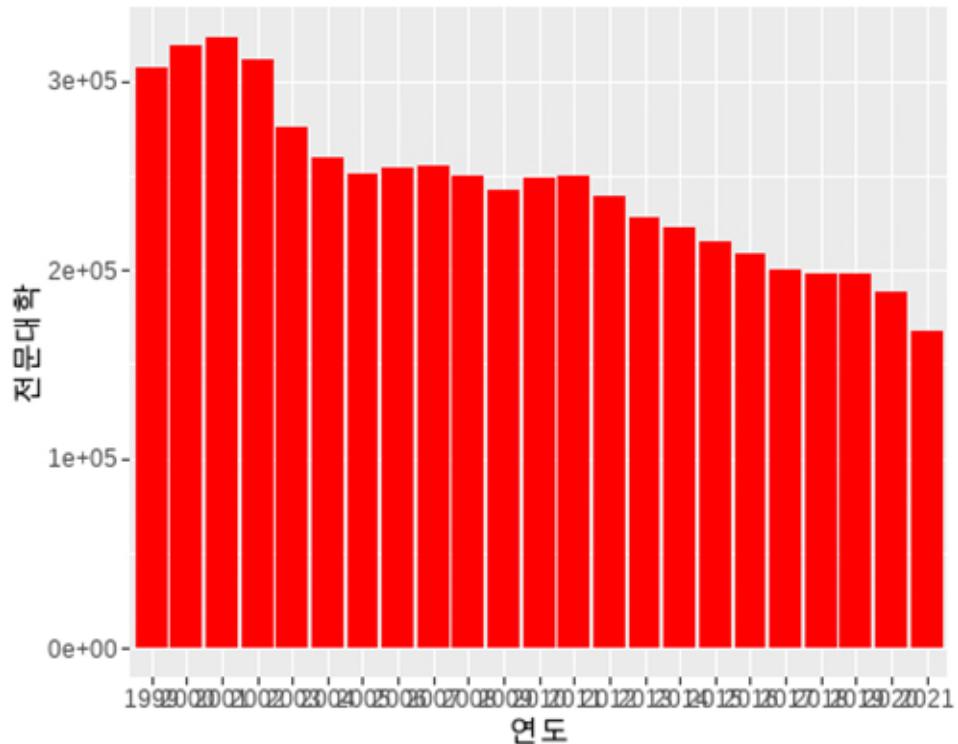
```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 막대의 외곽선 색상을 빨강으로 설정  
df_입학자 |> filter(지역 == '전체') |>  
  ggplot(aes(x = 연도, y = 전문대학)) +  
  geom_col(color = '#ff0000')
```



2.2.3. fill

기하요소의 내부 색상의 설정을 설정한다. 색상의 설정은 `color` 설정과 동일하다.

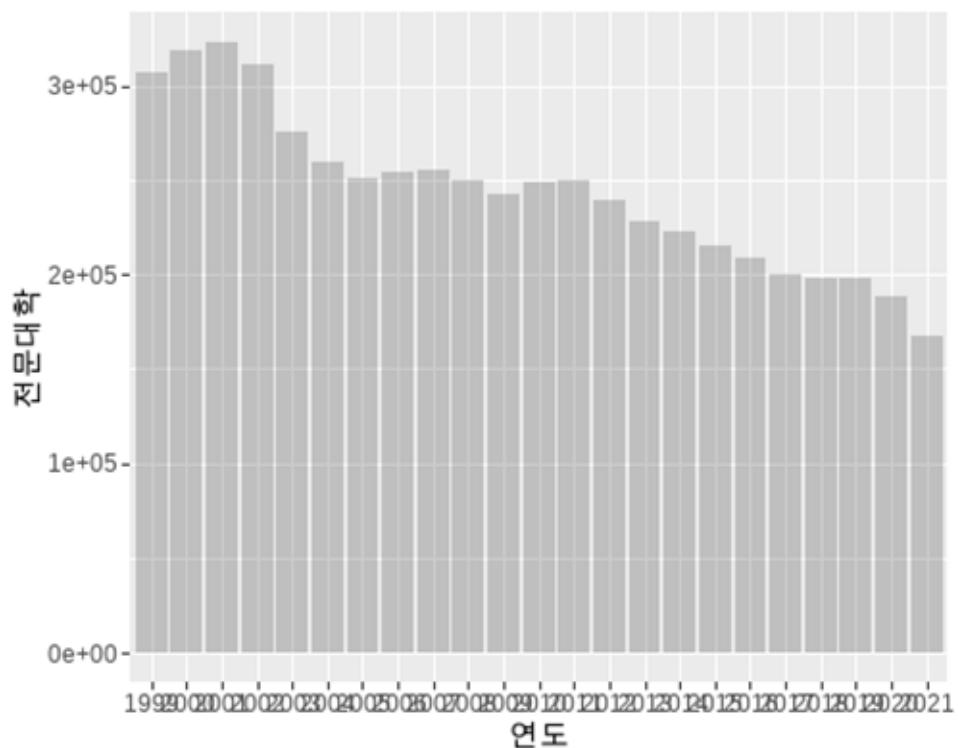
```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 막대의 내부 색상을 빨강으로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_col(fill = '#ff0000')
```



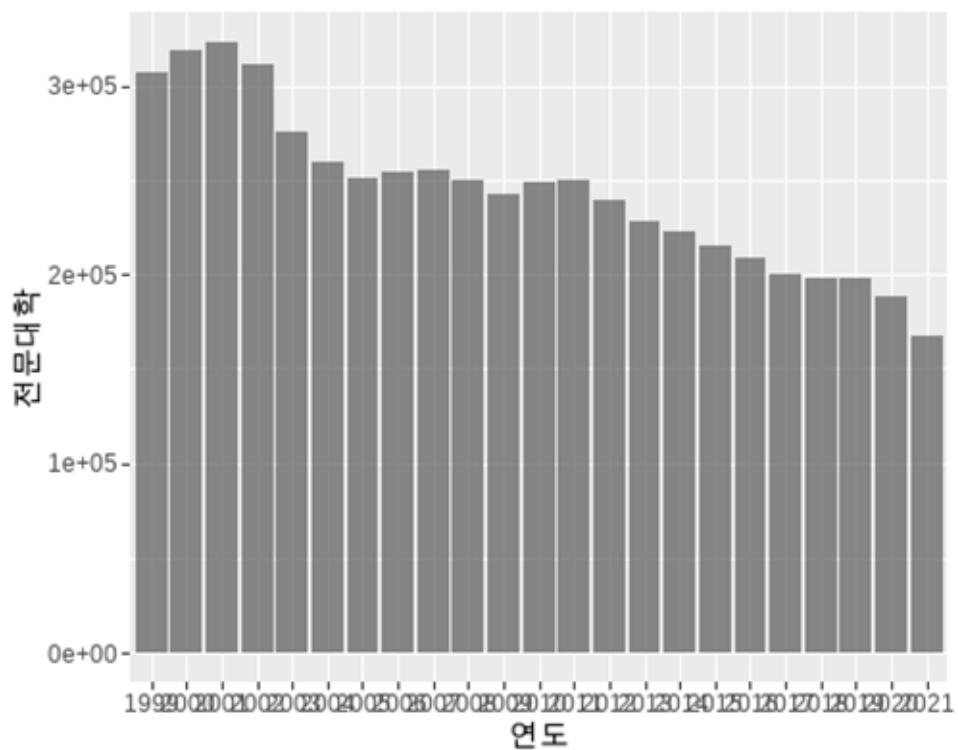
2.2.4. alpha

기하요소의 투명도를 설정한다. `alpha` 는 정수형 수치로 설정하는데 0 부터 1 사이의 값을 가진다. 0에 가까울수록 투명해지고 1에 가까울 수록 불투명해진다.

```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 막대의 투명도를 0.3으로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_col(alpha = 0.3)
```



```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 막대의 투명도를 0.7로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_col(alpha = 0.7)
```



2.2.5. linetype

기하요소 중 선으로 그려지는 기하 요소의 선 타입을 결정한다. R 에서는 총 7 가지의 선 타입을 제공하는데 0 부터 6 까지의 숫자나 선 타입의 이름을 사용하여 설정할 수 있다.

rodash, 6 -----

ngdash, 5 -----

otdash, 4 -----

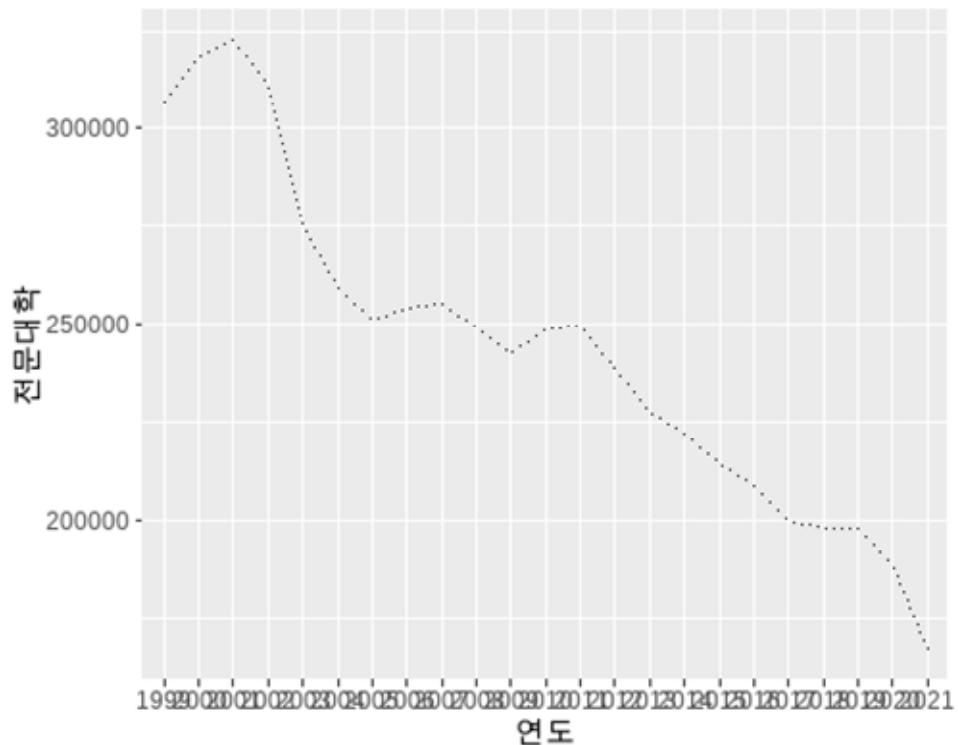
lotted, 3

ashed, 2 -----

solid, 1 -----

blank, 0

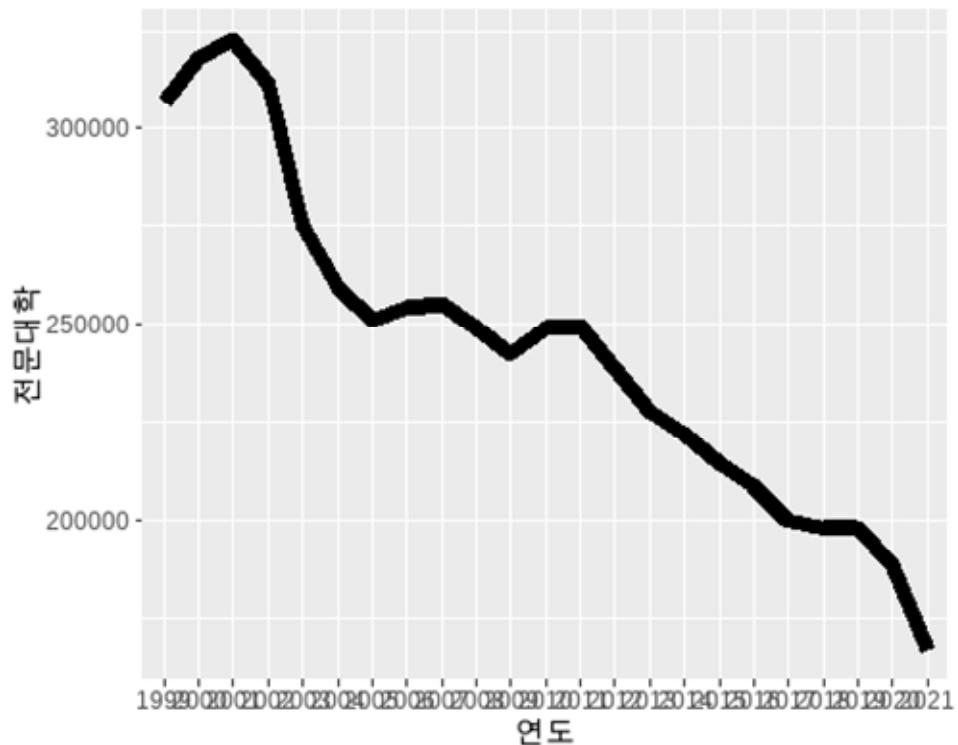
```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 선의 타입을 'dashed'로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_line(aes(group = 1), linetype = 'dotted')
```



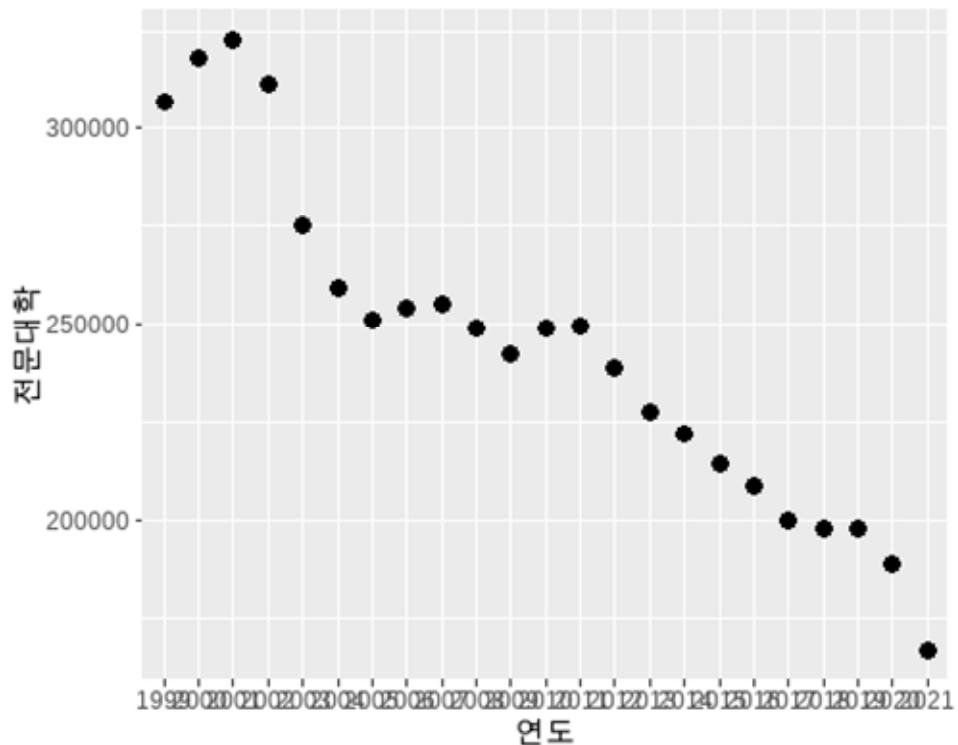
2.2.6. size

`size` 는 기하요소의 크기를 설정한다. 기하요소가 점이면 점의 크기, 선이면 선의 굵기를 설정한다. 점의 크기를 결정할 때는 반지름의 길이를 밀리미터 단위로 지정한다. 또 선의 굵기도 밀리미터 단위로 지정할 수 있다.

```
## df_입학자의 지역이 '전체' 인 데이터를 시각화하는데 선의 크기를 3 으로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_line(aes(group = 1), size = 3)
```

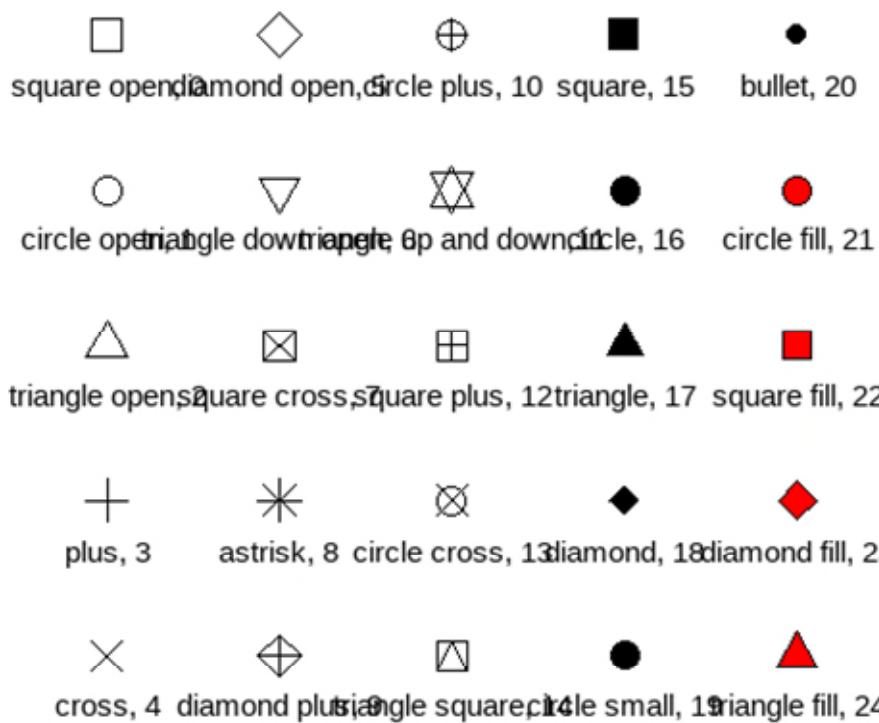


```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 점의 크기를 3으로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_point(size = 3)
```



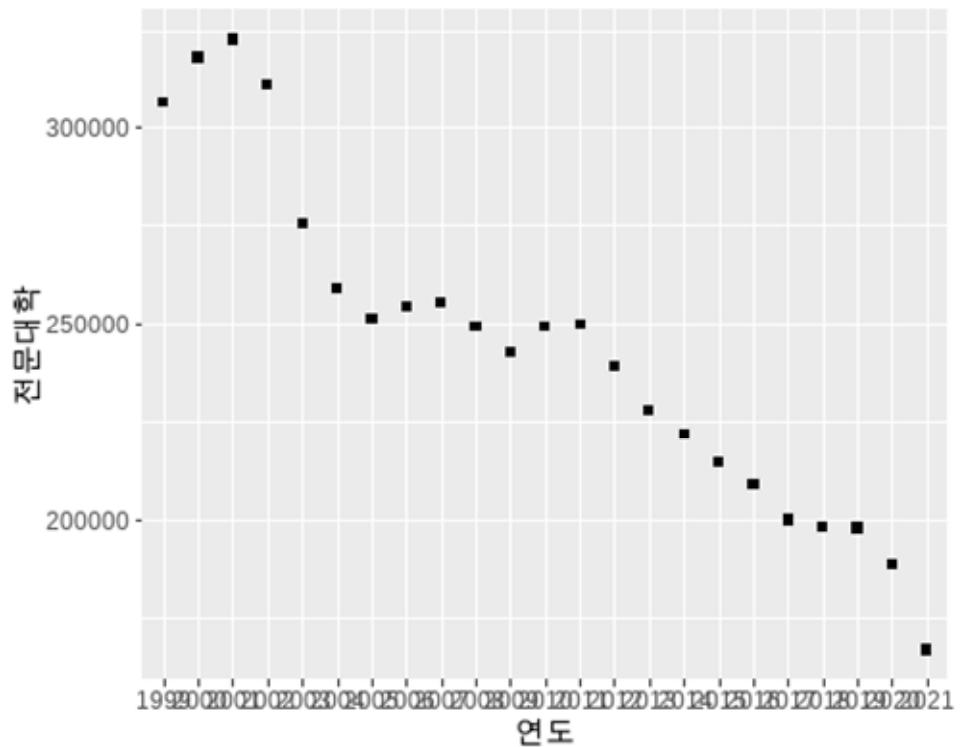
2.2.7. shape

`shape` 는 기하요소가 점일때 점의 모양을 표현하는 미적요소이다. R 에서는 점을 표현하는 기하요소를 25 가지 지원한다. 앞선 라인타입과 유사하게 번호로 표현할 수도 있고 사전에 정의된 `shape` 이름을 사용할 수도 있다. 다음은 R 에서 제공하는 `shape` 이다.

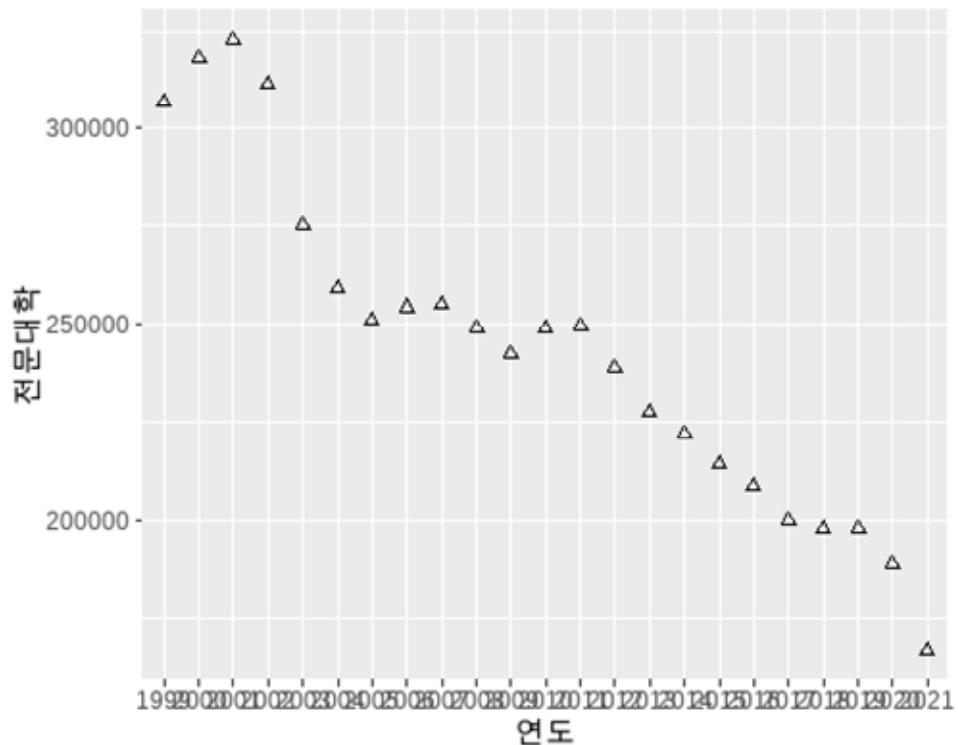


위의 **shape** 를 실제 사용하는 예제는 다음과 같다.

```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 점의 모양을 15로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_point(shape = 15)
```



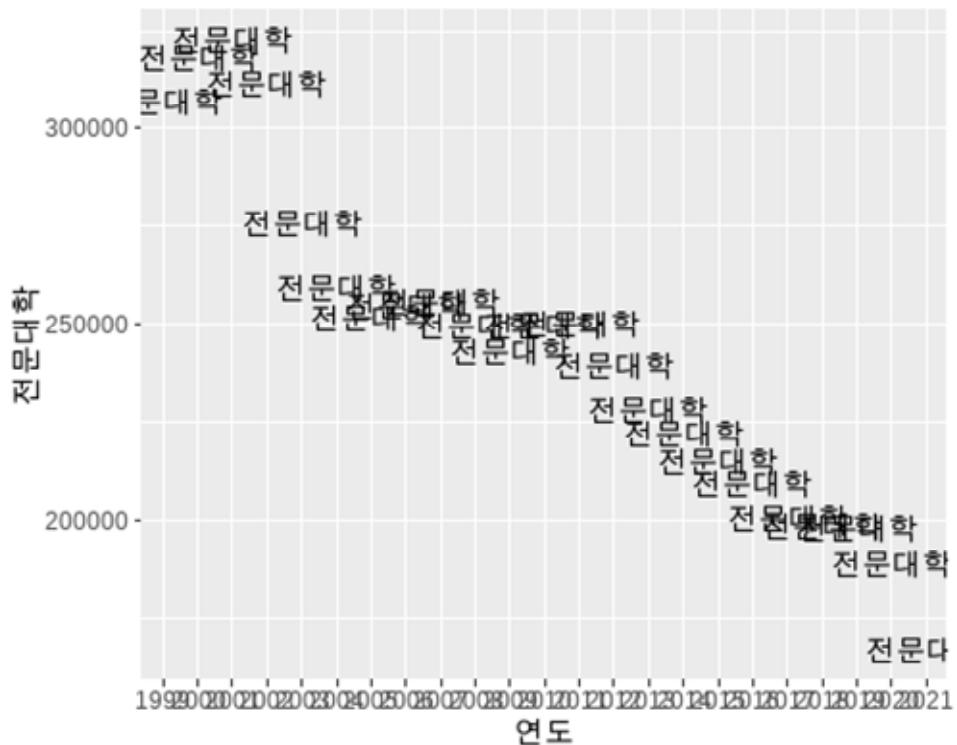
```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 점의 모양을 'triangle open'으로 설정  
df_입학자 |> filter(지역 == '전체') |>  
  ggplot(aes(x = 연도, y = 전문대학)) +  
  geom_point(shape = 'triangle open')
```



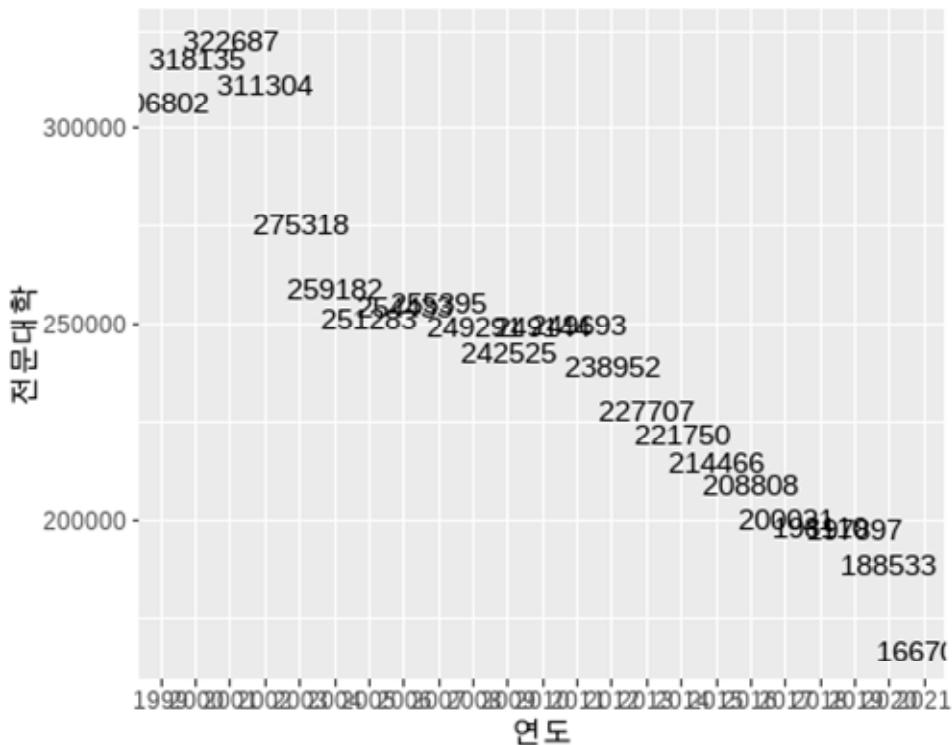
2.2.8. label

`label` 은 텍스트로 표현되는 기하요소를 말한다. 데이터의 위치에 대한 수치값을 표현한다거나 데이터의 분류에 대한 문자를 표현할 때 사용되는 미적요소이다. `label` 은 주로 텍스트에 관련된 기하요소인 `geom_text()`나 `geom_label()`에서 주로 사용된다.

```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 라벨을 '전문대학'으로 설정
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_text(label = '전문대학')
```



```
## df_입학자의 지역이 '전체'인 데이터를 시각화하는데 라벨을 '전문대학' 옆에 매팅
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_text(aes(label = 전문대학))
```



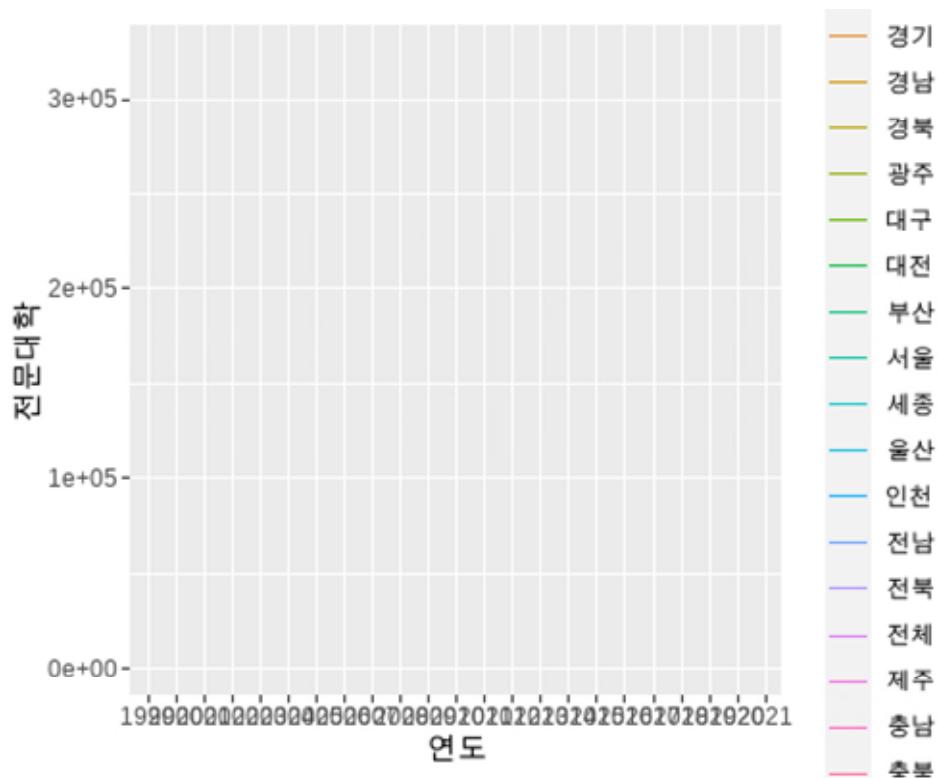
2.2.9. group

기하요소는 개별(individual) 기하요소와 집합(collective) 기하요소로 구분될 수 있다. 개별 기하요소는 각각의 관측이(행)을 각각의 기하요소로 매핑해서 표현하지만 집합 기하요소는 여러개의 관측치를 하나의 기하요소로 표현한다.³ 집합 기하요소는 통계적인 요약이 필요한 Box Plot이나 한번에 여러개의 변량이 연결되어 표현되야하는 도형(polygon), 선, 세그먼트 등에서 연결되는 구분이 설정되어야한다. 이를 표현하기 위해 사용되는 미적요소가 **group** 이다. 설명한대로 **group** 은 직접적으로 표현되는 미적요소는 아니지만 미적요소가 적용되는 범위를 설정하기 위해 사용되는 미적요소이다.

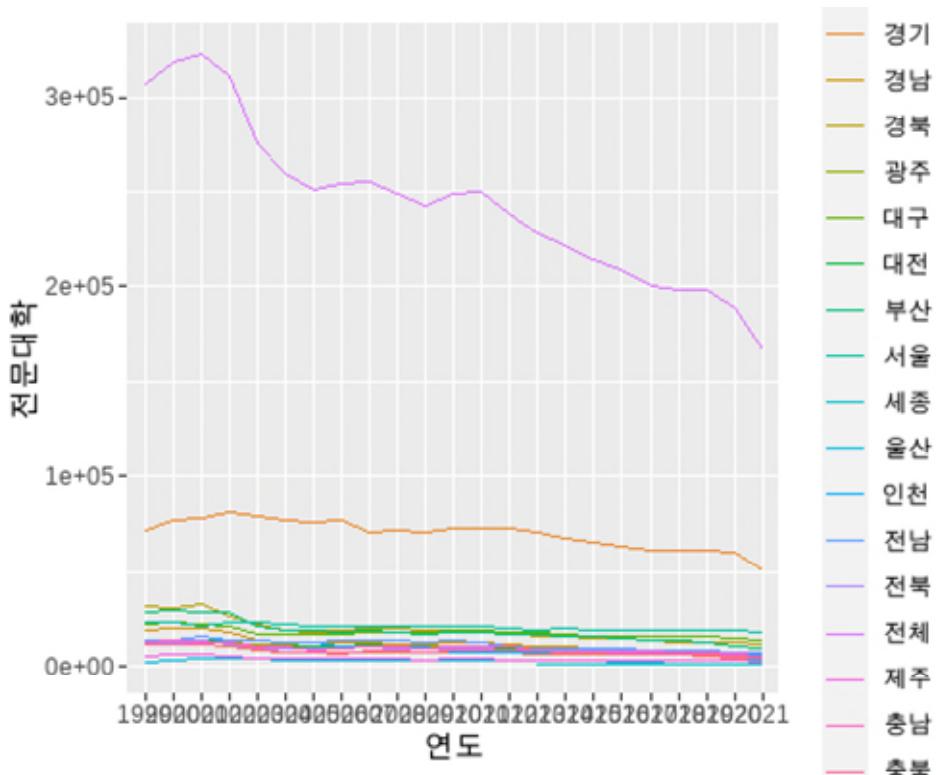
아래의 코드를 살펴보자.

```
## df_입학자 중 전문대학 입학생을 선 그래프로 시각화하는데 지역별 구분이 되지 않음
df_입학자 |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_line(aes(color = 지역))
```

³ Hadley Wickham, ggplot2:Elegant Graphics for Data Analysis 2nd edition, P 46, Springer, 2016.



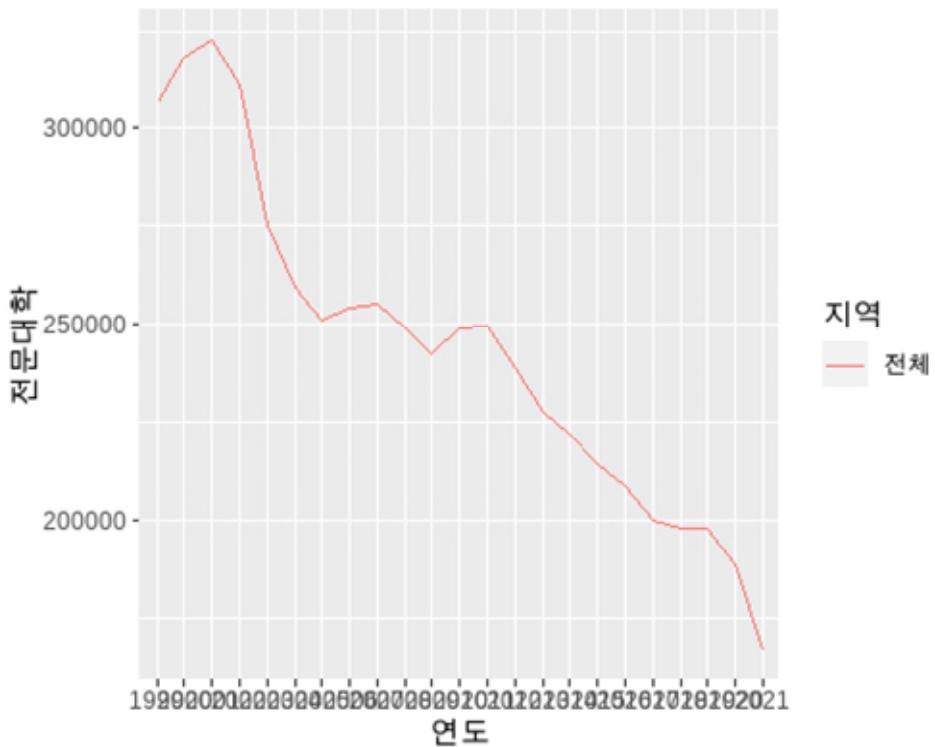
```
## df_입학자 중 전문대학 입학생을 선 그래프로 시각화하는데 group 을 지역으로 설정함으로
써 지역별 구분이 가능
df_입학자 |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_line(aes(color = 지역, group = 지역))
```



위의 코드에서 앞선 코드는 `group` 을 지정하지 않고 `color` 만 `df_입학자` 데이터의 지역 열에 매핑하였다. 하지만 선으로 표현될 미적요소는 지역별로 그루핑되어야 정상적으로 표현되는데 단지 `color` 만의 매핑으로는 정상적으로 표현되지 않는다. 따라서 `geom_line()` 으로 여러개의 선이 표현되는 선 그래프를 표현하기 해서는 반드시 `group` 으로 그루핑되는 열을 지정해야 정상적으로 표현된다.

그렇다면 단 하나의 선만이 있는 선 그래프는 어떻게 될까? 단 하나의 선이기 때문에 `group` 이 필요없을까? 답은 그렇지 않다. 다만 단 하나의 선이 있는 경우에는 `group = 1` 로 선언함으로써 열을 매핑하지 않고 사용할 수도 있다.

```
## df_입학자 중 지역이 전체에 해당하는 전문대학 입학생을 선 그래프로 시각화하는데 지역이 하나뿐이므로 group 을 1로 설정함으로써 선그래프를 그릴 수 있음
df_입학자 |> filter(지역 == '전체') |>
  ggplot(aes(x = 연도, y = 전문대학)) +
  geom_line(aes(color = 지역, group = 1))
```



2.3. 기하요소

기하요소는 `ggplot()`로 생성된 초기화 `ggplot` 객체에 데이터를 표현하는 방법을 지정하는 요소이다. 2.기하요소에는 여러가지가 있지만 우리가 흔히 생각하는 것은 점(Point), 선(Line), 막대(Bar, Col) 등이 대표적이다.

기하요소를 생성하기 위해서는 `geom_*`() 함수를 사용하고 각각의 `geom_*`() 함수를 호출할 때마다 각각의 기하요소 레이어가 생성되고 이 레이어들이 계속 겹쳐서 그려짐으로써 데이터 시각화가 진행된다. 기하요소를 위한 `geom_*`()의 주요 함수는 각각의 함수에 따라 선언되는 시각화 요소들이 다르지만 데이터, 미적요소 매핑, 기하요소 지정, 통계 변환, 위치 조정 등이 선언된다.

기하요소를 선택할 때는 각각의 기하요소에 따라 표현되는 값의 제한이 있다. 예를 들어 막대 그래프의 경우 Y 축은 연속된 정수값이 표현되는 것이 가능하지만 X 축에는 연속된 정수값이 아닌 값의 구별이 가능한 이산 값(discrete value)가 와야한다. 그래야 분리된 하나의 이산값에 하나의 막대가 표현될 수 있다. 이렇게 표현하고자 하는 값의 종류에 따라 적합한 기하요소를 선택하여야 한다. 값의 종류는 연속된 값(continuous value), 분리된 이산값(discrete value) 나 팩터(factor)의 여부, 일변량, 다변량의 여부 등으로 나눌 수 있다.

2.3.1. 일변수(One Variable) 데이터 시각화

일변수 데이터는 수치형 데이터 열 하나를 의미한다. `ggplot()` 는 보통 X 축과 Y 축의 2 차원 표현이 기본이기 때문에 하나의 데이터 열만이 정의되면 나머지 하나의 데이터는 자동적으로 결정되어야 한다. 이렇게 자동적으로 결정되는 데이터는 보통 면적(Area), 밀도 분포(Density), 도수 분포(Histogram) 등이다. 일변수 데이터 시각화는 일변수로 설정되는 데이터가 연속성 수치 데이터인지 이산성 데이터인지에 따라 구분하여 기하요소 함수를 사용할 수 있다.

2.3.1.1. 연속형 수치 데이터

2.3.1.1.1. `geom_histogram()`

`geom_histogram` 은 도수분포표를 그리는 기하요소 함수이다. 도수분포는 초등학교 때 배우는 가장 기본적인 막대그래프로 각각의 변수 변량에 따른 데이터의 개수를 표현하는 시각화 방법이다. 변수 변량에 따른 데이터의 개수를 표현하기 때문에 X 축 데이터만 설정하면 데이터를 자동적으로 분석하여 X 축에 매핑된 변수의 변량별로 데이터 개수를 산출하게 되고 이 개수를 막대 그래프로 표현하게 된다. 따라서 도수분포는 막대그래프에 속하는 종류 중 하나일 뿐이다.

앞에서 막대그래프는 연속된 수치값이 아닌 분리된 이산값이 X 축에 매핑되어야 한다고 설명는데 연속된 일변량 기하요소에 `geom_histogram()` 이 속하는 것은 왜일까?

`geom_histogram()` 은 연속된 수치값을 X 축에 매핑한다. 하지만 내부적으로 적절한 단위로 전체 X 값을 분리하여 이산값으로 만들어 준 후에 막대그래프를 생성해 준다. 이 과정이 통계요소이고 `geom_histogram()` 에서 사용되는 유일한 통계요소은 연속된 값을 층화하여 구간하는 방법인 binning 이다. 이처럼 자동적으로 계산되는 binning 때문에 연속된 일변량 기하요소에 속하고 이를 위해 통계요소를 'bin'으로 사용한다.

`geom_histogram` 의 사용법과 주요 매개변수는 다음과 같다.

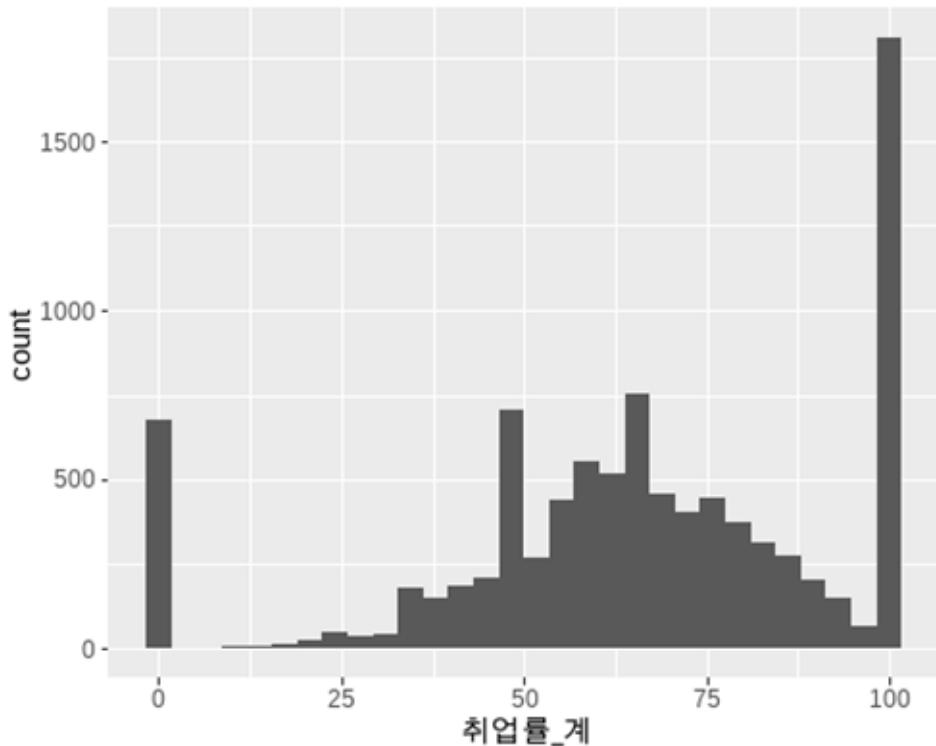
```
geom_histogram(mapping = NULL, data = NULL, stat = "bin", position = "stack", bins, binwidth, ...)  
- mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용  
- data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용  
- stat : 시각화에 적용될 통계요소, 기본값은 'bin'  
- position : 시각화에 적용될 위치요소, 기본값은 'stack'  
- bins : X 축을 나누는 bin 의 개수 설정
```

- `binwidth` : X 축을 나누는 bin의 너비 설정, 숫자벡터를 사용할 수 있다. (`bin`과 `binwidth`은 동시에 사용될 수 없다)

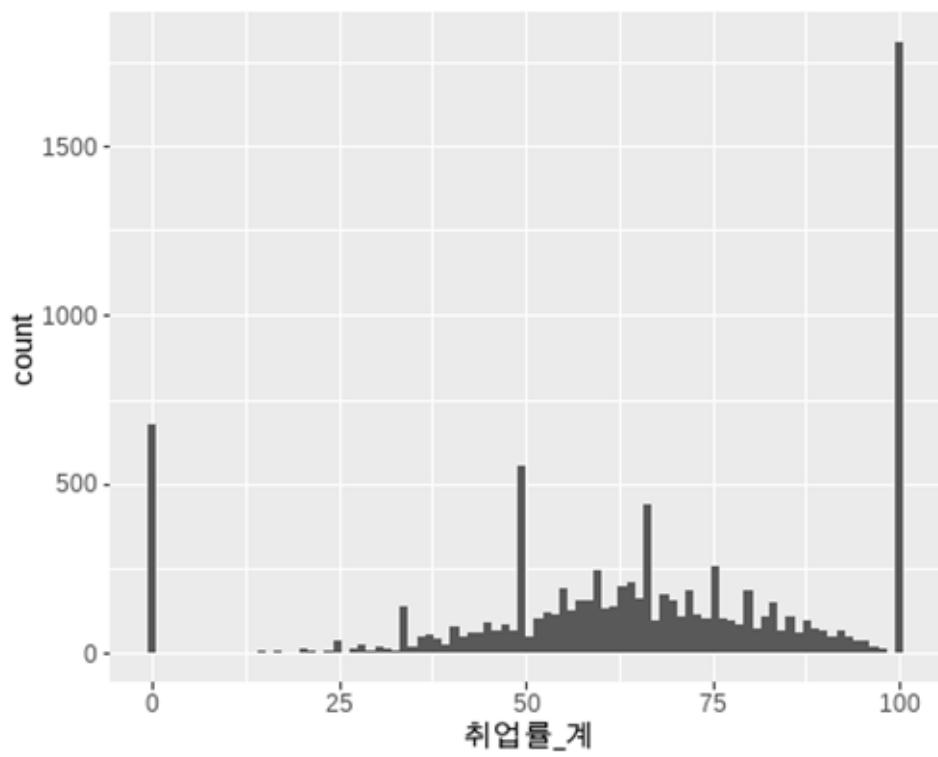
`geom_histogram()`에 매핑될 수 있는 미적요소은 위치(x, y), alpha, color, fill, linetype, size 등이다.

```
## df_취업통계를 ggplot 객체로 생성하고 p_histogram에 저장
p_histogram <- df_취업통계 |>
  ggplot()

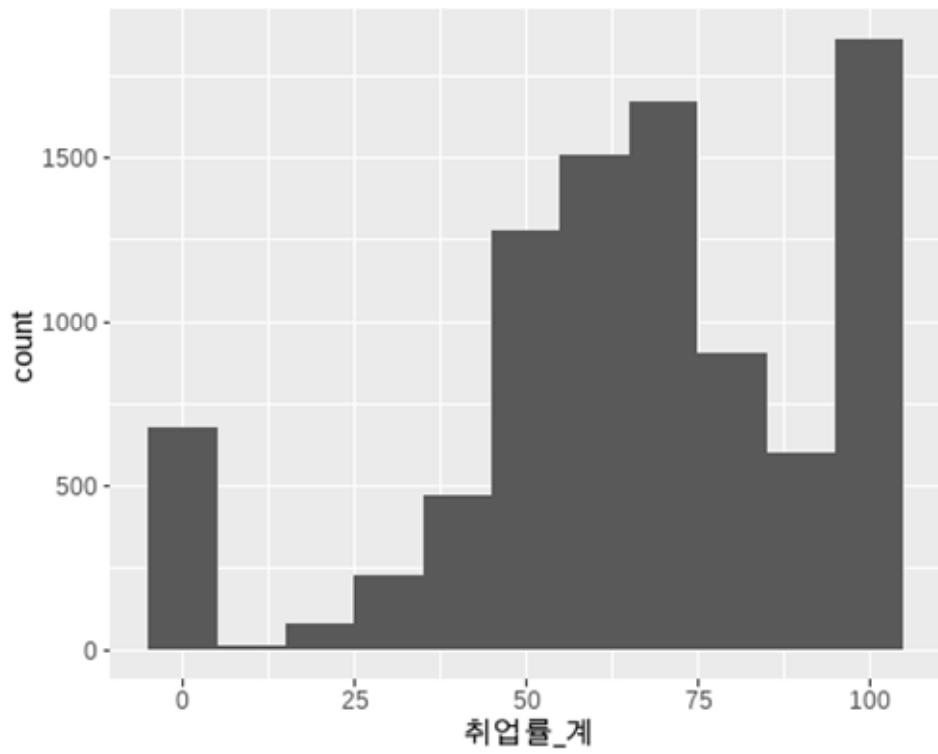
## p_histogram에 x 축을 '취업률_계' 열로 맵핑, binning 옵션을 주지 않았으므로 bins = 30 이 기본값으로 설정한 geom_histogram 레이어를 생성
p_histogram +
  geom_histogram(aes(x = 취업률_계))
```



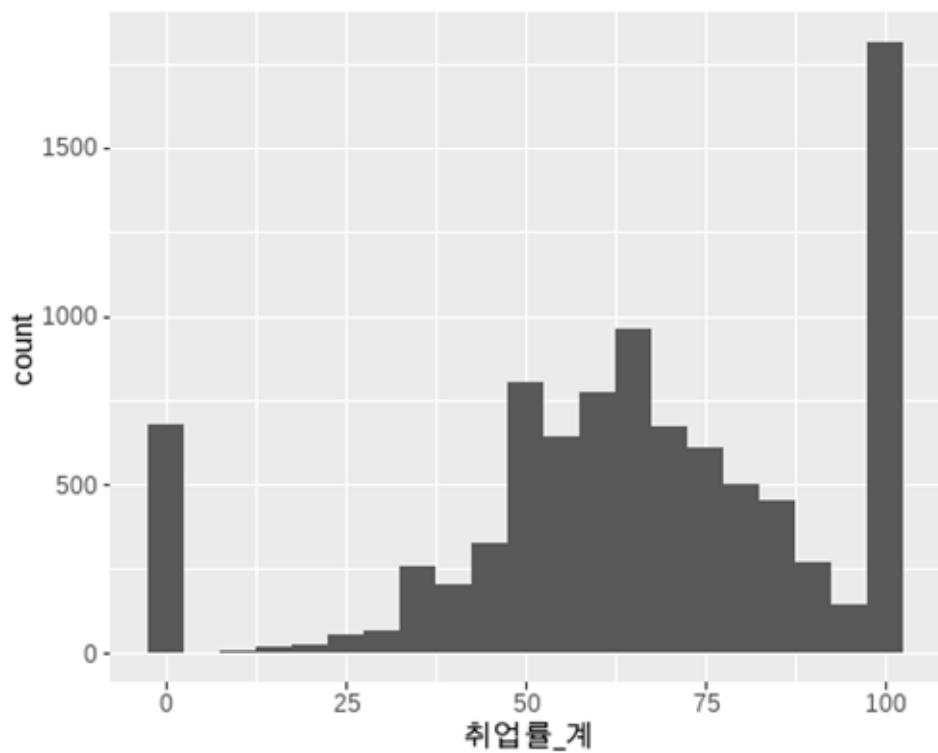
```
## p_histogram에 x 축을 '취업률_계' 열로 맵핑, bins = 90 으로 설정한 geom_histogram 레이어를 생성
p_histogram +
  geom_histogram(aes(x = 취업률_계), bins = 90)
```



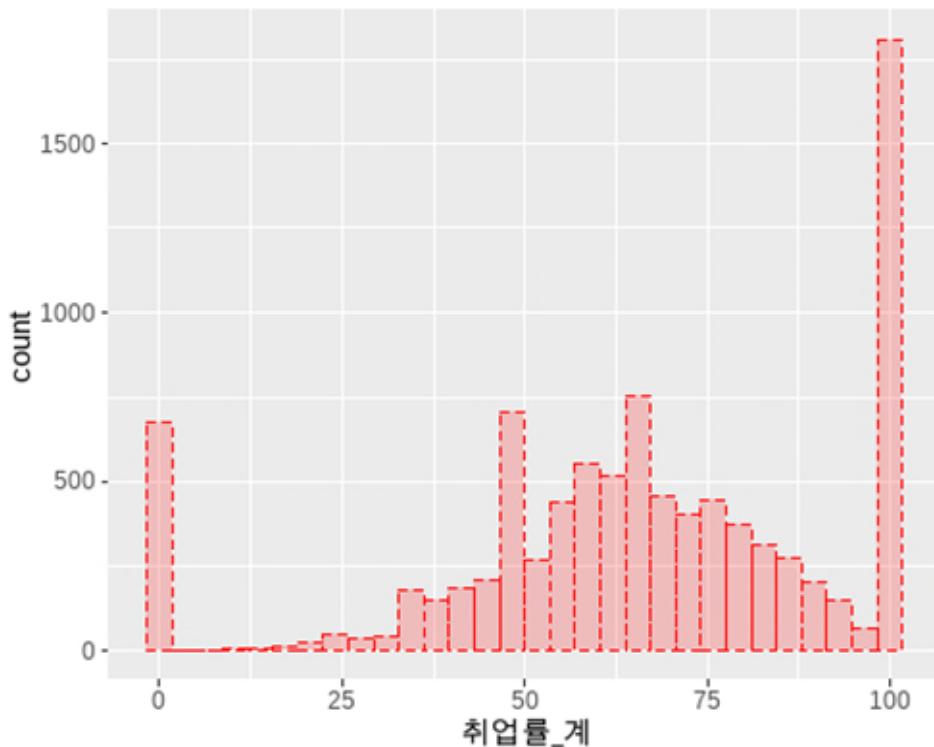
```
## p_histogram 에 x 축을 '취업률_계' 열로 맵핑, binwidth = 10 으로 설정한 geom_histogram 레이어를 생성
p_histogram +
  geom_histogram(aes(x = 취업률_계), binwidth = 10)
```



```
## p_histogram 에 x 축을 '취업률_계' 열로 맵핑, binwidth = 5 으로 설정한 geom_histogram 레이어를 생성
p_histogram +
  geom_histogram(aes(x = 취업률_계), binwidth = 5)
```



```
## p_histogram에 x 축을 '취업률_계' 열로 맵핑, 각각의 미적요인을 설정한 geom_histogram 레이어를 생성
p_histogram +
  geom_histogram(aes(x = 취업률_계), color = 'red', fill = 'red', alpha = 0.2, linetype = 2)
```



2.3.1.1.2. *geom_freqpoly()*

`geom_freqpoly()` 는 하나의 데이터 열에 기록된 연속형 수치 데이터를 표현하는데 사용되는 기하요소 함수이다. 앞선 `geom_histogram()` 은 막대를 사용하여 데이터의 수치를 표현해지만 `geom_freqpoly()` 는 데이터 값들을 연결한 다각형을 사용하여 데이터의 수치를 표현한다. `geom_freqpoly()` 도 `geom_histogram()` 과 같이 binning 을 기본적으로 사용하기 때문에 통계요소를 'bin'이 기본적으로 적용된다. 만약 다각형을 조금 부드럽게(smoothing) 하려면 통계요소를 'density'로 설정하는데 이 경우는 `geom_density()` 와 동일한 결과가 나온다.

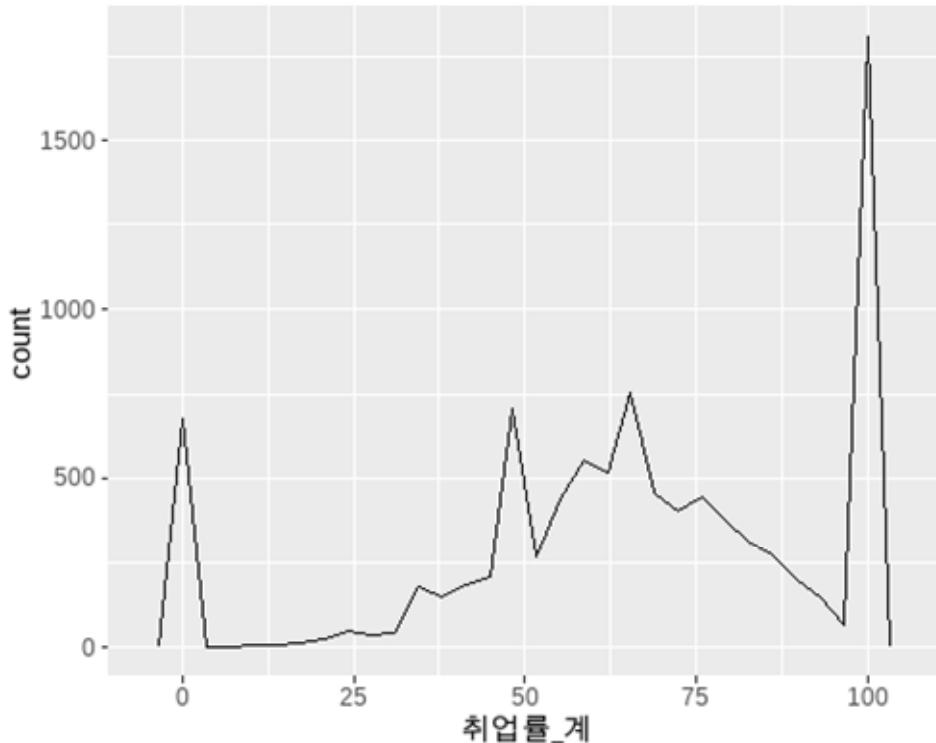
```
geom_freqpoly(mapping = NULL, data = NULL, stat = "bin", position = "identity",
  na.rm = FALSE, show.legend = NA, ...)
  - mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용
  - data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
  - stat : 시각화에 적용될 통계요소, 기본값은 'bin'
  - position : 시각화에 적용될 위치요소, 기본값은 'identity'
  - na.rm : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
  - show.legend : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)
```

`geom_freqpoly()`에서 사용이 가능한 미적요소는 위치(x, y), alpha, color, linetype, size, group 이다.

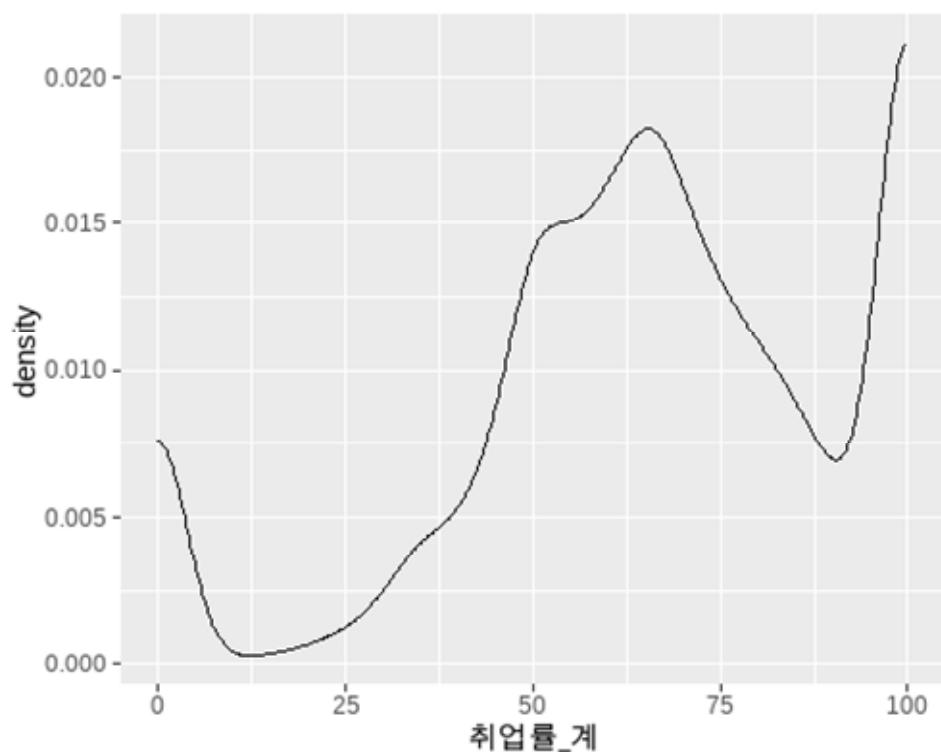
`geom_freqpoly()`를 사용한 데이터 시각화는 다음과 같다.

```
## df_취업통계를 ggplot 객체로 생성하고 p_freqpoly에 저장
p_freqpoly <- df_취업통계 |>
  ggplot()

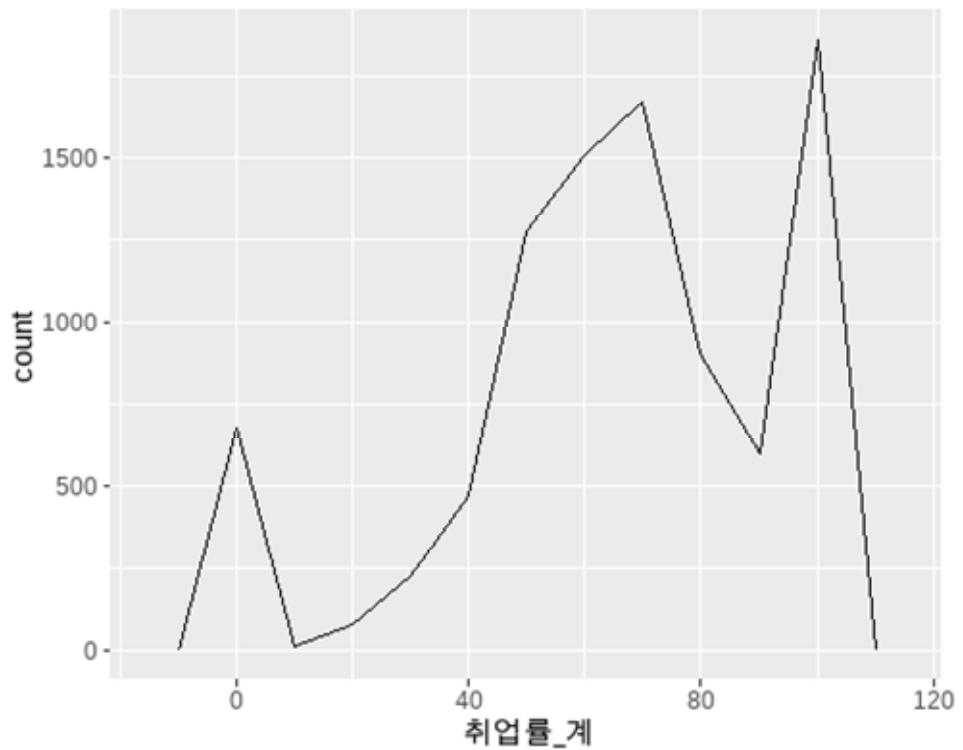
## p_freqpoly에 x 축을 '취업률_계' 열로 매팅, bins = 30으로 설정한 geom_freqpoly 레이어를 생성
p_freqpoly +
  geom_freqpoly(aes(x = 취업률_계), bins = 30)
```



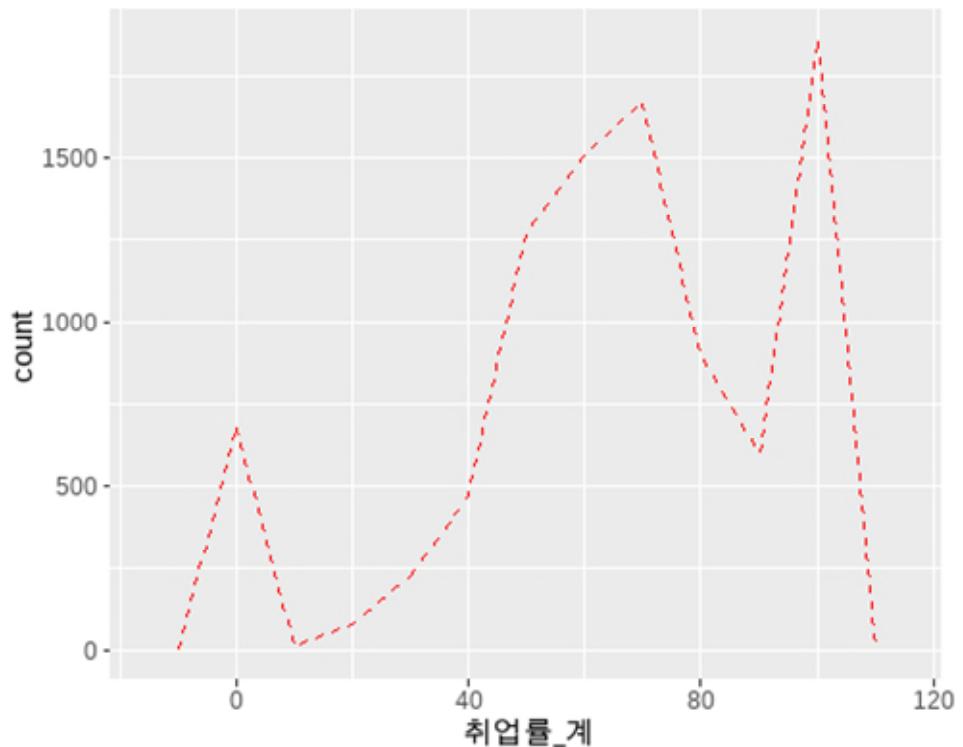
```
## p_freqpoly에 x 축을 '취업률_계' 열로 매팅, bins = 30으로 설정하는데 통계요소를 'density'로 설정한 geom_freqpoly 레이어를 생성
p_freqpoly +
  geom_freqpoly(aes(x = 취업률_계), stat = 'density', bins = 30)
```



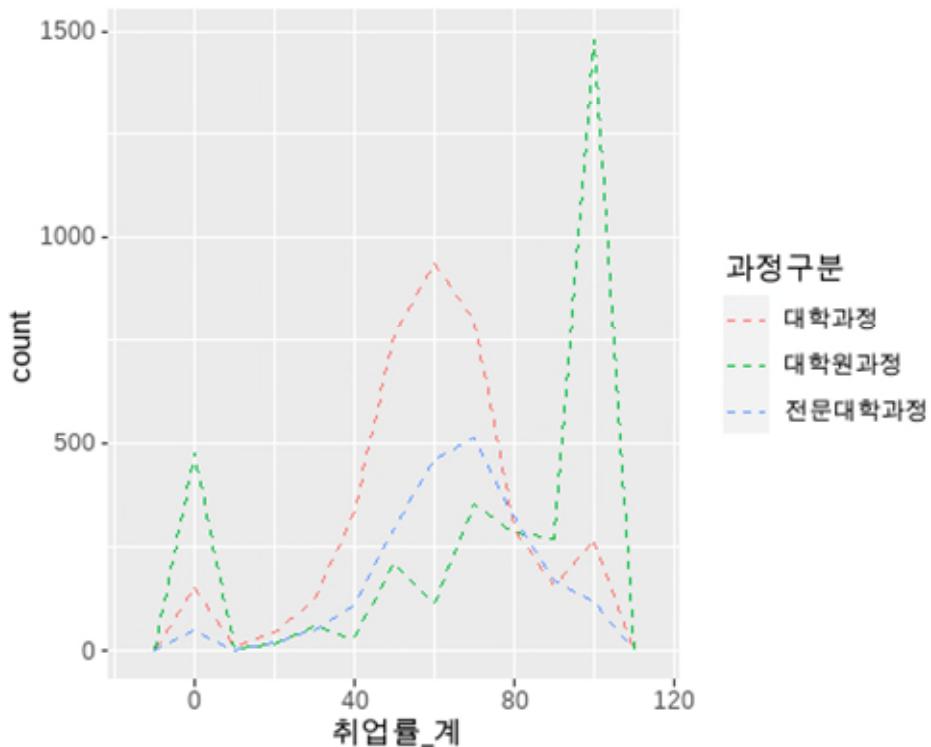
```
## p_freqpoly 에 x 축을 '취업률_계' 열로 매팅, binwidth = 10 으로 설정한 geom_freqpoly  
레이어를 생성  
p_freqpoly +  
  geom_freqpoly(aes(x = 취업률_계), binwidth = 10, rm.na = TRUE)
```



```
## p_freqpoly 에 x 축을 '취업률_계' 열로 맵핑, binwidth = 10 으로 설정하고 미적요소를  
설정한 geom_freqpoly 레이어를 생성  
p_freqpoly +  
  geom_freqpoly(aes(x = 취업률_계), binwidth = 10, color = 'red', linetype = 2)
```



```
## p_freqpoly 에 x 축을 '취업률_계' 열로 맵핑, group 과 color 를 과정구분으로 맵핑, bin  
width = 10 으로 설정한 geom_freqpoly 레이어를 생성  
p_freqpoly +  
  geom_freqpoly(aes(x = 취업률_계, group = 과정구분, color = 과정구분), binwidth =  
  10, linetype = 2)
```



2.3.1.1.3. *geom_density()*

geom_density() 도 하나의 데이터 열에 기록된 연속형 수치 데이터를 표현하는데 사용되는 기하요소 함수인데 데이터의 분포에 따른 확률분포함수를 표현하는 함수이다. 데이터 값으로 주어진 연속된 수치 데이터를 확률밀도함수에 대입하여 계산된 연속확률분포를 시각화한다. 따라서 X 값에 따라 계산된 연속확률분포값은 확률값이기 때문에 1 보다 작은 값들이고 이 값들이 연결되어 표현된다. *geom_density()* 를 사용하여 연속확률분포를 시각화할 수 있다. 따라서 통계요소가 앞선 두 함수와는 달리 'density'가 기본값으로 설정된다.

```
geom_density(mapping = NULL, data = NULL, stat = "density", position = "identity",
na.rm = FALSE, show.legend = NA, ...)
```

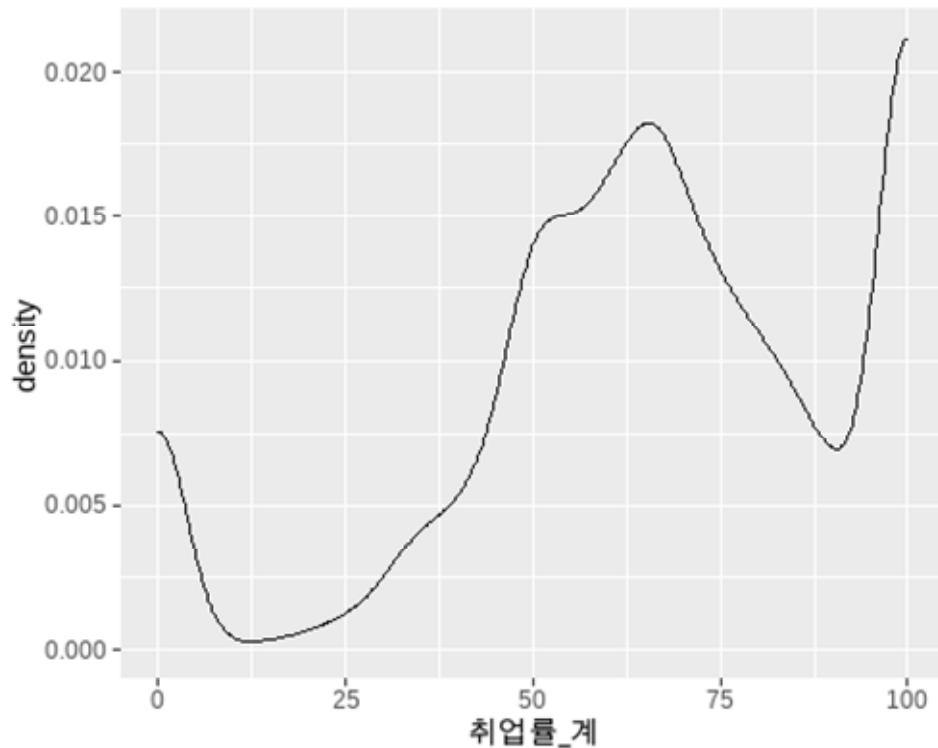
- **mapping** : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

- **data** : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- **stat** : 시각화에 적용될 통계요소, 기본값은 'density'
- **position** : 시각화에 적용될 위치요소, 기본값은 'identity'
- **na.rm** : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- **show.legend** : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

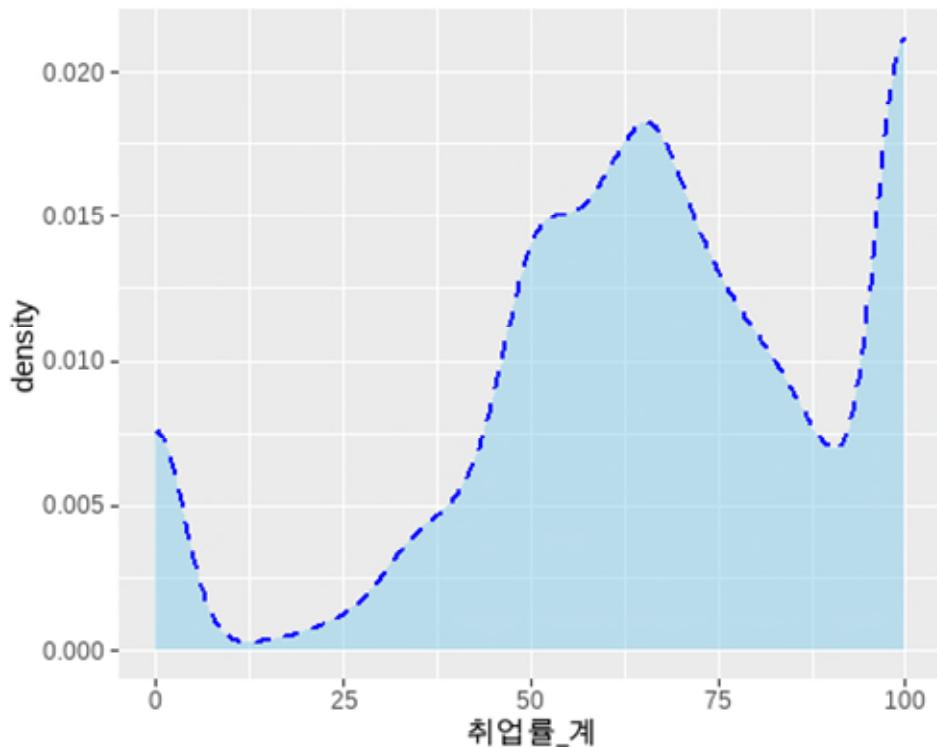
`geom_density()`에서 사용할 수 있는 미적요소는 위치(x, y), alpha, color, fill, linetype, size 등이다.

```
## df_취업통계를 ggplot 객체로 생성하고 p_density에 저장
p_density <- df_취업통계 |>
  ggplot()

## p_density 객체에 x 축을 '취업률_계' 열로 맵핑한 geom_density 레이어를 생성
p_density +
  geom_density(aes(x = 취업률_계))
```



```
## p_density 객체에 x 축을 '취업률_계' 열로 맵핑, 미적요소 설정한 geom_density 레이어를
## 생성
p_density +
  geom_density(aes(x = 취업률_계), color = 'blue', fill = 'skyblue', linetype =
2, size = 1, alpha = 0.5)
```



2.3.1.2. 이산(Discrete) 데이터 : `geom_bar()`

앞 절에서 설명한 `geom_histogram()`, `geom_freqpoly()`, `geom_density()` 는 연속된 수치값에 대한 일변량 시각화 기하요소함수였다. 그렇다면 팩터와 같은 구분되고 분리되는 이산된 일변량 시각화는 어떻게 시각화되는가? 이산된 일변량 데이터를 위한 기하요소 함수는 `geom_bar()` 가 유일하다. 사실 `geom_bar()` 는 뒤에서 설명될 `geom_col()` 과 거의 유사한 결과를 내는 막대 그래프 시각화이다. 다만 `geom_bar()` 는 앞선 연속된 일변량 데이터 시각화와 같이 하나의 데이터만 제공되면 해당 데이터의 빈도를 자동적으로 계산하여 변수의 변량에 따른 데이터의 빈도를 막대 그래프 형태로 시각화한다. 이는 또 `geom_histogram()` 과 유사하게 나타나지만 x 축으로 제공되는 데이터가 연속형 수치 데이터인가 이산형 데이터인가에 따라 차이가 있다. 또 x 축에 매핑되는 데이터가 이산형 데이터이기 때문에 꼭 수치 데이터가 아닐 수 있다.

```
geom_bar(mapping = NULL, data = NULL, stat = "count", position = "stack", width = NULL,
na.rm = FALSE, show.legend = NA, ...)
```

- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

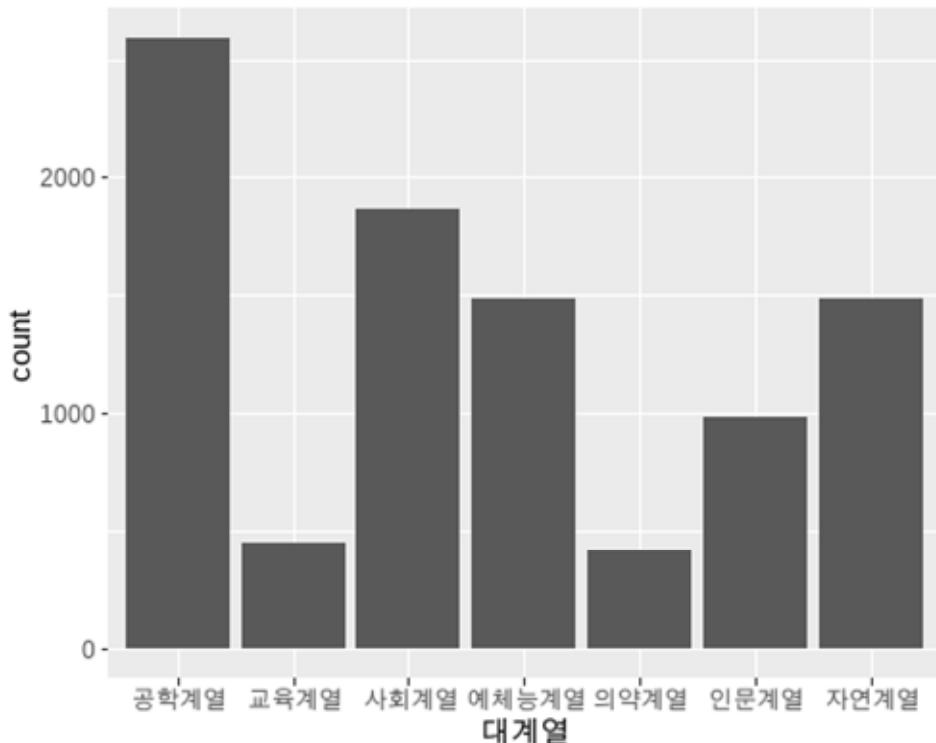
- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 'count'

- `position` : 시각화에 적용될 위치요소, 기본값은 '`stack`'
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(`TRUE/FALSE`)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(`TRUE/FALSE`)

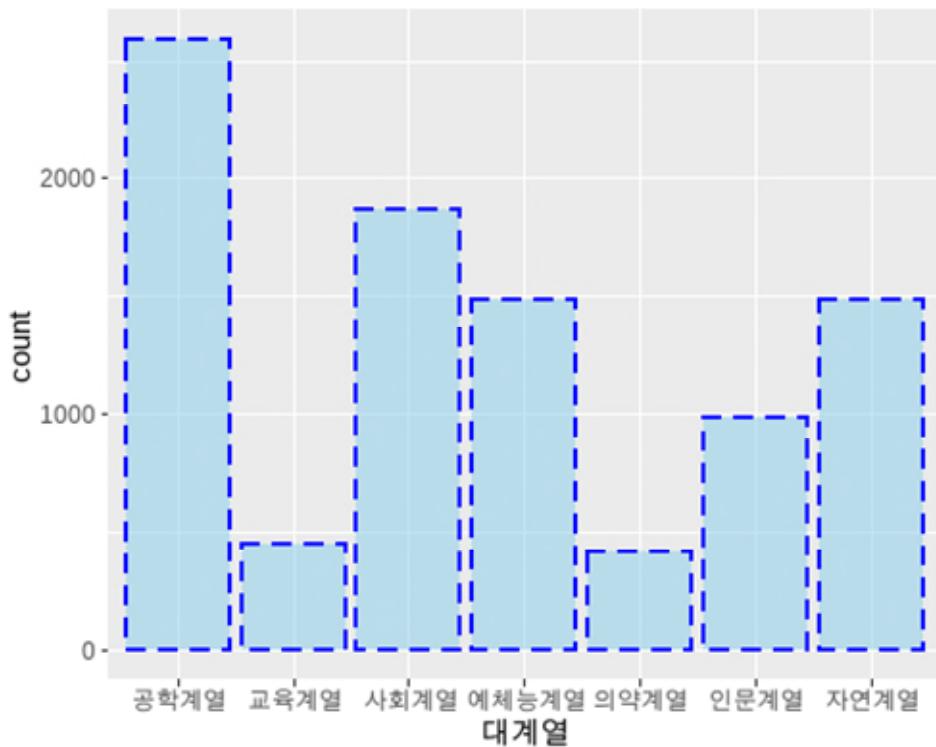
`geom_bar()`에서 사용이 가능한 미적요소는 `x, alpha, color, fill, linetype, size, weight` 가 있다.

```
## df_취업통계를 ggplot 객체로 생성하고 p_bar에 저장
p_bar <- df_취업통계 |>
  ggplot()

## p_bar 객체에 x 축을 대계열로 맵핑한 geom_bar 레이어 생성
p_bar +
  geom_bar(aes(x = 대계열))
```



```
## p_bar 객체에 x 축을 대계열로 맵핑하고 미적요소를 설정한 geom_bar 레이어 생성
p_bar +
  geom_bar(aes(x = 대계열), color = 'blue', fill = 'skyblue', linetype = 2, size = 1, alpha = 0.5)
```



2.3.2. 이변수(Two Variable) 데이터 시각화

앞서 언급한 바와 같이 `ggplot` 객체는 보통 X, Y 두 개의 축으로 표현되는 2 차원 시각화를 기본으로 하고 있다. 따라서 변수 두개를 사용한 데이터 시각화가 일반적 형태의 시각화이다. 변수 두개를 사용한 데이터 시각화도 일변수 데이터 시각화와 마찬가지로 시각화해야 할 데이터가 연속형 수치 데이터인지 이산형 데이터인지에 따라 사용하는 기하요소의 종류와 함수가 달라진다.

2.3.2.1. 2 개의 연속형 수치 변수

X 축과 Y 축에 매핑되어 시각화해야하는 두개의 변수가 모두 연속형 수치 변수인 경우에는 X 축과 Y 축의 매핑에 따라 위치를 표시해 주는 시각화가 대부분이다. 보통 데이터의 분포를 확인해야 하는 경우와 데이터에 수치값이나 텍스트를 표현해야 할 때 많이 사용된다.

2.3.2.1.1. `geom_point()`

데이터의 전반적 분포를 확인할 때 가장 많이 사용되는 기하요소가 점을 사용한 데이터 시각화이다. 이렇게 X, Y 축에 따라 데이터를 점으로 표현한 시각화를 산점도(Scatter Plot)라고 한다. 산점도는 X 축과 Y 축의 좌표값에 따라 점을 표시함으로서 데이터의 전반적인 분포를 살펴보고 X, Y 축의 증감에 따른 데이터의 상관관계를 살펴보는데 효과적인 시각화 방법이다.

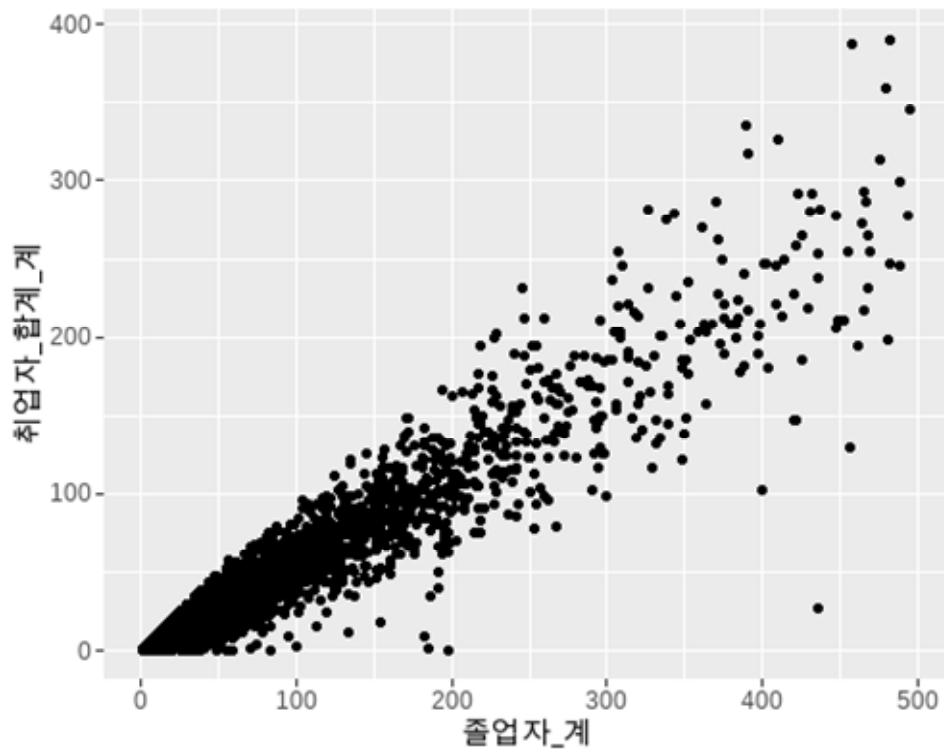
따라서 시각화 과정에서 데이터 값을 통계처리하지 않고 사용되기 때문에 통계요소가 'identity'로 설정된다.

```
geom_point(mapping = NULL, data = NULL, stat = "identity", position = "identity",
na.rm = FALSE, show.legend = NA, ...)
- mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용
- data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
- stat : 시각화에 적용될 통계요소, 기본값은 'identity'
- position : 시각화에 적용될 위치요소, 기본값은 'identity'
- na.rm : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- show.legend : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)
```

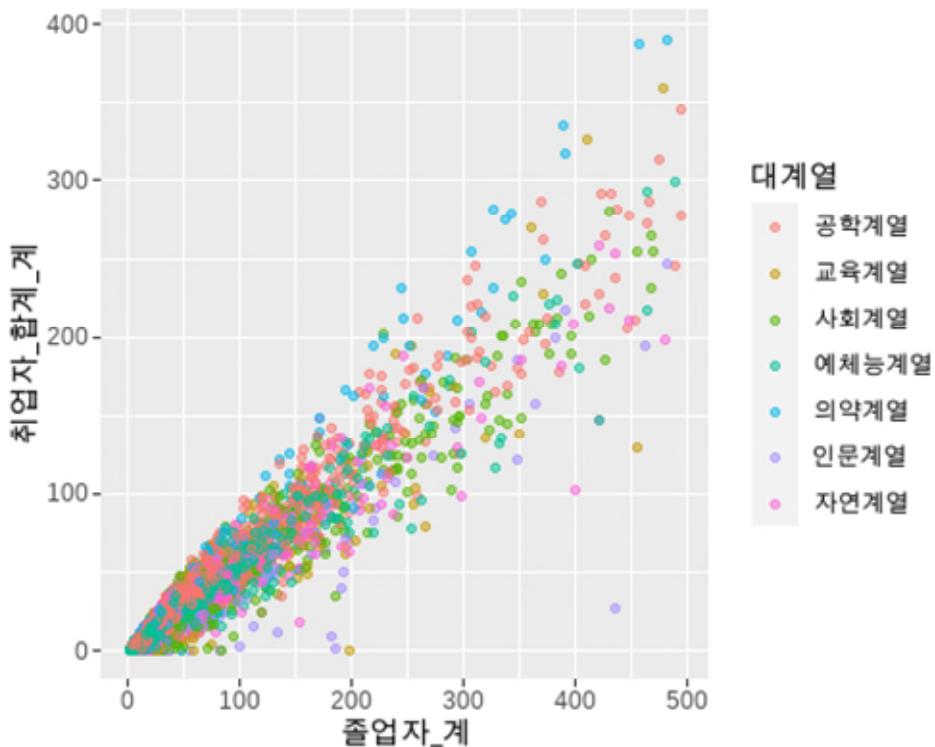
geom_point()에서 사용이 가능한 미적요소는 x, y, alpha, colour, fill, group, shape, size, stroke 등이다.

```
## df_취업통계 데이터 중 졸업자가 500 명 이하인 학과를 필터링하여 ggplot 객체로 생성하고 p_point에 저장
p_point <- df_취업통계 |> filter(졸업자_계 < 500) |>
ggplot()

## p_point 객체에 x 축은 졸업자_계, y 축은 취업자_합계_계로 매핑한 geom_point 레이어를 생성
p_point +
  geom_point(aes(x = 졸업자_계, y = 취업자_합계_계))
```



```
## p_point 객체에 x 축은 졸업자_계, y 축은 취업자_합계_계, color 를 대계열로 맵핑하고  
투명도를 설정한 geom_point 레이어를 생성  
p_point +  
  geom_point(aes(x = 졸업자_계, y = 취업자_합계_계, color = 대계열), alpha = 0.5)
```



2.3.2.1.2. `geom_text()`

`geom_text()`는 `geom_point()`와 데이터 표현방식은 동일하나 표현되는 데이터가 문자열이라는 점에서 차이가 있다. `geom_text()`는 주로 점이나 선, 막대 등으로 표현된 기하요소 레이어에 추가하여 해당 데이터들의 값이나 정보를 표현하기 위한 부가 정보를 제공하는데 활용된다. 그렇기 때문에 기본 기하요소와 겹쳐지 않기 위해 위치 조정을 위한 매개변수가 추가될 수 있다.

```
geom_text(mapping = NULL, data = NULL, stat = "identity", position = "identity",
  nudge_x = 0, nudge_y = 0, check_overlap = FALSE, na.rm = FALSE, show.legend = NA, ...)
```

- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 'identity'
- `position` : 시각화에 적용될 위치요소, 기본값은 'identity'
- `nudge_x` : X 축 방향으로 문자열의 이동
- `nudge_y` : Y 축 방향으로 문자열의 이동
- `check_overlap` : 텍스트가 겹치는 것을 허용할지에 대한 논리값(TRUE/FALSE)

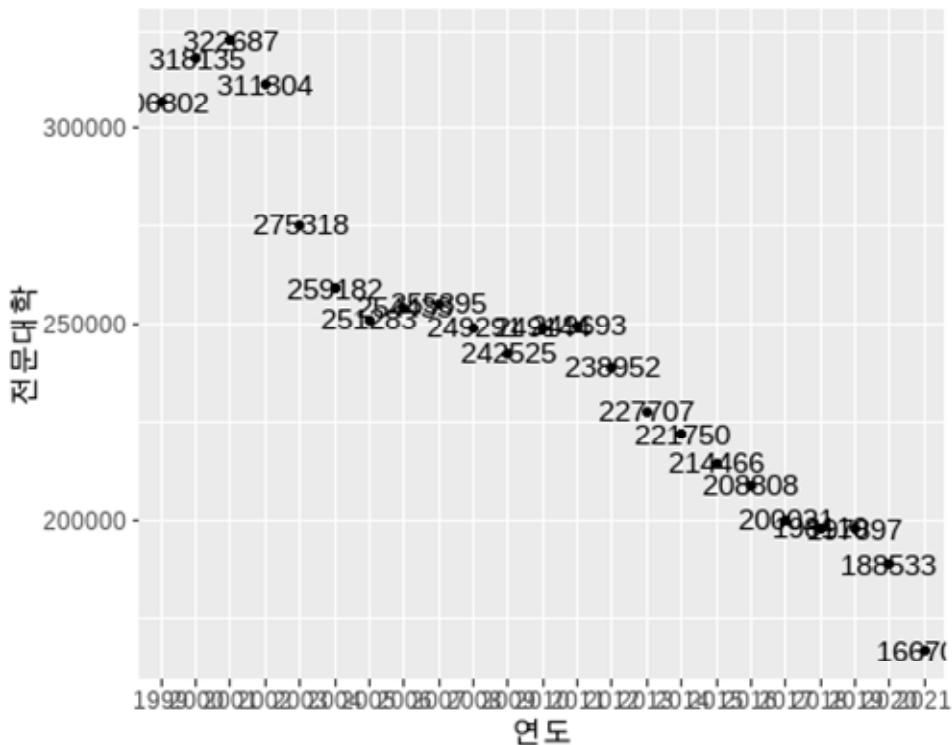
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

`geom_text()`에서 사용 가능한 미적요소는 `x`, `y`, `label`, `alpha`, `angle`, `color`, `family`, `fontface`, `group`, `hjust`, `lineheight`, `size`, `vjust` 등이다.

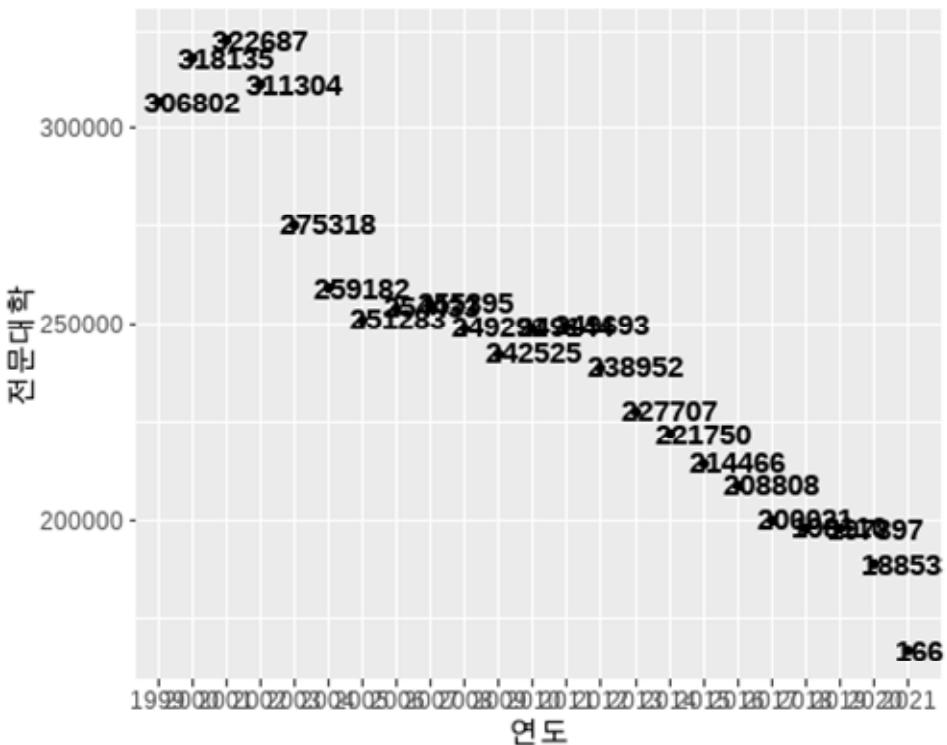
- `angle` : 문자열이 표현되는 각도 설정
- `family` : 문자열 시각화에 사용될 폰트 설정
- `fontface` : 문자열 시각화에 사용될 문자 특성("plain", "bold", "italic", "bold.italic")
- `hjust` : 수평 맞춤 설정, 0 부터 1 사이의 값을 지정
- `vjust` : 수직 맞춤 설정, 0 부터 1 사이의 값을 지정
- `lineheight` : 줄간격 설정, 0 부터 1 사이의 값을 지정

```
## df_입학자 데이터 중 지역이 전체인 데이터를 필터링하여 ggplot 객체로 생성하고 p_text에 저장
p_text <- df_입학자 |> filter(지역 == '전체') |>
  ggplot()

## p_text에 x 축은 연도, y 축은 전문대학을 매핑한 geom_point 레이어와 geom_text 레이어를 생성
p_text +
  geom_point(aes(x = 연도, y = 전문대학)) +
  geom_text(aes(x = 연도, y = 전문대학, label = 전문대학))
```



```
## p_text에 x 축은 연도, y 축은 전문대학을 매핑하고 미적요소를 설정한 geom_point 레이어와 geom_text 레이어를 생성
p_text +
  geom_point(aes(x = 연도, y = 전문대학)) +
  geom_text(aes(x = 연도, y = 전문대학, label = 전문대학), nudge_x = 1, fontface = 'bold')
```



2.3.2.1.3. `geom_label()`

`geom_label()`은 `geom_text()`과 거의 유사한 시각화를 제공한다. 다만 `geom_text()`는 문자열만을 시각화하지만 `geom_label()`은 박스로 둘러싸여진 문자열을 시각화한다. 따라서 문자열을 둘러싸고 있는 박스에 대한 설정외에는 `geom_text()`의 설정과 동일하다.

```
geom_label(mapping = NULL, data = NULL, stat = "identity", position = "identity",
           nudge_x = 0, nudge_y = 0, label.padding = unit(0.25, "lines"), label.r = unit(0.15, "lines"),
           label.size = 0.25, na.rm = FALSE, show.legend = NA, ...)
```

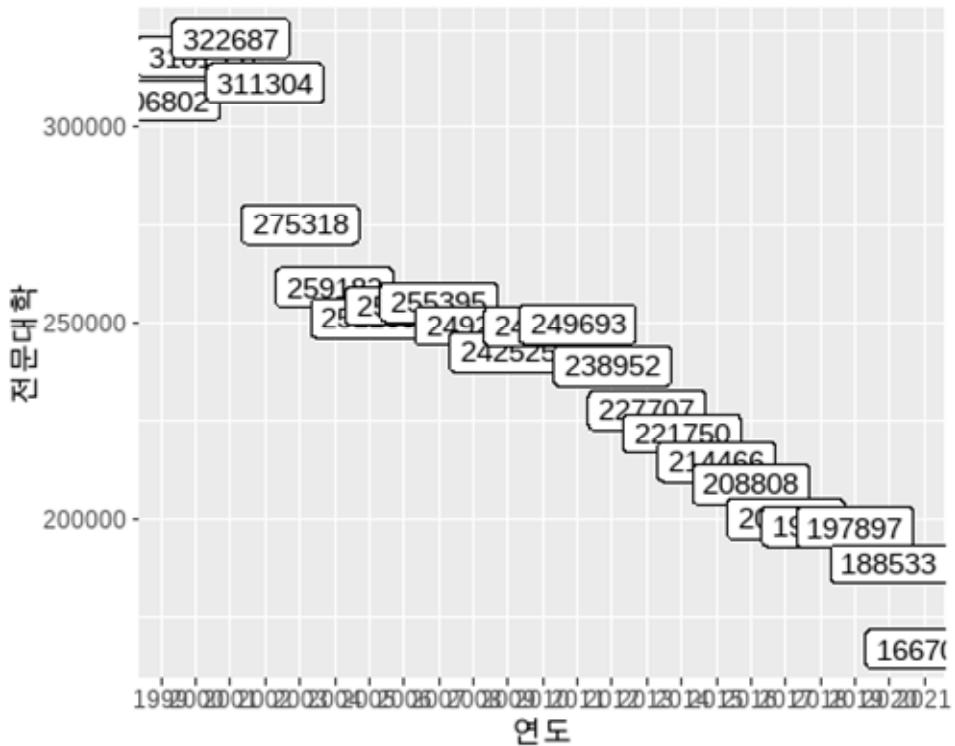
- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 '`identity`'
- `position` : 시각화에 적용될 위치요소, 기본값은 '`identity`'
- `nudge_x` : X 축 방향으로 문자열의 이동
- `nudge_y` : Y 축 방향으로 문자열의 이동
- `label.padding` : 박스와 문자열간의 여백 설정
- `label.r` = 박스의 귀퉁이를 둥글게 설정할 경우 반지름 설정
- `label.size` = 박스 라인의 두께 설정
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(`TRUE/FALSE`)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(`TRUE/FALSE`)

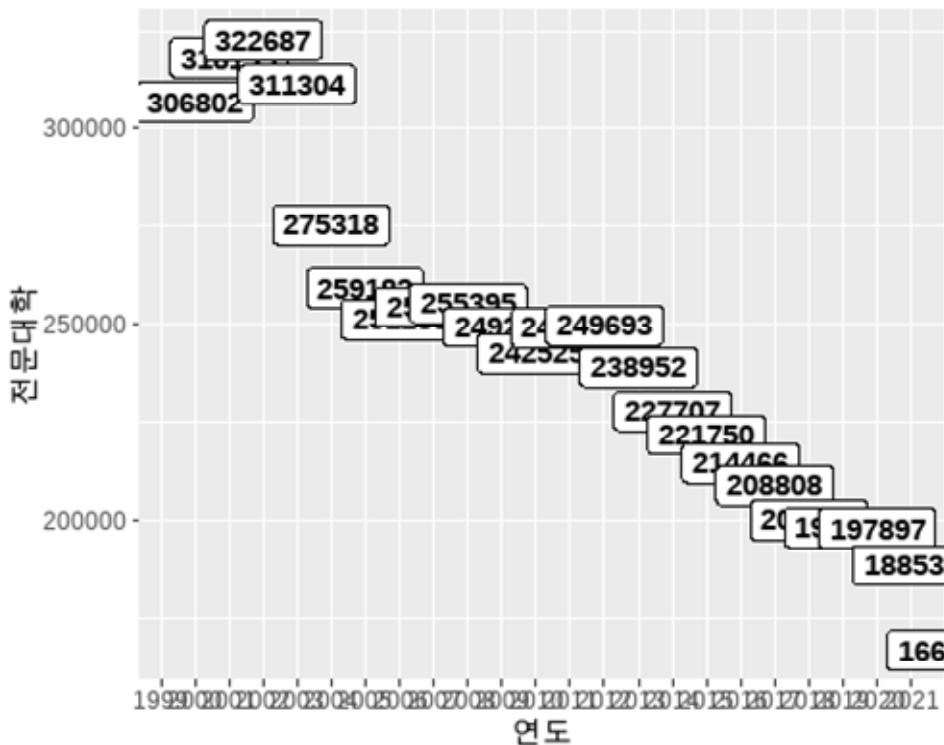
`geom_label()`에서 사용 가능한 미적요소는 x, y, label, alpha, angle, color, family, fontface, group, hjust, lineheight, size, vjust 등이다.

```
## df_입학자 데이터 중 지역이 전체인 데이터를 필터링하여 ggplot 객체로 생성하고 p_Label에 저장
p_Label <- df_입학자 |> filter(지역 == '전체') |>
  ggplot()

## p_Label에 x 축은 연도, y 축은 전문대학을 맵핑 geom_point 레이어와 geom_label 레이어를 생성
p_Label +
  geom_point(aes(x = 연도, y = 전문대학)) +
  geom_label(aes(x = 연도, y = 전문대학, label = 전문대학))
```



```
## p_Label에 x 축은 연도, y 축은 전문대학을 맵핑하고 미적요소를 설정한 geom_point 레이어와 geom_label 레이어를 생성
p_Label +
  geom_point(aes(x = 연도, y = 전문대학)) +
  geom_label(aes(x = 연도, y = 전문대학, label = 전문대학), nudge_x = 1, fontface = 'bold')
```



2.3.2.1.4. *geom_smooth()*

*geom_smooth()*는 데이터의 전반적인 추세선을 표현하는 기하요소 함수이다. 이 추세선은 데이터의 패턴을 찾기 위해 사용되고 특히 오버플로팅된 데이터에 대한 패턴을 찾아낼 때 유용하게 사용된다. *geom_smooth()*에서 추세선을 결정하기 위해 제공하는 통계 방법은 “lm”, “glm”, “gam”, “loess”의 네 가지를 제공한다.

```
geom_smooth(mapping = NULL, data = NULL, stat = "smooth", position = "identity",
method = NULL, formula = NULL, se = TRUE, na.rm = FALSE, orientation = NA,
show.legend = NA, ...)
```

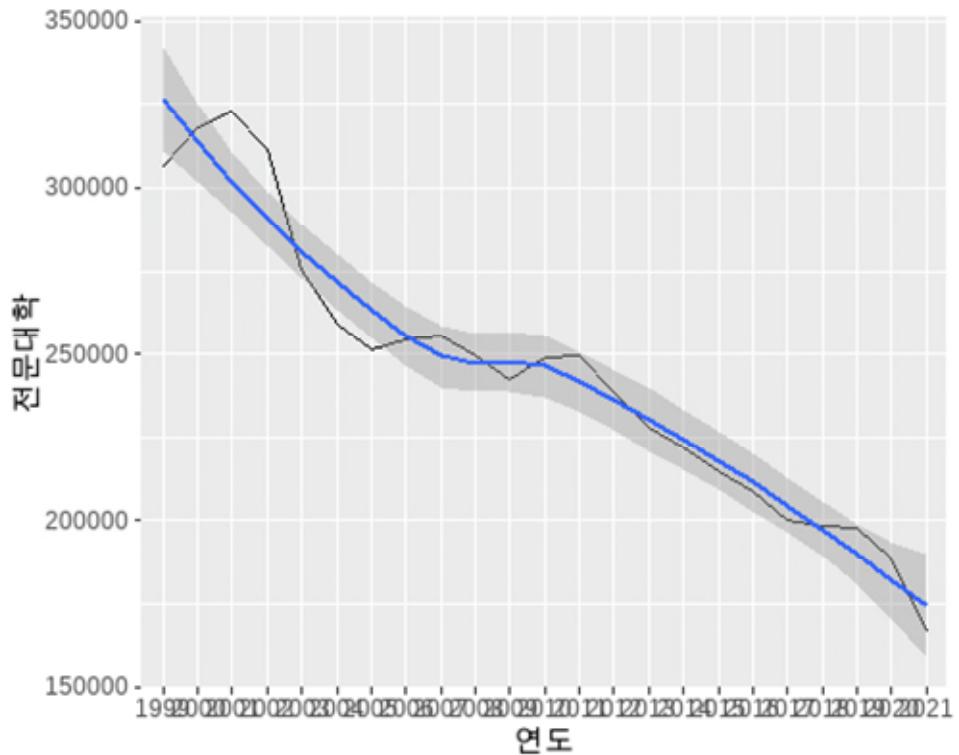
- **mapping** : *aes()*를 사용하여 매핑할 미적요소, 생략되면 *ggplot()*에 정의된 미적매핑 사용

- **data** : 시각화를 위해 사용될 데이터, 생략되면 *ggplot()*에 정의된 데이터 사용
- **stat** : 시각화에 적용될 통계요소, 기본값은 'smooth'
- **position** : 시각화에 적용될 위치요소, 기본값은 'identity'
- **method** : 추세선을 결정하기 위해 사용하는 통계 방법 설정, "lm", "glm", "gam", "loess" 중 하나임
- **formula** : 추세선을 결정하기 위해 사용하는 함수식 설정
- **se** : 추세선에 대한 표준오차 설정을 위한 논리값(TRUE/FALSE)
- **na.rm** : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- **show.legend** : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

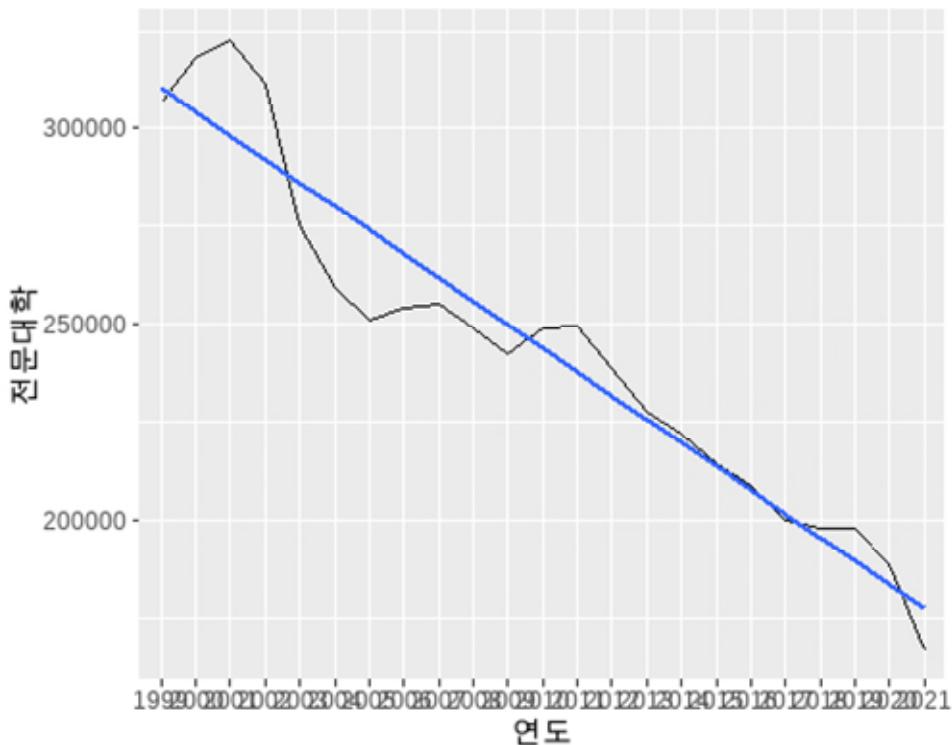
`geom_smooth()`에서 사용 가능한 미적 요소는 x, y, alpha, color, fill, group, linetype, size, weight, ymax, ymin 등이다.

```
## df_입학자 데이터 중 지역이 전체인 데이터를 필터링하여 ggplot 객체로 생성하고 p_smooth에 저장
p_smooth <- df_입학자 |> filter(지역 == '전체') |>
  ggplot()

## p_smooth에 x 축은 연도, y 축은 전문대학을 맵핑, geom_smooth의 추세선 결정 방법은
# 'loess'로 설정한 geom_line 레이어와 geom_smooth 레이어를 생성
p_smooth +
  geom_line(aes(x = 연도, y = 전문대학, group = 1)) +
  geom_smooth(aes(x = 연도, y = 전문대학, group = 1), method = 'loess')
```



```
## p_smooth에 x 축은 연도, y 축은 전문대학을 맵핑, geom_smooth의 추세선 결정 방법은
# 'lm', se는 FALSE로 설정한 geom_line 레이어와 geom_smooth 레이어를 생성
p_smooth +
  geom_line(aes(x = 연도, y = 전문대학, group = 1)) +
  geom_smooth(aes(x = 연도, y = 전문대학, group = 1), method = 'lm', se = FALSE)
```



2.3.2.2. 연속형 수치 변수와 이산형 변수

반별 성적 평균이라던지 지역별 매출액 평균등을 시각화 할 때같이 연속형 수치변수와 이산형 변수를 사용하여 데이터를 시각화하는 경우가 많이 있다. 사실 시각화의 가장 효과적인 활용 방법이 이 경우일 것이다.

2.3.2.2.1. `geom_col()`

`geom_col()`은 막대그래프를 그리기 위해 사용하는 시각화 함수이다. 앞에서 막대그래프는 이미 `geom_histogram()`과 `geom_bar()`가 설명되었다. 그런데 또 `geom_col()`이 제공된다. 막대그래프는 시각화하고자 하는 데이터의 종류에 따라 각각의 특성에 맞는 함수가 제공된다. 하지만 `geom_col()`이 가장 포괄적인 막대그래프 시각화 함수로써 `geom_col()`을 사용하여 `geom_histogram()`과 `geom_bar()`와 동일한 막대그래프를 그릴 수도 있다. 특히 `geom_col()`은 `geom_bar(stat = 'identity')`와 동일한 시각화 효과를 낸다.

```
geom_col(mapping = NULL, data = NULL, position = "stack", width = NULL, na.rm = FALSE, show.legend = NA, ...)

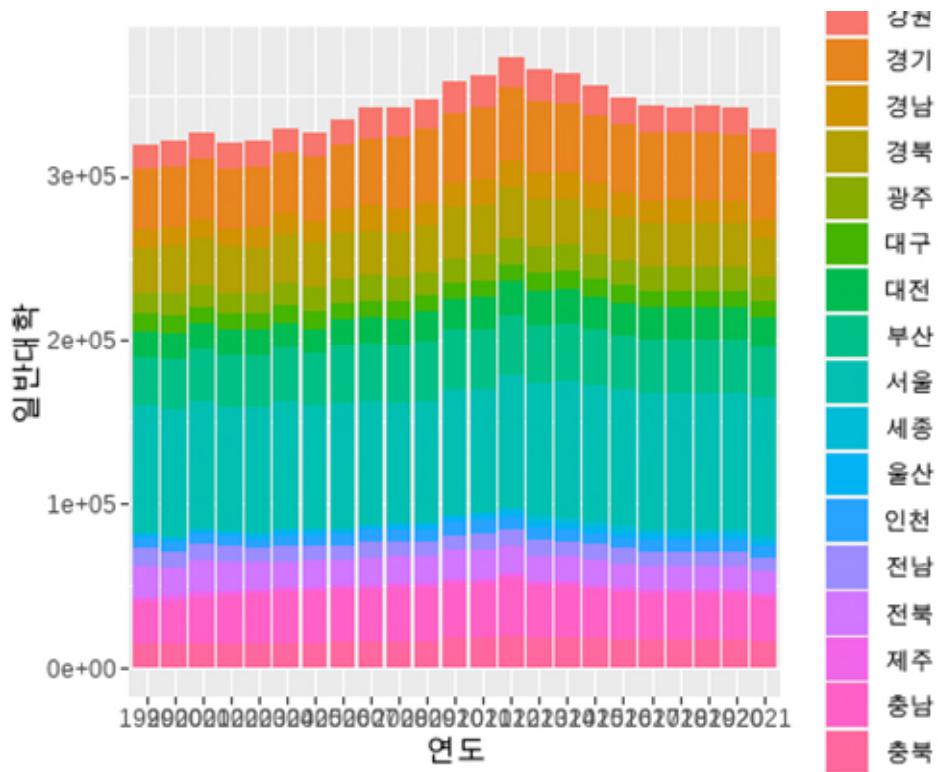
- mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용
- data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
- position : 시각화에 적용될 위치요소, 기본값은 'stack'
```

- width : 막대의 너비 설정
- na.rm : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- show.legend : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

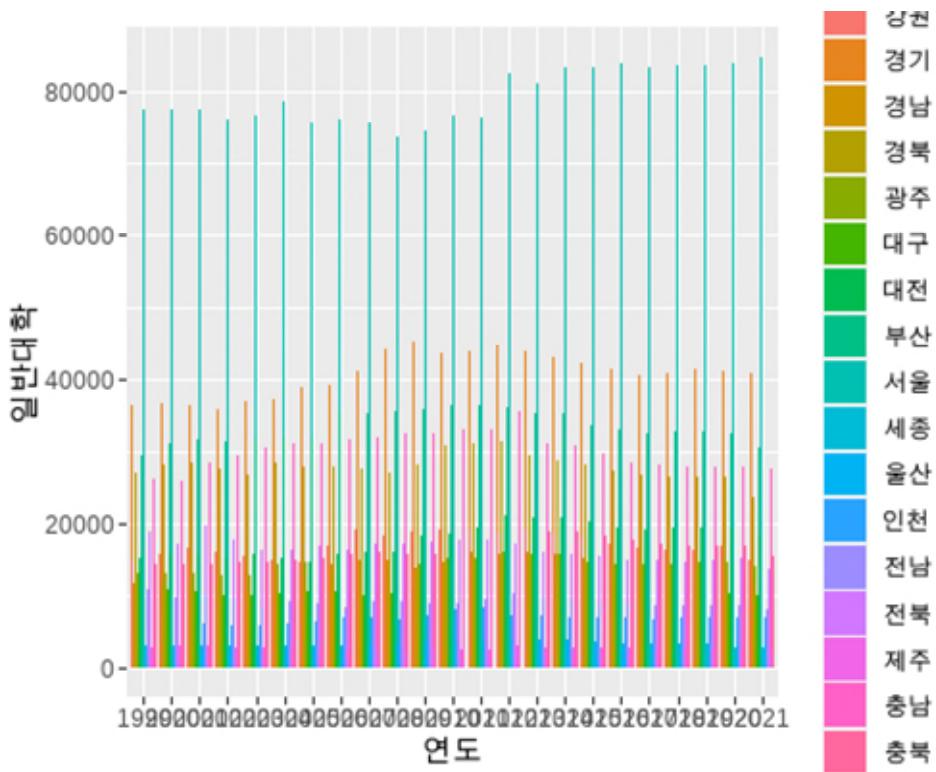
`geom_col()`에서 특별하게 활용되는 매개변수가 `position`이다. 이 `position`을 설정하여 막대그래프의 타입을 달리하여 시각화가 가능하다는 것이다. `position`의 종류는 `stack`, `dodge`, `fill`의 세 가지가 많이 사용된다. `stack`은 막대를 구성하는 그룹을 하나의 막대에 쌓아가면서 표현하고 `dodge`는 막대를 구성하는 그룹에 대한 막대를 각각 구성하여 옆에 붙여준다. `fill`은 전체를 100%로 막대를 구성하고 그 막대를 구성하는 그룹의 구성 비율에 따라 막대를 구분해 준다. `fill`은 전체를 100%로 변환하기 때문에 모든 막대의 길이는 같게 되고 Y 축의 레벨이 0 부터 1 까지로 변경된다.

```
## df_입학자 데이터 중 지역이 전체가 아닌 데이터를 필터링하여 ggplot 객체로 생성하고
p_col에 저장
p_col <- df_입학자 |> filter(지역 != '전체') |>
  ggplot()

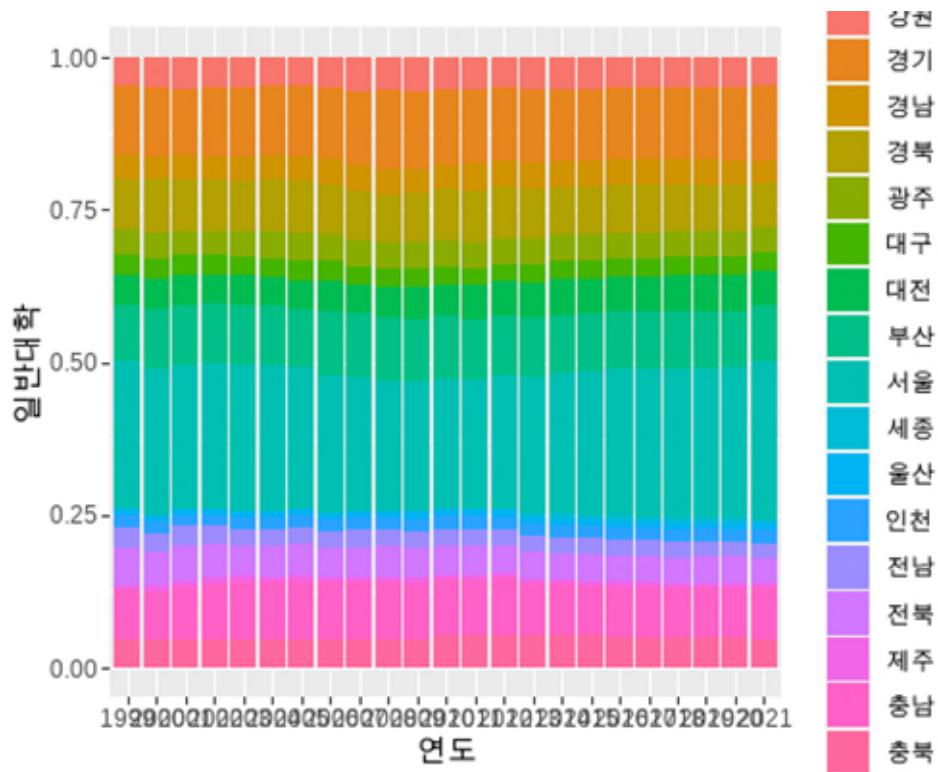
## p_col에 x 축은 연도, y 축은 일반대학을 맵핑하고 position 을 stack 으로 설정한 geom_col 레이어를 생성
p_col +
  geom_col(aes(x = 연도, y = 일반대학, fill = 지역), position = 'stack')
```



```
## p_col에 x 축은 연도, y 축은 일반대학을 매핑하고 position 을 stack 으로 설정한 geo
## m_col 레이어를 생성
p_col +
  geom_col(aes(x = 연도, y = 일반대학, fill = 지역), position = 'dodge')
```

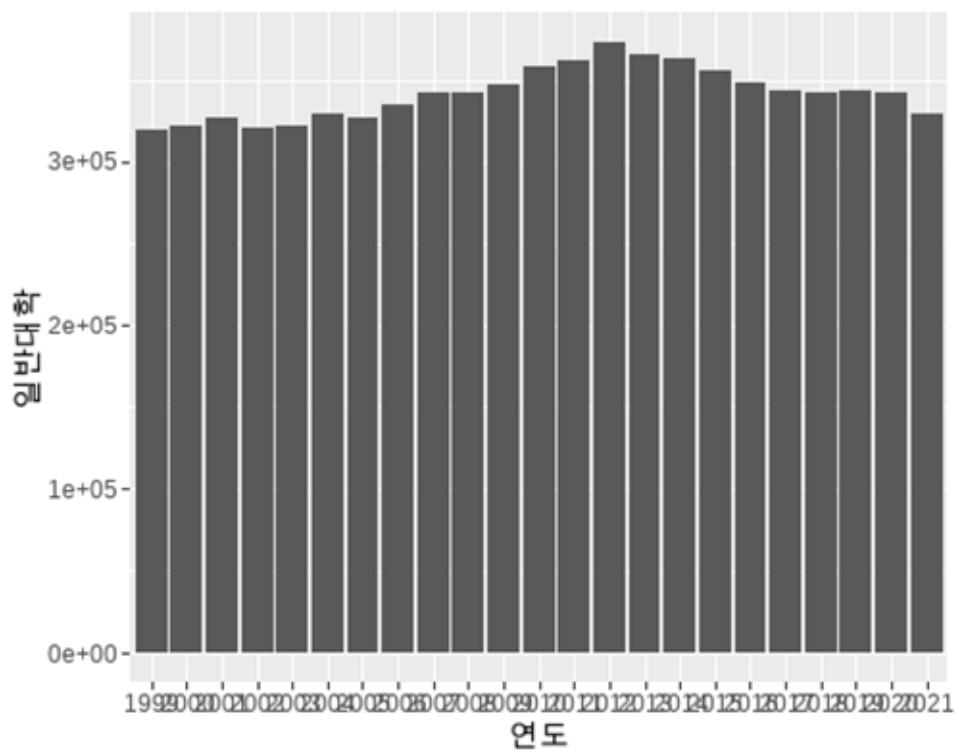


```
## p_col에 x축은 연도, y축은 일반대학을 맵핑하고 position을 stack으로 설정한 geom_col 레이어를 생성
p_col +
  geom_col(aes(x = 연도, y = 일반대학, fill = 지역), position = 'fill')
```

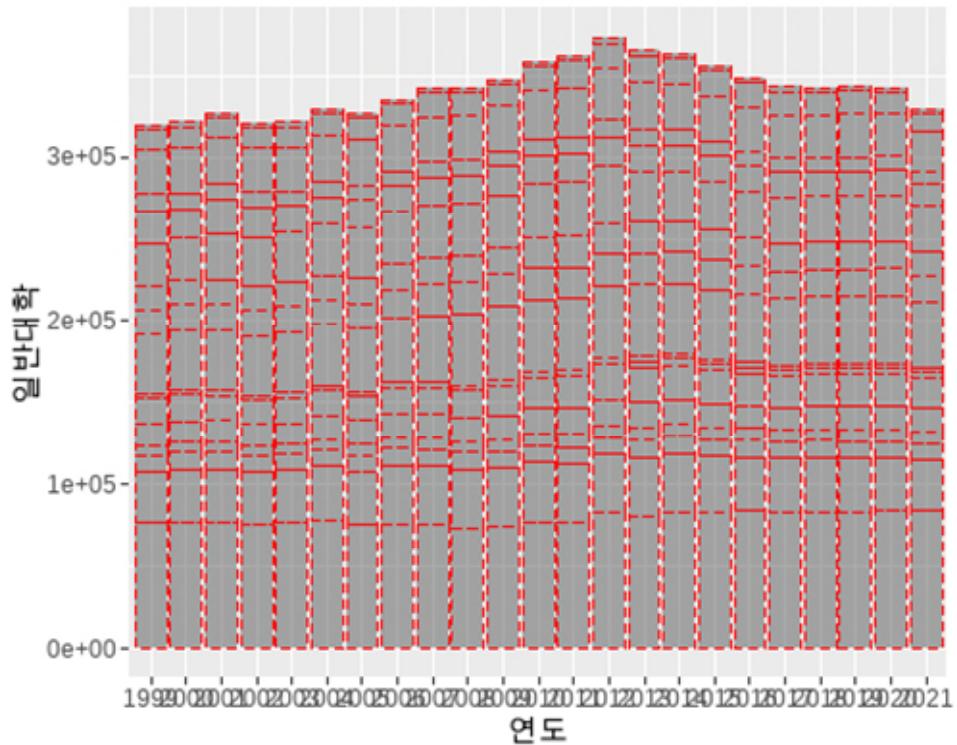


geom_col()`에서 사용 가능한 미적요소는 x, y, alpha, color, fill, group, linetype, size 등이다.

```
## p_col에 x 축은 연도, y 축은 일반대학을 맵핑하고 stat 을 'identity'로 설정한 geom_col 레이어를 생성, geom_col 과 동일한 결과
p_col +
  geom_bar(aes(x = 연도, y = 일반대학), stat = 'identity')
```



```
## p_col에 x 축은 연도, y 축은 일반대학으로 매핑하고 미적요소를 설정한 geom_col 레이어  
를 생성  
p_col +  
  geom_col(aes(x = 연도, y = 일반대학), alpha = 0.5, color = 'red', linetype = 2)
```

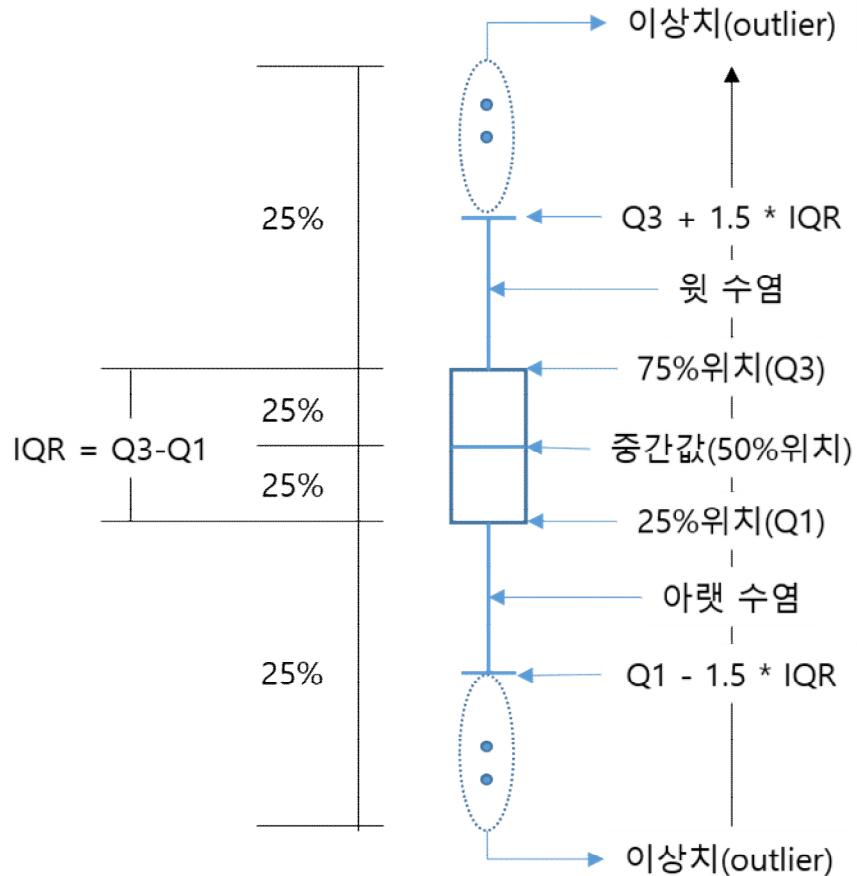


2.3.2.2.2. *geom_boxplot()*

박스 플롯은 데이터의 전반적 분포를 살펴볼 수 있는 매우 좋은 시각화 방법이다. 앞서 설명한 산점도는 X, Y 축 모두 연속형 수치 변수를 사용하지만 박스 플롯은 이산형 변수로 분류되는 데이터의 전반적 분포를 시각화할 수 있다는 점에서 활용도가 크고 Scatter Plot 에서는 표현할 수 없었던 중간값, IQR 범위, 이상치 등의 추가적인 정보를 같이 볼 수 있다는 장점이 있다. R에서 박스 플롯 레이어를 생성하기 위해서는 `geom_boxplot()`을 사용한다.

박스 플롯으로 표현되는 상자의 중심은 중앙값이다. 중앙값은 가로선으로 표현되며 그 주위를 상자가 둘러싸고 있다. 상자의 상단과 하단은 관측값들의 25%~75%를 나타내고 중간 선은 전체 데이터의 중간값(50%)을 표현한다. 상자의 상단과 하단 범위를 벗어나면 사분위 범위의 1.5 배에 해당하는 '수염(whisker)'으로 표현된다. 수염 밖으로 존재하는 값은 점으로 표기하며

일반적으로 이상치로 간주한다.



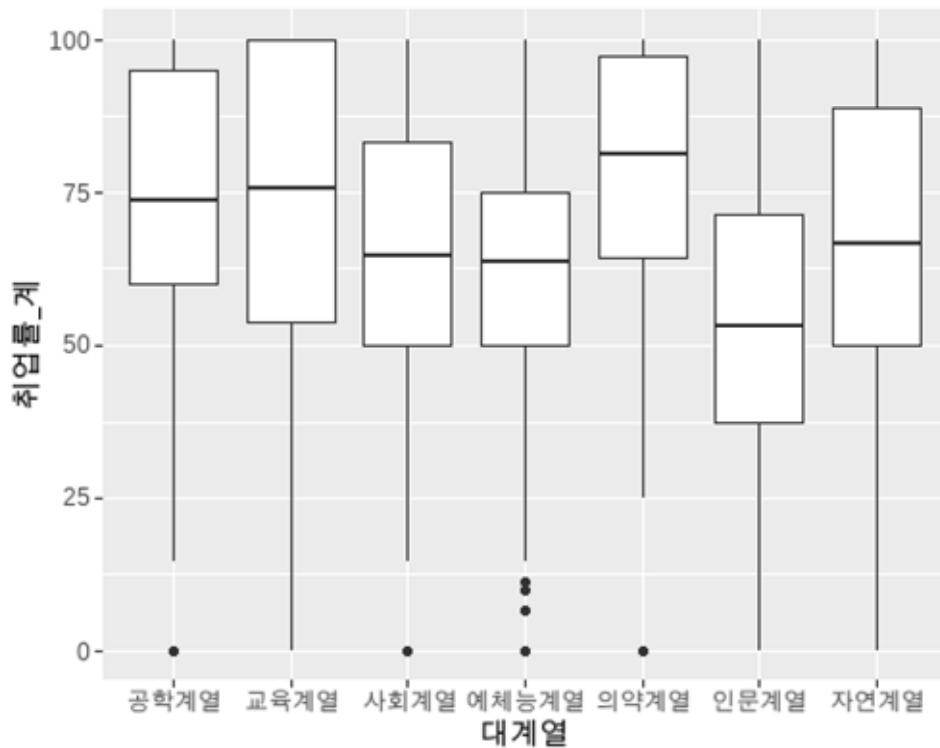
```
geom_boxplot(mapping = NULL, data = NULL, stat = "boxplot", position = "dodge2",
  outlier.color = NULL, outlier.fill = NULL, outlier.shape = 19, outlier.size = 1.5,
  outlier.stroke = 0.5, outlier.alpha = NULL, notch = FALSE, notchwidth = 0.5,
  na.rm = FALSE, show.legend = NA, ...)
```

- **mapping** : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용
- **data** : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
- **stat** : 시각화에 적용될 통계요소, 기본값은 'boxplot'
- **position** : 시각화에 적용될 위치요소, 기본값은 'dodge2'
- **outlier.colour**, **outlier.color**, **outlier.fill**, **outlier.shape**, **outlier.size**, **outlier.stroke**, **outlier.alpha** : 이상치로 표현되는 점의 미적요소 설정
- **notch** : 상자의 중간값을 구분하는 귀퉁이를 표현할지를 설정하기 위한 논리값(TRUE/FALSE)
- **na.rm** : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- **show.legend** : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

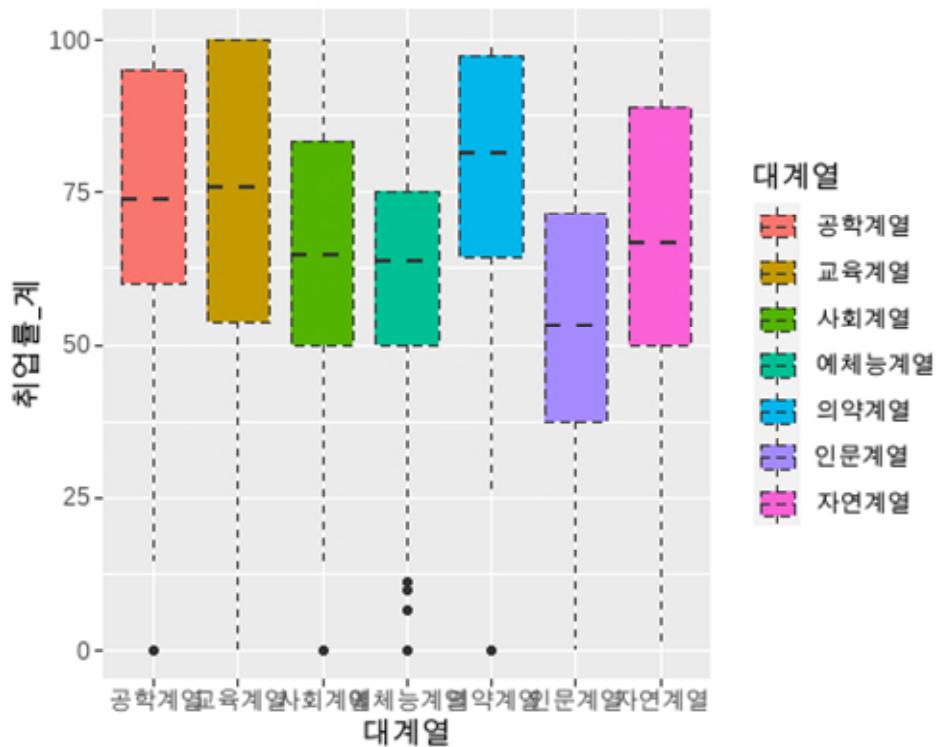
`geom_boxplot()`에서 사용이 가능한 미적요소는 x, y, lower(xlower), upper(xupper), middle(xmiddle), ymin(xmin), ymax(xmax), alpha, colour, fill, group, linetype, shape, size, weight 등이 있다. 이 중 lower(xlower), upper(xupper), middle(xmiddle), ymin(xmin), ymax(xmax)는 박스를 수동으로 설정하기 위해 사용하는 미적요소이다.

```
## df_취업통계 데이터를 ggplot 객체로 생성하고 p_boxplot에 저장
p_boxplot <- df_취업통계 |>
  ggplot()

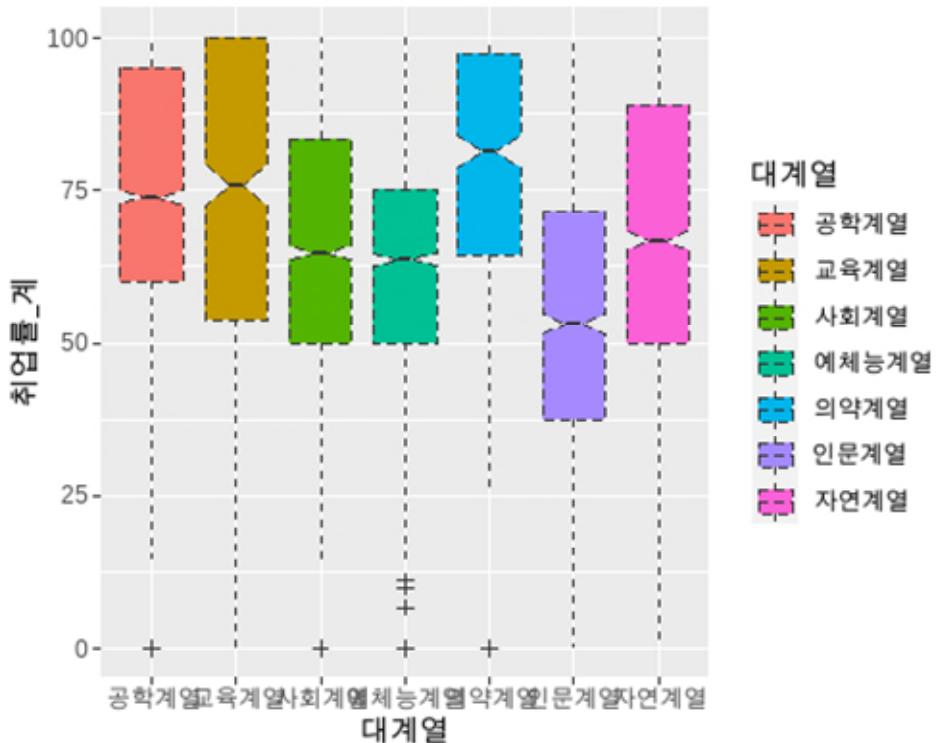
## p_boxplot에 x 축에 대계열, y 축에 취업률_계로 맵핑한 geom_boxplot 레이어를 생성
p_boxplot +
  geom_boxplot(aes(x = 대계열, y = 취업률_계))
```



```
## p_boxplot에 x 축에 대계열, y 축에 취업률_계, fill을 대계열로 맵핑, 미적요소를 설정
# 한 geom_boxplot 레이어를 생성
p_boxplot +
  geom_boxplot(aes(x = 대계열, y = 취업률_계, fill = 대계열), linetype = 2)
```



```
## p_boxplot 에 x 축에 대계열, y 축에 취업률_계, fill 을 대계열로 맵핑, 미적요소를 설정
한 geom_boxplot 레이어를 생성
p_boxplot +
  geom_boxplot(aes(x = 대계열, y = 취업률_계, fill = 대계열), linetype = 2, notch
= TRUE, notchwidth = 0.2, outlier.shape = 3)
```



2.3.2.2.3. *geom_violin()*

*geom_violin()*은 박스플롯과 거의 유사한 시각화 방법인데 박스대신 데이터의 분포를 표현하는 바이올린과 유사한 형태의 도형으로 표현하는 시각화 방법이다. 박스 플롯에서는 25%부터 75%까지의 범위를 표현하는데 유용하지만 그 범위내에서 데이터가 어느 구간에 많이 분포되어 있는지는 표현하지 못한다. 이와 같은 단점을 극복하기 위해 *geom_violin()*은 데이터가 많이 분포하는 구간은 좌우를 넓게 표현하고 데이터가 적게 분포하는 구간은 좌우를 좁게 표현해서 데이터의 전체 구간에 데이터의 분포를 시각화할 수 있다.

```
geom_violin(mapping = NULL, data = NULL, stat = "ydensity", position = "dodge",
draw_quantiles = NULL, trim = TRUE, scale = "area", na.rm = FALSE, show.legend = NA, ...)
```

- **mapping** : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

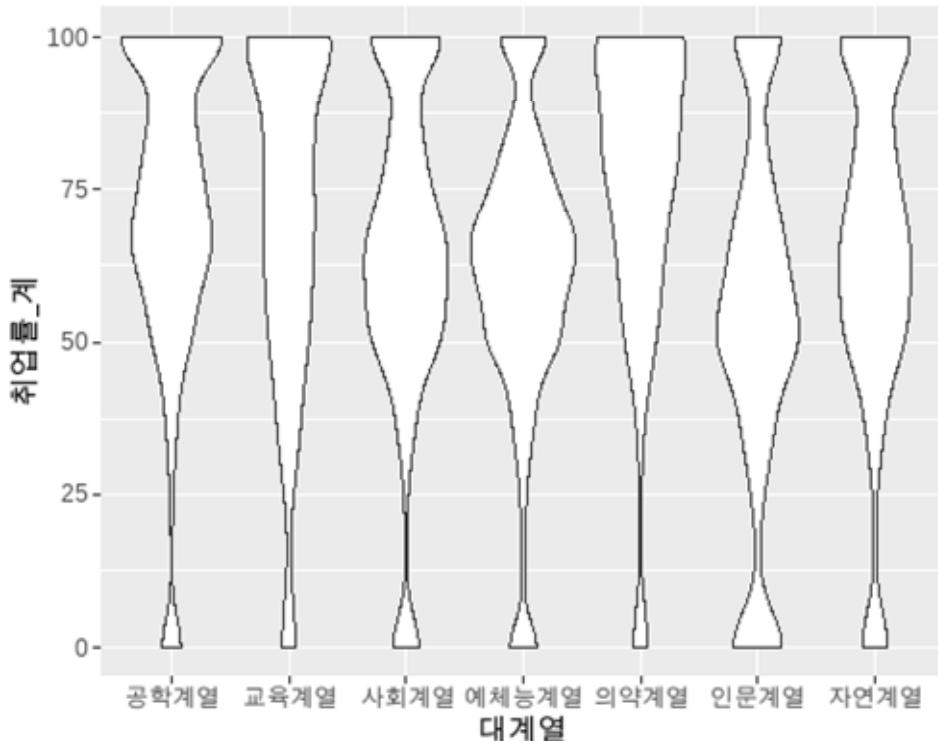
- **data** : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- **stat** : 시각화에 적용될 통계요소, 기본값은 'ydensity'
- **position** : 시각화에 적용될 위치요소, 기본값은 'dodge'
- **draw_quantiles** : 박스플롯에서 표현하는 것과 같이 분위라인을 그리는데 사용될 분위 설정
- **trim** : 끝부분을 잘라낼지 여부를 결정하는 논리값(TRUE/FALSE)

- `scale` : 데이터의 분포를 표현하는 도형의 크기를 표현하는 단위 설정, '`area`'는 모든 도형의 크기를 동일하게 설정, '`count`'는 데이터 개수에 따라 도형의 크기를 설정, '`width`'는 모든 도형의 최대 너비를 동일하게 설정
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

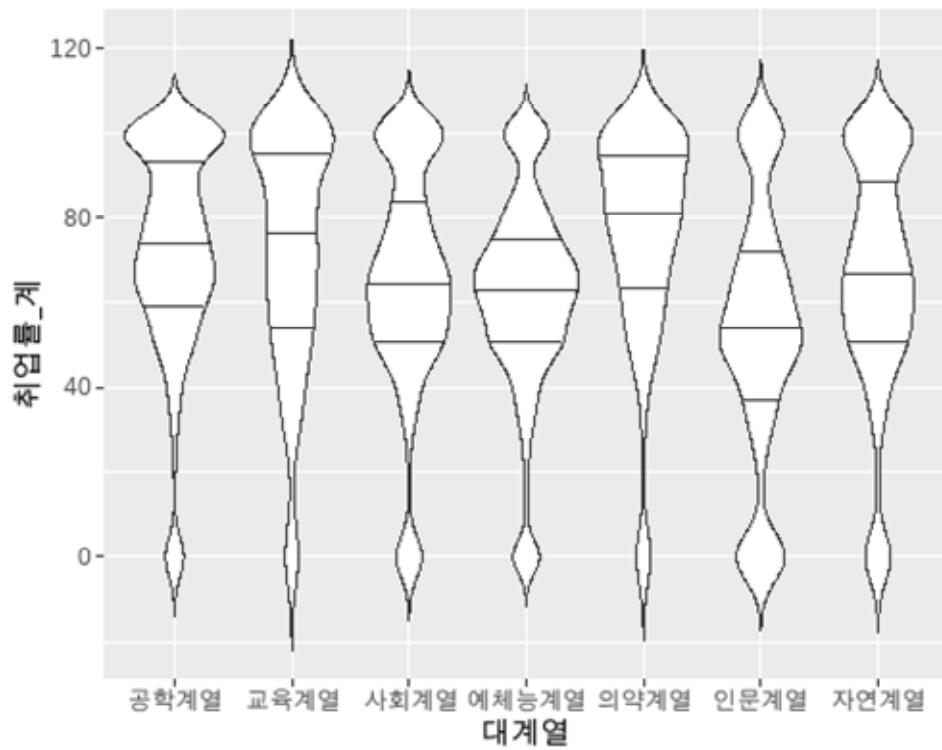
`geom_violin()`에서 사용할 수 있는 미적요소는 `x`, `y`, `alpha`, `color`, `fill`, `group`, `linetype`, `size`, `weight` 등이다.

```
## df_취업통계 데이터를 ggplot 객체로 생성하고 p_violin에 저장
p_violin <- df_취업통계 |>
  ggplot()

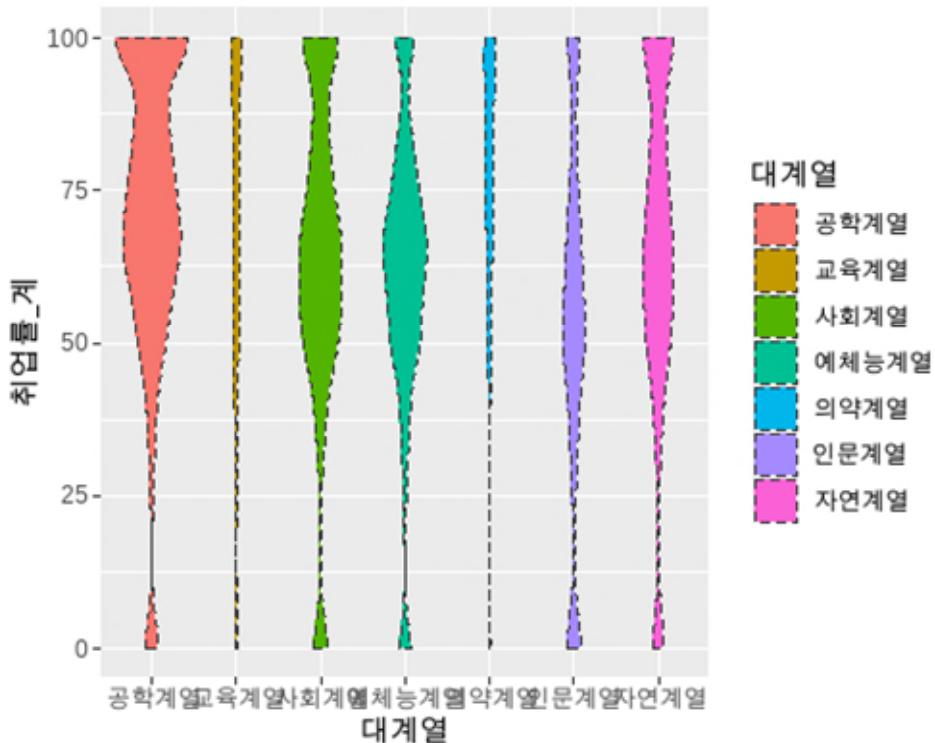
## p_violin에 x 축에 대계열, y 축에 취업률_계로 매팅한 geom_violin 레이어를 생성
p_violin +
  geom_violin(aes(x = 대계열, y = 취업률_계))
```



```
## p_violin에 x 축에 대계열, y 축에 취업률_계로 매팅하고 0.25, 0.5, 0.75 위치에 선을
그림고 양쪽 끝을 자른 geom_violin 레이어를 생성
p_violin +
  geom_violin(aes(x = 대계열, y = 취업률_계), draw_quantiles = c(0.25, 0.5, 0.75),
              trim = F)
```



```
## p_violin에 x 축에 대계열, y 축에 취업률_계, color 를 대계열로 매핑하고 선타입을 2  
로, 도형 표현 단위를 'count'로 설정한 geom_violin 레이어를 생성  
p_violin +  
  geom_violin(aes(x = 대계열, y = 취업률_계, fill = 대계열), linetype = 2, scale =  
  'count')
```



2.3.2.3. 두개의 이산형 변수

X 축과 Y 축 모두 이산형 변수를 표현하고자 하는 경우는 X 축과 Y 축이 만나는 위치에 해당하는 데이터의 개수를 표현하는 방법이 거의 유일한 데이터 시각화 방법이다. 이를 위해 제공하는 함수가 `geom_count()`이다. `geom_count()`는 양 축의 교차되는 위치에 해당하는 데이터의 크기를 도형의 크기로 표현하는 시각화 방법이다.

```
geom_count(mapping = NULL, data = NULL, stat = "sum", position = "identity", na.rm = FALSE,
show.legend = NA, ...)
- mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용
- data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
- stat : 시각화에 적용될 통계요소, 기본값은 'sum'
- position : 시각화에 적용될 위치요소, 기본값은 'identity'
- na.rm : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- show.legend : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)
```

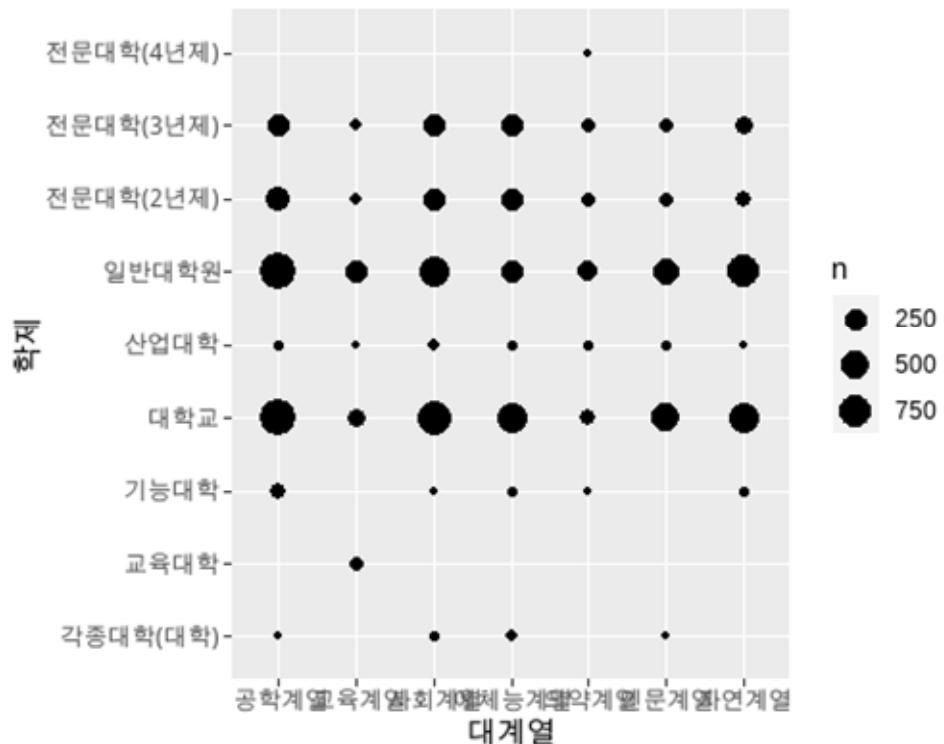
`geom_count()`에서 사용이 가능한 미적요소는 `x`, `y`, `alpha`, `color`, `fill`, `group`, `shape`, `size`, `stroke` 등이다.

```

## df_취업통계 데이터를 ggplot 객체로 생성하고 p_count에 저장
p_count <- df_취업통계 |>
  ggplot()

## p_count에 x 축에 대계열, y 축에 학제를 매핑한 geom_count 레이어를 생성
p_count +
  geom_count(aes(x = 대계열, y = 학제))

```



2.3.3. 세개의 연속형 변수

`ggplot`은 3 차원 시각화는 지원하지 않는다. 하지만 3 차원 공간을 2 차원 평면에서 구현하는 방법을 제공한다. 그래서 X, Y, Z 의 세 축으로 표현되는 3 차원을 2 차원으로 표현하기 위해서는 먼저 X, Y 의 조합이 중복되지 않는 유일한 조합으로 설정되어야 하고 z 값이 같은 위치들을 선으로 이어 등고선 형태로 이어주거나 해당 위치의 데이터 값을 색으로 표현하는 두가지 방법이 있다. 앞의 등고선을 사용하는 방법은 `geom_contour()`를 사용하고 색으로 표현하는 방법은 `geom_contour_filled()`를 사용한다.

```

geom_contour(mapping = NULL, data = NULL, stat = "contour", position = "identity",
  bins = NULL, binwidth = NULL, breaks = NULL, na.rm = FALSE, show.legend = NA,
  ...)
  - mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑

```

사용

- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 'contour'
- `position` : 시각화에 적용될 위치요소, 기본값은 'identity'
- `bins` : X 축을 나누는 bin 의 개수 설정
- `binwidth` : X 축을 나누는 bin 의 너비 설정, 숫자벡터를 사용할 수 있다. (`bin`과 `binwidth`는 동시에 사용될 수 없다)
- `breaks` : 등고선이 구분되는 수치 벡터 설정
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

`geom_contour()`에서 사용이 가능한 미적요소는 `x`, `y`, `alpha`, `color`, `group`, `linetype`, `size`, `weight` 등 이다.

```
## df_취업통계 데이터를 ggplot 객체로 생성하고 p_contour에 저장
p_contour <- df_취업통계 |>
  ggplot()

## p_contour에 x 축에 졸업자_계, y 축에 취업자_합계_계, z 축에 취업률_계를 매핑한 geom_contour 레이어를 생성
p_contour +
  geom_contour(aes(x = 졸업자_계, y = 취업자_합계_계, z = 취업률_계))
```



`geom_contour_filled()`의 사용법과 주요 매개변수는 다음과 같다.

```
geom_contour_filled(mapping = NULL, data = NULL, stat = "contour_filled", position = "identity", bins = NULL, binwidth = NULL, breaks = NULL, na.rm = FALSE, show.legend = NA, ...)
```

- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 '`contour_filled`'
- `position` : 시각화에 적용될 위치요소, 기본값은 '`identity`'
- `bins` : X 축을 나누는 bin 의 개수 설정
- `binwidth` : X 축을 나누는 bin 의 너비 설정, 숫자벡터를 사용할 수 있다. (`bin` 과 `binwidth`는 동시에 사용될 수 없다)
- `breaks` : 등고선이 구분되는 수치 벡터 설정
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

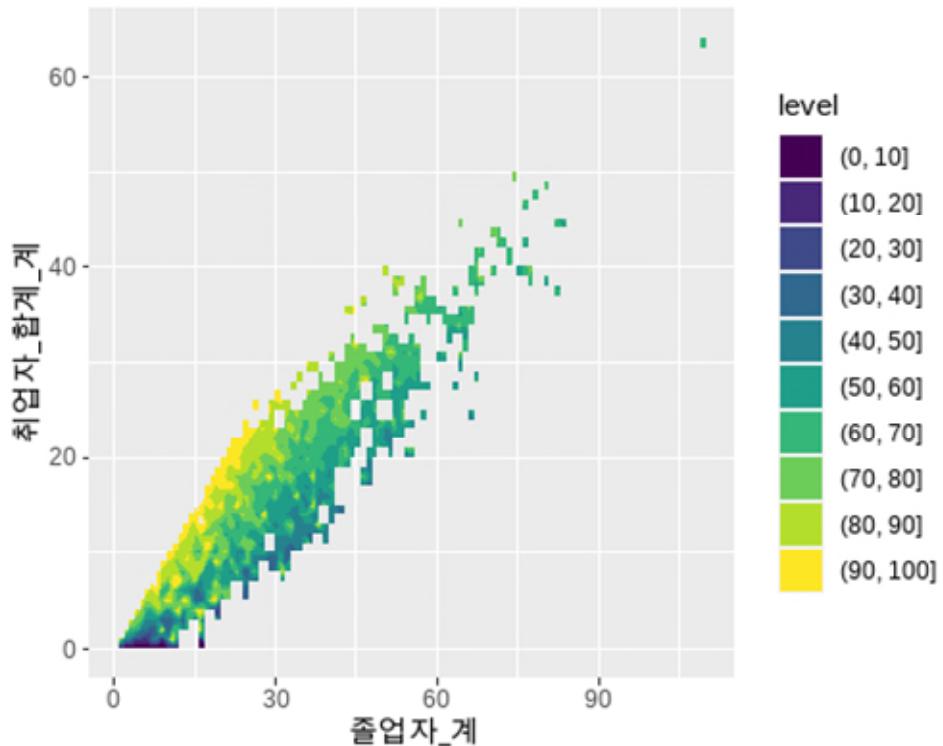
`geom_contour_filled()`에서 사용이 가능한 미적요소는 `x`, `y`, `alpha`, `color`, `group`, `linetype`, `size`, `subgroup` 등 이다.

```

## df_취업통계 데이터를 ggplot 객체로 생성하고 p_contour_filled에 저장
p_contour_filled <- df_취업통계 |>
  ggplot()

## p_contour_filled에 x 축에 졸업자_계, y 축에 취업자_합계_계, z 축에 취업률_계를 매핑
## 한 geom_contour_filled 레이어를 생성
p_contour_filled +
  geom_contour_filled(aes(x = 졸업자_계, y = 취업자_합계_계, z = 취업률_계))

```



2.3.4. 관측치 연결

관측치 연결이 가장 많이 사용되는 데이터가 시계열 데이터이다. 시계열 데이터는 시간의 흐름에 따라 표현되는 데이터이다. 이 데이터는 보통 X 축의 왼쪽에서 오른쪽으로 이동하면서 데이터가 표현된다. 따라서 X 축에는 이산형 데이터로 표현되는 시간(연, 월, 일 등)이나 횟수가 매핑되고 하나의 X 변량에는 하나의 데이터만이 표현되어 이를 데이터들을 서로 연결하여 하나의 시계열 데이터를 생성한다. 보통 이 시계열 데이터는 선 그래프로 표현되는데 하나의 시각화에 여러개의 선으로 여러 시계열 데이터를 표현할 수도 있다.

2.3.4.1. geom_line()

시계열 데이터를 표현하는 선 그래프를 시각화하는데에는 `geom_line()`이 사용된다.

```
geom_line(mapping = NULL, data = NULL, stat = "identity", position = "identity",
na.rm = FALSE, show.legend = NA, ...)
```

- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

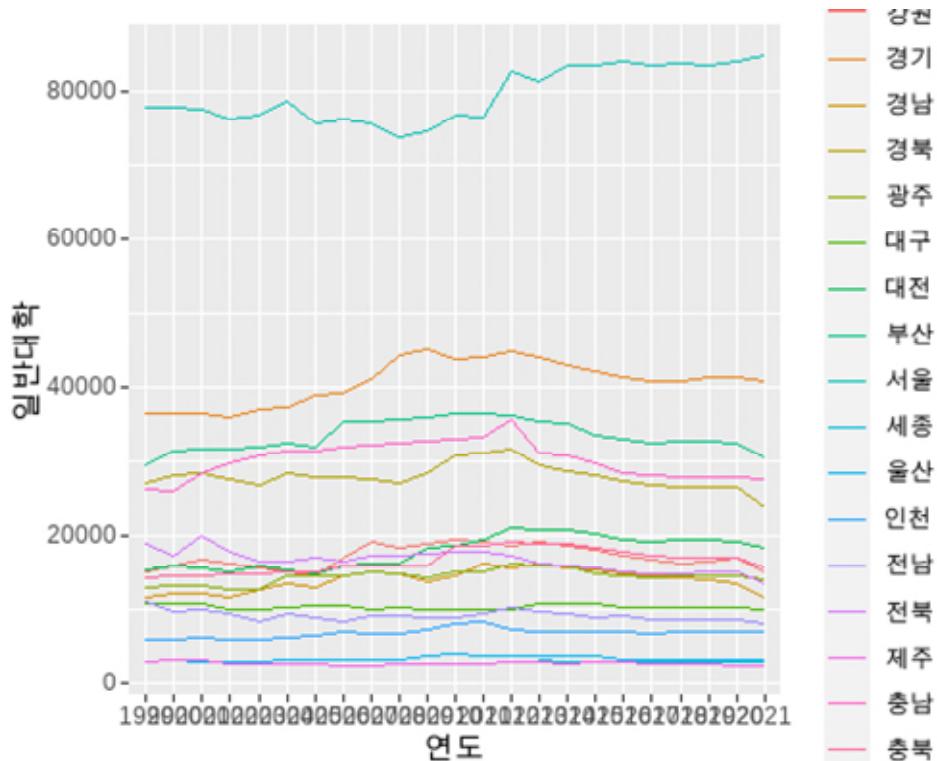
- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `stat` : 시각화에 적용될 통계요소, 기본값은 '`identity`'
- `position` : 시각화에 적용될 위치요소, 기본값은 '`identity`'로 권장됨
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(`TRUE/FALSE`)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(`TRUE/FALSE`)

`geom_line()`에서 사용이 가능한 미적요인은 x, y, alpha, color, group, linetype, size 등이다.

df_입학자 데이터 중 지역이 전체가 아닌 데이터를 필터링하여 ggplot 객체로 생성하고 p_line에 저장

```
p_line <- df_입학자 |> filter(지역 != '전체') |>  
ggplot()
```

```
## p_line 객체에 x 축에 연도, y 축에 일반대학으로 매핑하고 각각의 line 은 지역으로 그루핑하고, color 는 지역으로 매핑한 geom_line 레이어를 생성  
p_line +  
  geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역))
```



2.3.4.2. geom_area()

geom_area()는 geom_line()과 유사한 선 그래프를 그린다. 다만 geom_line()은 데이터를 연결하는 선만이 그려지는 반면 geom_area()는 데이터를 연결한 선에서부터 X 축까지의 면적을 색으로 채운다. 하지만 더 큰 차이는 여러 변량이 표현되는 다중 선 그래프의 경우 면적이 계속 쌓이는 방식으로 표현된다는 점이다.

```
geom_area(mapping = NULL, data = NULL, stat = "identity", position = "stack", na.rm = FALSE, show.legend = NA, outline.type = "upper", ...)
```

- mapping : aes()를 사용하여 매핑할 미적요소, 생략되면 ggplot()에 정의된 미적매핑 사용

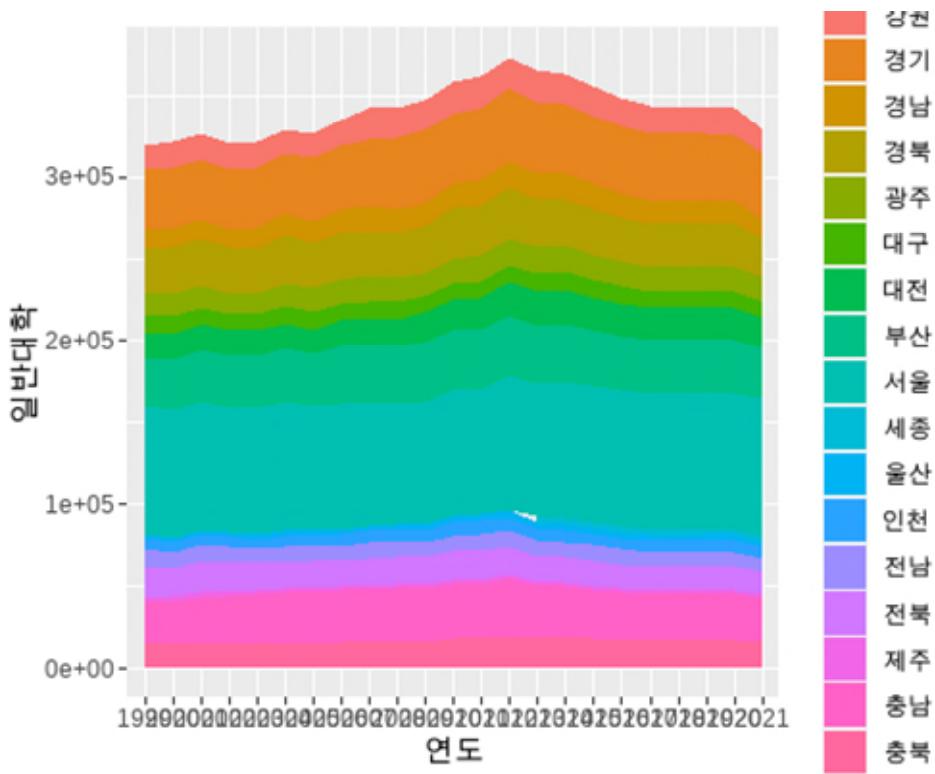
- data : 시각화를 위해 사용될 데이터, 생략되면 ggplot()에 정의된 데이터 사용
- stat : 시각화에 적용될 통계요소, 기본값은 'identity'으로 권장됨
- position : 시각화에 적용될 위치요소, 기본값은 'identity'로 권장됨
- na.rm : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- show.legend : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

geom_area()에서 사용이 가능한 미적요인은 x, y, alpha, color, fill, group, linetype, size 등 이다.

```
## df_입학자 데이터 중 지역이 전체가 아닌 데이터를 필터링하여 ggplot 객체로 생성하고  
p_area에 저장
```

```
p_area <- df_입학자 |> filter(지역 != '전체') |>  
ggplot()
```

```
## p_line 객체에 x 축에 연도, y 축에 일반대학으로 매핑하고 각각의 line 은 지역으로 그루  
핑하고, fill 는 지역으로 매핑한 geom_line 레이어를 생성  
p_area +  
geom_area(aes(x = 연도, y = 일반대학, group = 지역, fill = 지역))
```



2.3.5. 통계 요소

앞서 설명한 기하요소 중에 산점도와 같은 시각화는 데이터의 자체 값을 사용하는 기하요소이고 히스토그램과 같은 시각화는 데이터를 통계 처리한 후 그 결과를 사용하였다. 사실 데이터 자체값을 그대로 사용하는 경우에도 함수의 매개변수로 통계 변환을 뜻하는 매개변수인 `stat` 을 `identity` 로 설정함으로서 그 자체값을 사용하는 통계변환을 지정한다. 따라서 모든 기하요소에는 기본적으로 디폴트 통계 변환 요소가 포함된다.



2.3.5.1. `stat_*`()

`ggplot` 를 사용하는 사용자들은 주로 `geom_*` 함수를 사용하여 시각화의 주요 데이터 레이어를 생성한다. 하지만 `stat_()` 함수를 사용하여 데이터 레이어를 생성할 수도 있다. 모든 `geom_()` 로 생성된 레이어는 `stat_()` 사용하여 동일한 레이어를 생성할 수 있다. 다시

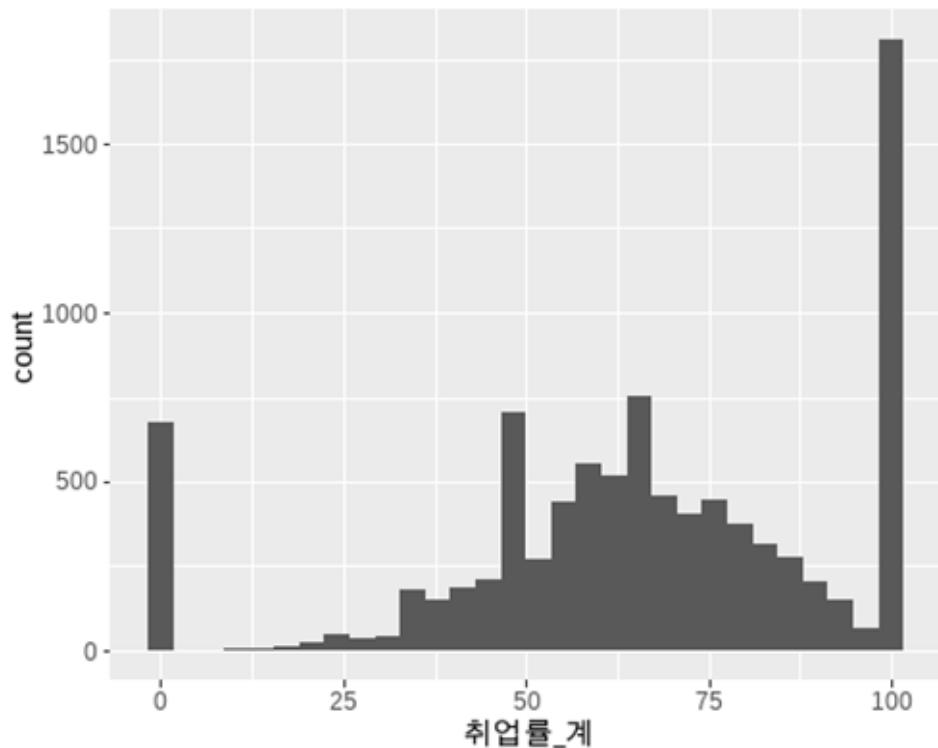
말하자면 `geom_*`()은 데이터가 표현될 도형을 위주로 시각화 레이어를 생성하는 방법이고 `stat_*`()은 데이터의 통계적 변환을 위주로 시각화 레이어를 생성하는 방법이다.

`stat_*`()을 사용하여 `geom_*`()과 동일한 데이터 레이어를 생성하는 방법은 생각보다 간단하다. `geom_*`()의 모든 함수에는 `stat =`이라는 매개변수가 있다. 이 매개변수를 `stat_` 다음에 붙여주고 `stat_*`() 매개변수에 `geom =`에 기하요소 이름을 넣어주면 된다. 이 명명 법칙이 일부 적용되지 않는 함수도 있지만 대부분 아래의 그림과 같이 적용이 가능하다.

```
geom_ 기하요소 이름 (data = 데이터, aes( 미적요소매핑 ), stat = 통계요소 이름, ...)  
stat_ 통계요소 이름 (data = 데이터, aes( 미적요소매핑 ), geom = 기하요소 이름, ...)
```

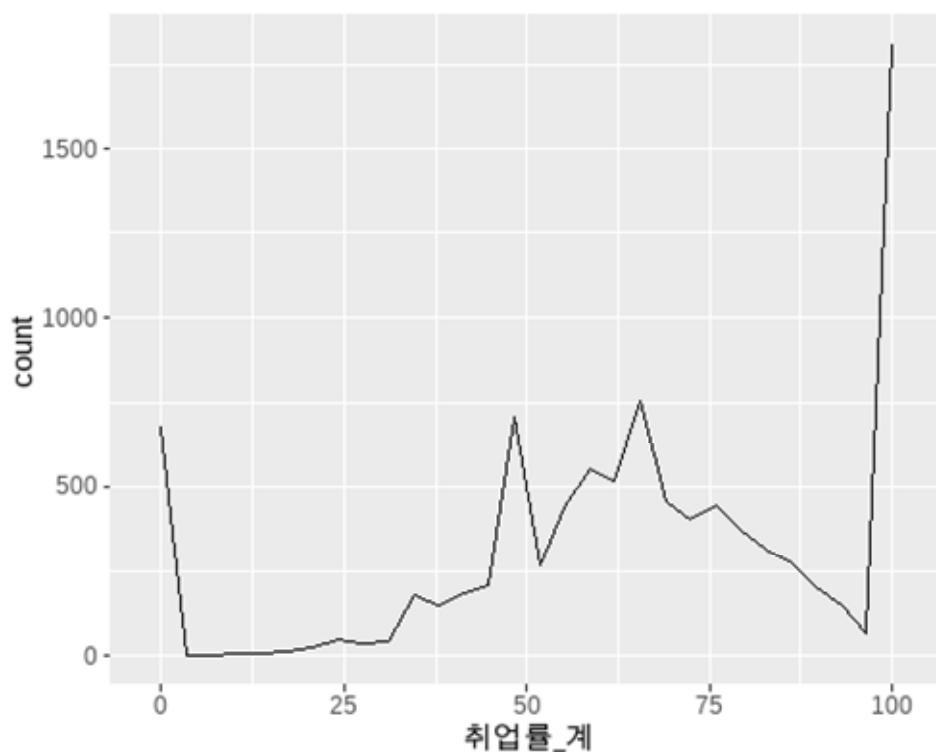
`geom_histogram()`과 동일한 결과

```
p_histogram + stat_bin(aes(x = 취업률_계), geom = 'bar')
```

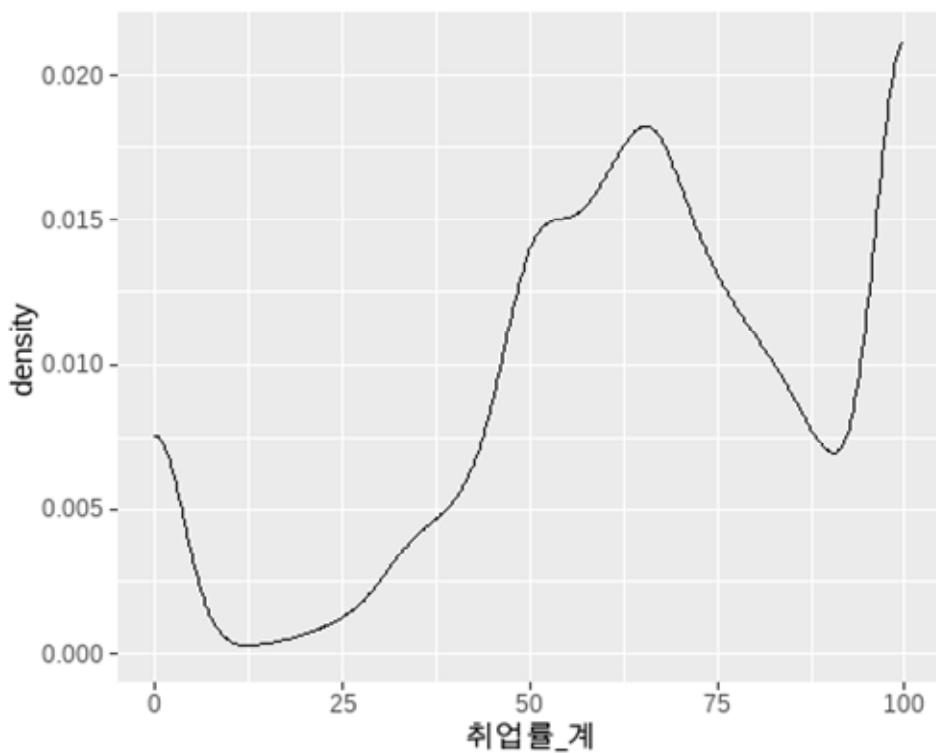


`geom_freqpoly()`과 동일한 결과

```
p_density +  
stat_bin(aes(x = 취업률_계), geom = 'line')
```

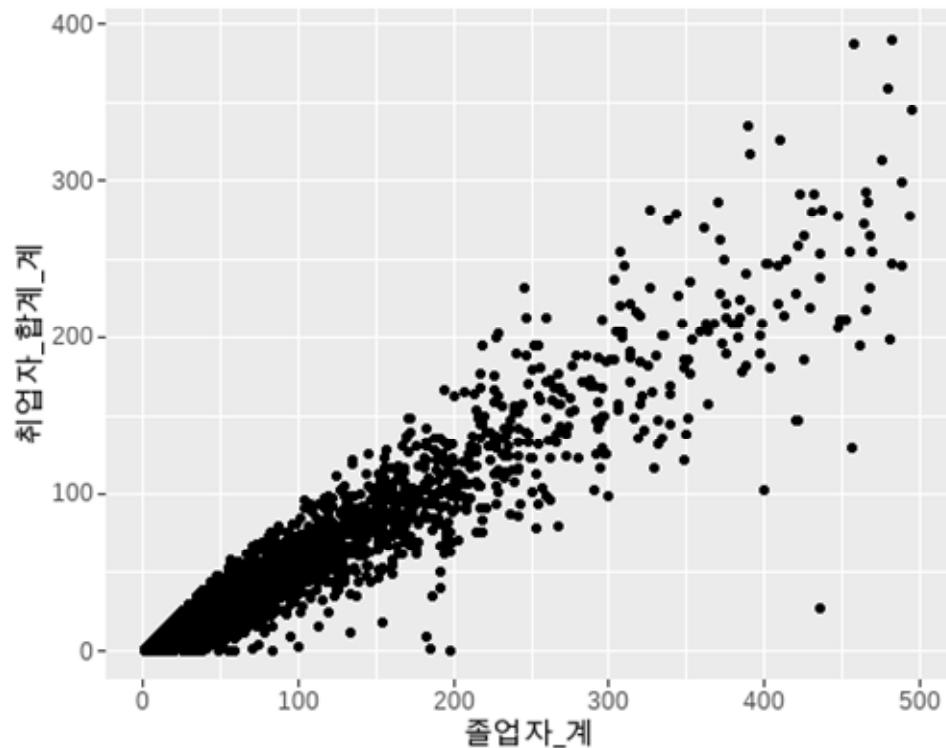


```
## geom_density()과 동일한 결과  
p_density +  
  stat_density(aes(x = 취업률_계), geom = 'line')
```



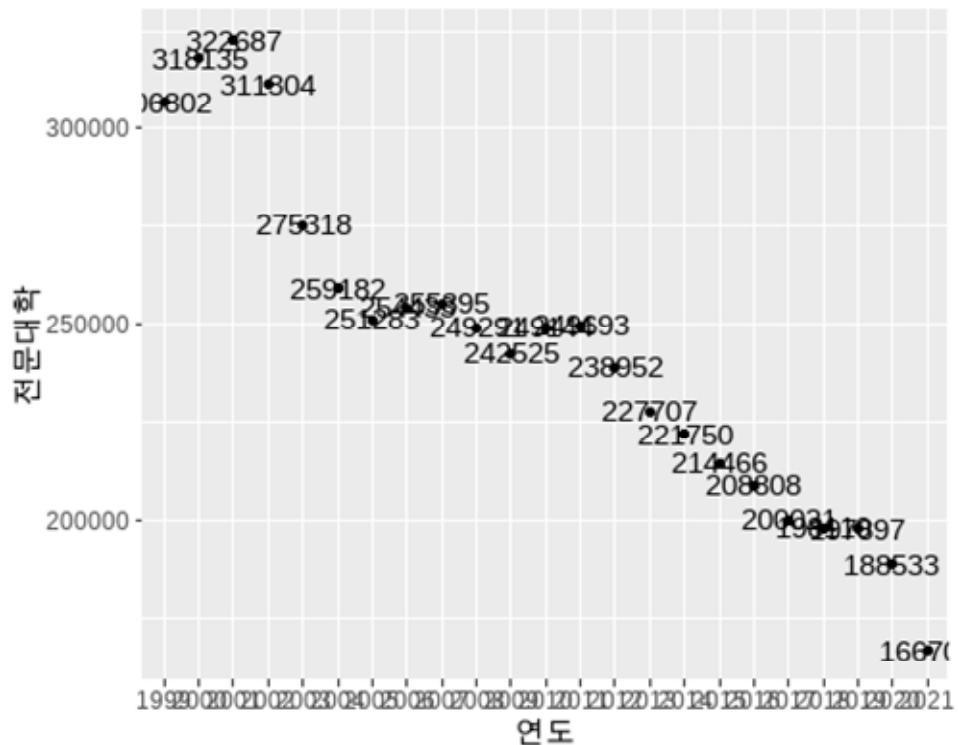
```
## geom_point()과 동일한 결과
```

```
p_point + stat_identity(aes(x = 졸업자_계, y = 취업자_합계_계), geom = 'point')
```



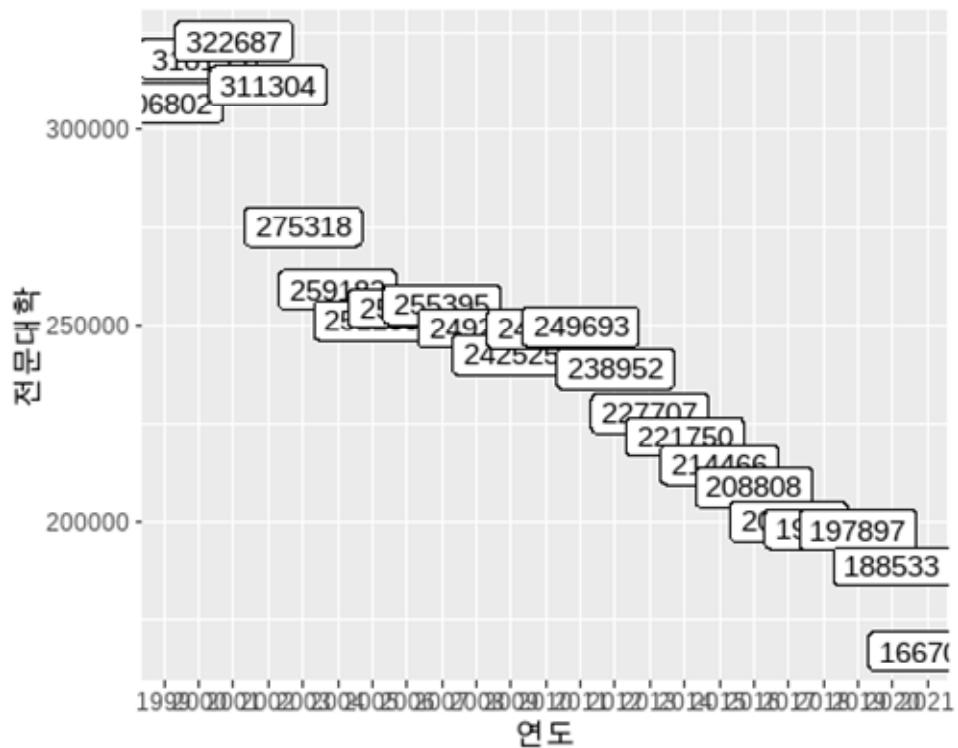
```
## geom_text()과 동일한 결과
```

```
p_text +  
stat_identity(aes(x = 연도, y = 전문대학), geom = 'point') +  
stat_identity(aes(x = 연도, y = 전문대학, label = 전문대학), geom = 'text')
```

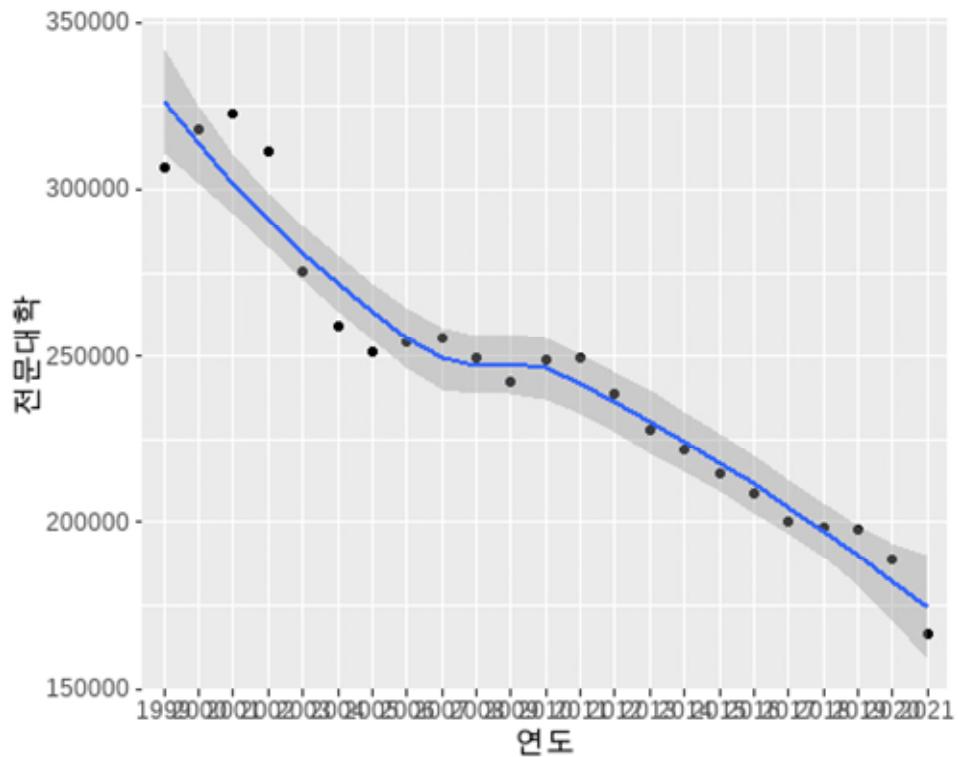


```
## geom_label()과 동일한 결과
```

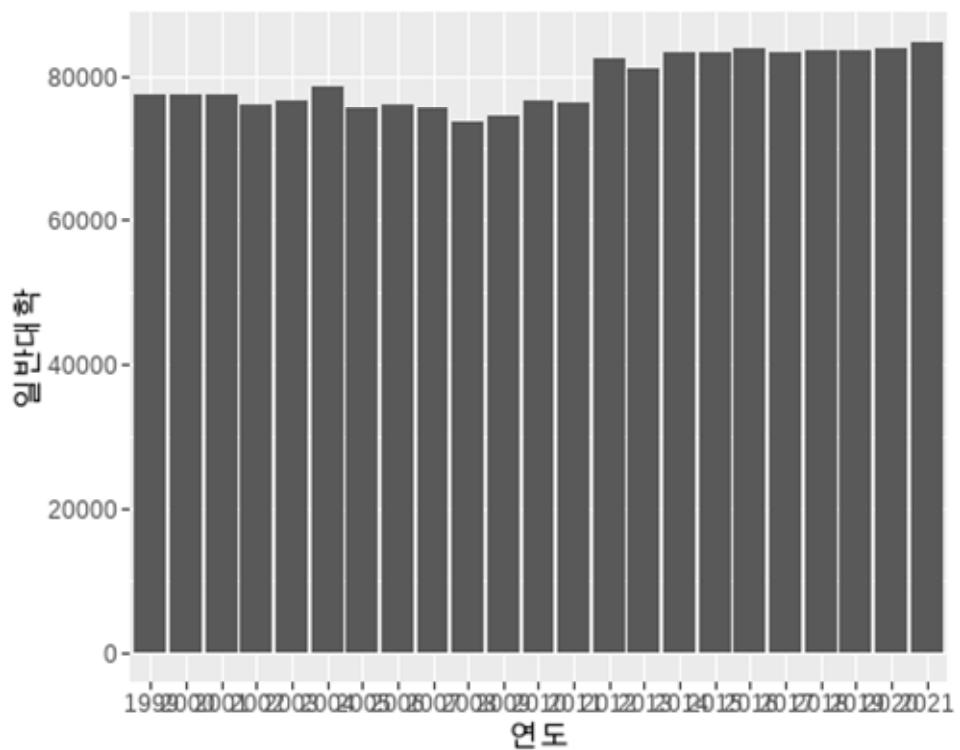
```
p_label +
  stat_identity(aes(x = 연도, y = 전문대학, label = 전문대학), geom = 'label')
```



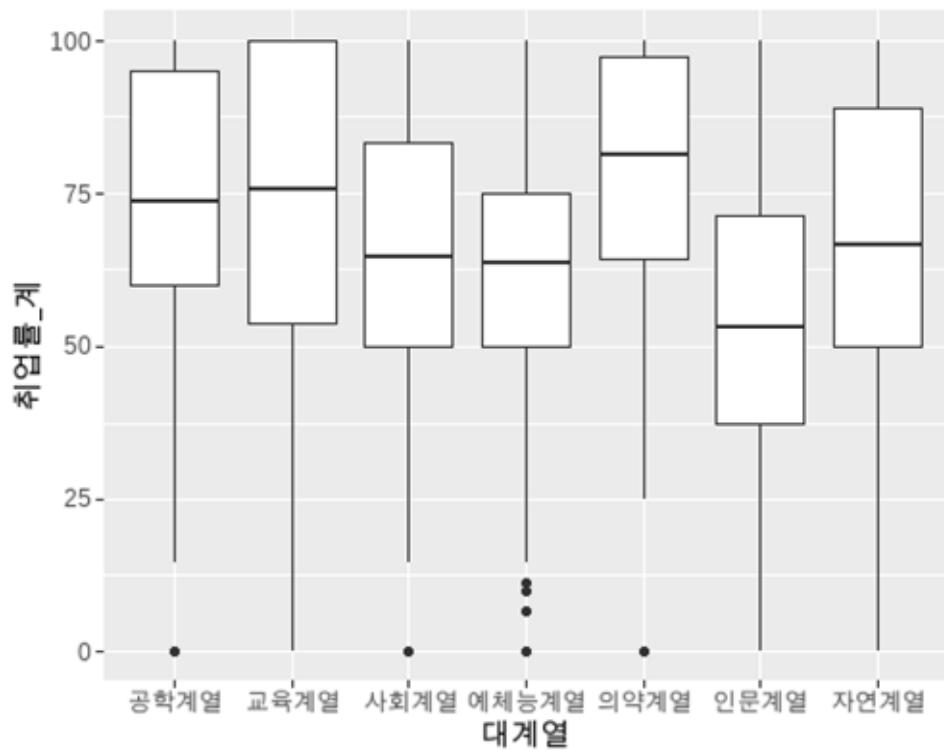
```
## geom_smooth()과 동일한 결과  
p_smooth +  
  stat_identity(aes(x = 연도, y = 전문대학), geom = 'point') +  
  stat_smooth(aes(x = 연도, y = 전문대학, group = 1), method = 'loess', geom = 'smooth')
```



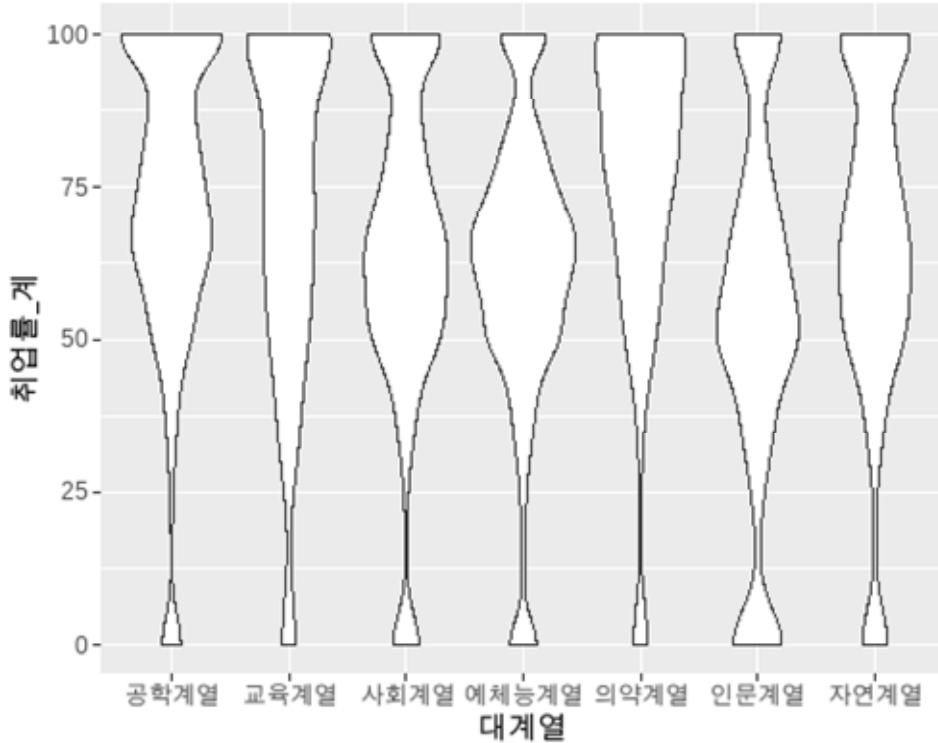
```
## geom_col()과 동일한 결과  
p_col +  
  stat_identity(aes(x = 연도, y = 일반대학), geom = 'bar')
```



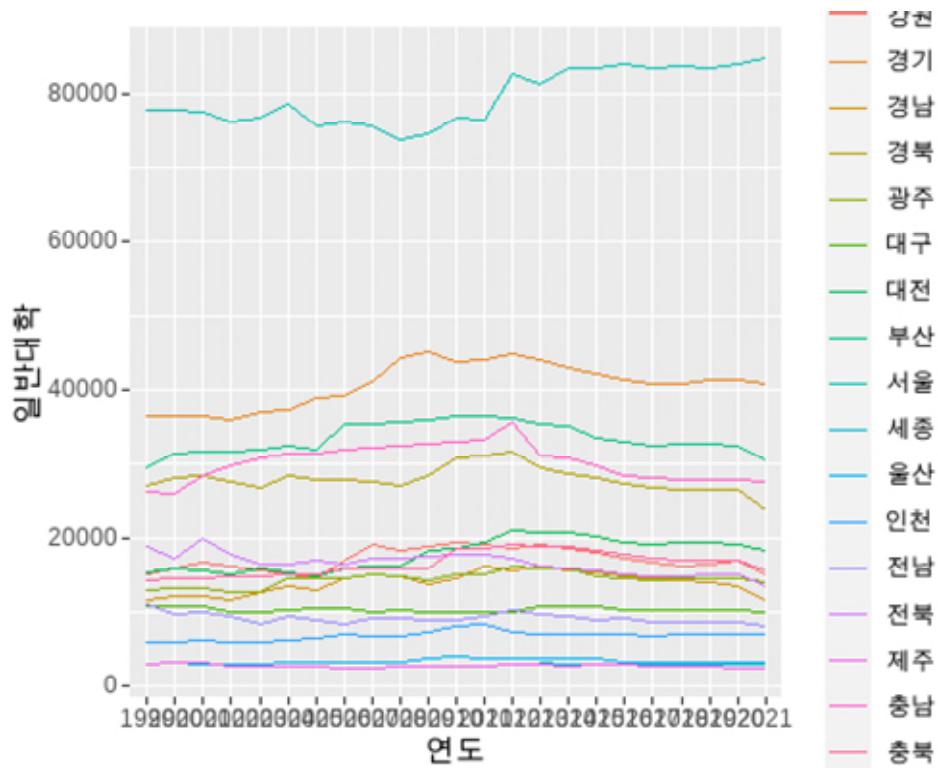
```
## geom_boxplot()과 동일한 결과
p_boxplot +
  stat_boxplot(aes(x = 대계열, y = 취업률_계))
```



```
## geom_violin()과 동일한 결과  
p_violin +  
  stat_ydensity(aes(x = 대계열, y = 취업률_계))
```

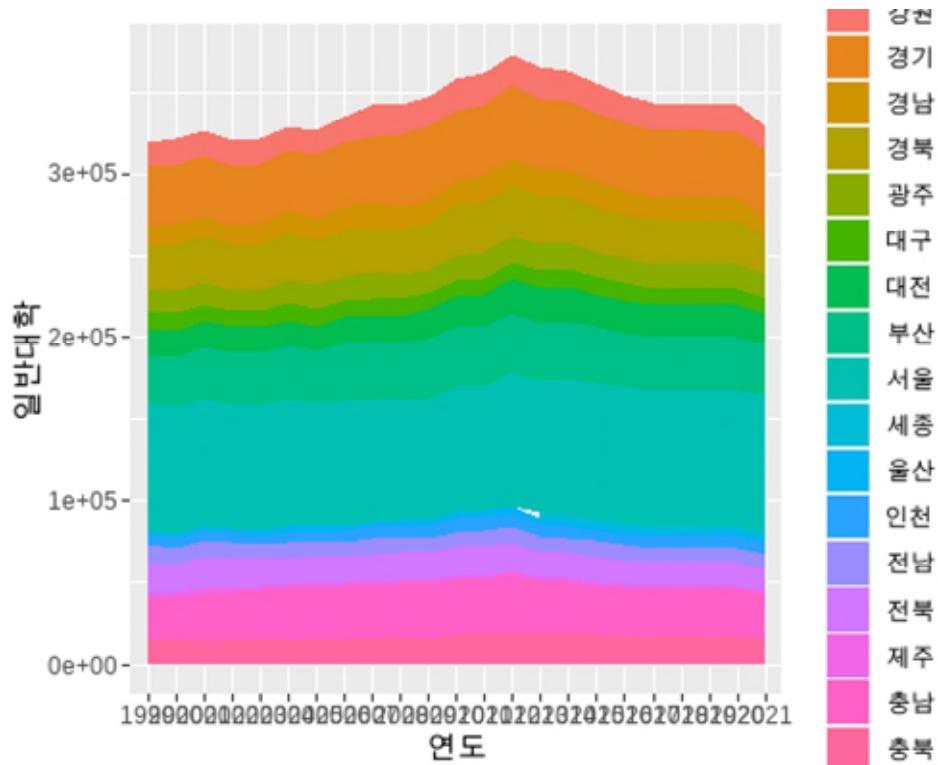


```
## geom_line()과 동일한 결과  
p_line +  
  stat_identity(aes(x = 연도, y = 일반대학, group = 지역, color = 지역), geom = 'line')
```



`geom_area()`과 동일한 결과

```
p_area +
  stat_identity(aes(x = 연도, y = 일반대학, group = 지역, fill = 지역), geom = 'area', position = 'stack')
```



위에서 살펴본 바와 같이 `geom_*`()으로 생성할 수 있는 기하요소 레이어는 `stat_*`()을 사용하여 동일하게 생성할 수 있다. 하지만 `geom_*`()에서는 제공하지 않는 `stat_*`()에서만 제공하는 특별한 레이어가 있다.

2.3.5.2. `stat_summary()`

`geom_*`()으로 생성한 기하요소 레이어에 통계적으로 변환된 데이터를 추가적으로 표현해야 할 경우가 있다. 가장 대표적인 예가 박스 플롯에서 중간값이 자동적으로 표현되지만 평균값을 추가적으로 표현해야 하는 경우가 있다. 이런 경우에 대비하여 특정 함수로 요약한 통계치를 표현하는 레이어를 추가할 때 `stat_summary()`를 사용한다.

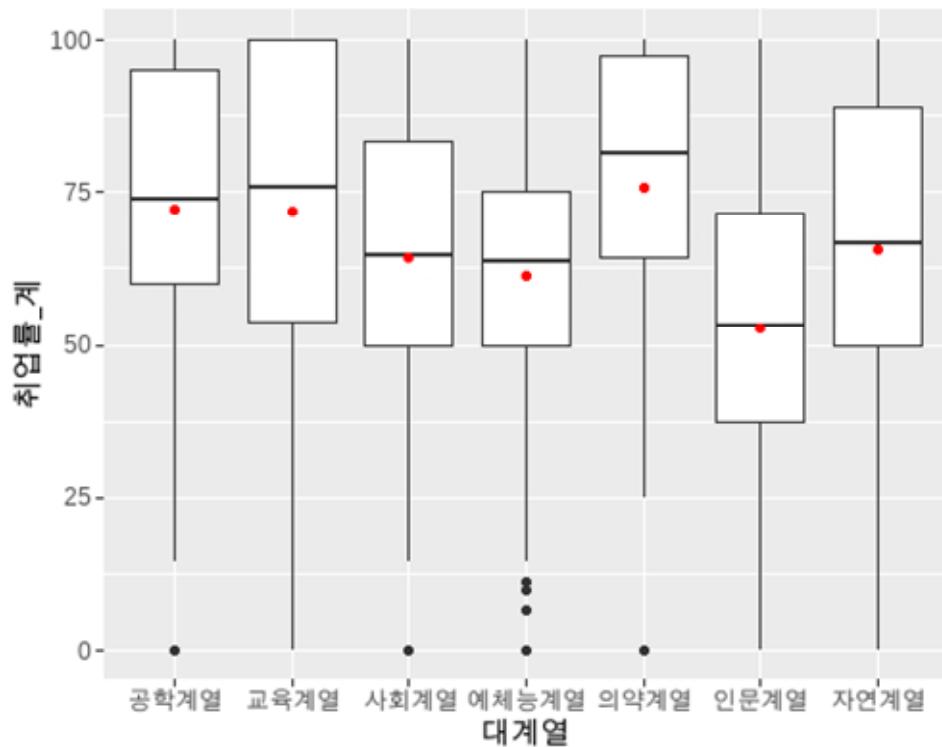
```
stat_summary(mapping = NULL, data = NULL, geom = "pointrange", position = "identity",
             fun.data = NULL, fun = NULL, fun.max = NULL, fun.min = NULL, fun.args = list(),
             na.rm = FALSE, show.legend = NA, fun.y, fun ymin, fun ymax)
```

- `mapping` : `aes()`를 사용하여 매핑할 미적요소, 생략되면 `ggplot()`에 정의된 미적매핑 사용

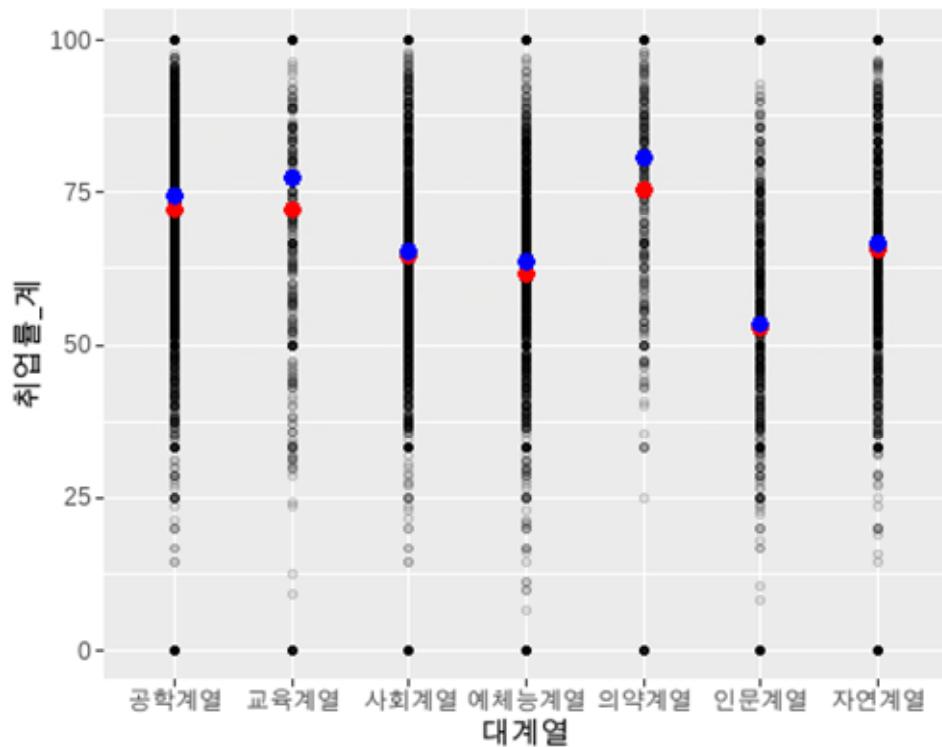
- `data` : 시각화를 위해 사용될 데이터, 생략되면 `ggplot()`에 정의된 데이터 사용
- `geom` : 시각화에 적용될 미적요소를 설정, 기본값은 'pointrange'
- `position` : 시각화에 적용될 위치요소 설정, 기본값은 'identity'
- `fun.data` : `ymin`, `y`, `ymax`을 포함하는 데이터프레임을 반환하는 함수 설정
- `fun`, `fun.y` : `x` 값에 대응하는 `y` 값을 구하는 함수 설정
- `fun.min`, `fun.ymin` : `x`에 대응하는 `y` 값의 최소값 함수 설정
- `fun.max`, `fun.ymax` : `x`에 대응하는 `y` 값의 최대값 함수 설정
- `fun.args` : 함수에 전달될
- `na.rm` : NA 값을 생략할 것인지를 설정하는 논리값(TRUE/FALSE)
- `show.legend` : 범례를 사용할 것인지를 설정하는 논리값(TRUE/FALSE)

아래의 코드는 박스 플롯에 평균을 추가하는 코드이다.

```
## p_boxplot 에 geom_boxplot 레이어를 만들고 x 값에 대한 y 값은 평균(mean), 미적요소는 point, color 는 'red'로 설정한 stat_summary 레이어를 추가
p_boxplot +
  geom_boxplot(aes(x = 대계열, y = 취업률_계)) +
  stat_summary(aes(x = 대계열, y = 취업률_계), fun = 'mean', geom = 'point', color = 'red')
```

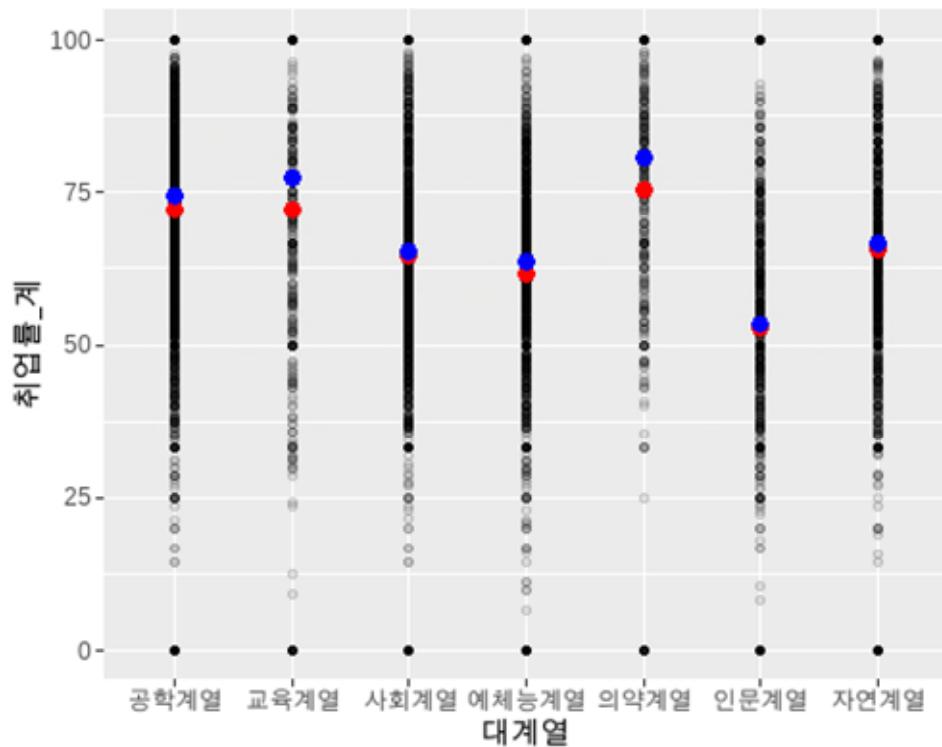


```
## p_point에 geom_point 레이어를 만들고 x 값에 대한 y 값은 평균(mean), 미적요소는 point,
## color는 'red'로 설정한 레이어와 stat_summary 레이어를 추가하는데 x 값에 대한 y
## 값은 평균(mean), 미적요소는 point, color는 'red'로 설정한 레이어를 추가
p_point +
  geom_point(aes(x = 대계열, y = 취업률_계), alpha = 0.1) +
  stat_summary(aes(x = 대계열, y = 취업률_계), fun = 'mean', geom = 'point', color
  = 'red', size = 3) +
  stat_summary(aes(x = 대계열, y = 취업률_계), fun = 'median', geom = 'point', c
  olor = 'blue', size = 3)
```



위의 `stat_summary()`를 `geom_*`()을 사용하여 코딩하면 다음과 같다.

```
p_point +
  geom_point(aes(x = 대계열, y = 취업률_계), alpha = 0.1) +
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun = 'mean', color = 'red', size = 3) +
  geom_point(aes(x = 대계열, y = 취업률_계), stat = 'summary', fun = 'median', color = 'blue', size = 3)
```



2.3.5.3. 통계 변수의 생성

`stat_*`() 레이어를 사용할 때는 먼저 원본 데이터프레임을 적절한 통계처리를 통해 처리한 결과 데이터프레임을 산출하고 이를 시각화 데이터로 사용한다. 그래서 통계처리된 후에는 원본 데이터프레임에는 없었던 새로운 변수(열)이 생성된다. `ggplot` 에서는 이렇게 내부적으로 변환되어 생성된 변수를 미적요소로 매핑이 가능하도록 지원한다. 하지만 내부적으로 계산되어 생성된 변수의 이름을 어떻게 알아낼 수 있을 것인가? `ggplot` 에서는 통계처리되는 방법에 따라 미리 변수 이름을 지정해 놓았기 때문에 이 변수를 쉽게 사용할 수 있다. `ggplot` 이 통계변환을 통해 생성하는 통계 변수는 다음과 같다.

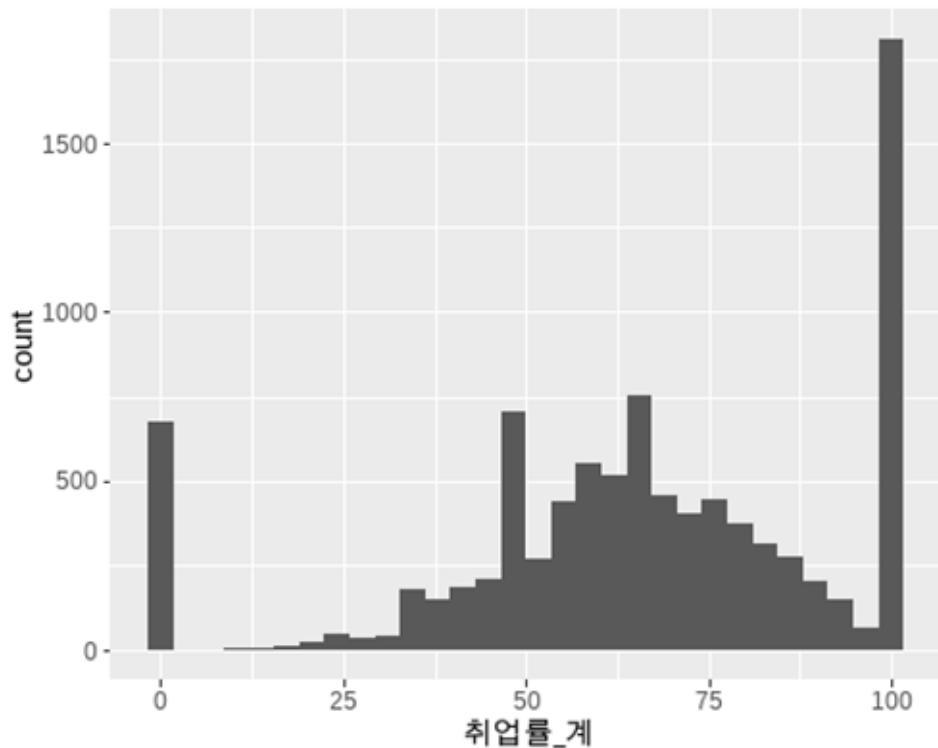
- ..count.. : 각각의 bin 에 해당하는 사례수
- ..density.. : 각각의 bin 에 해당하는 사례의 밀도
- ..x.. : 각각의 bin 의 중간값

사실 앞에서 설명한 일변수 데이터 시각화인 `geom_histogram()`은 y 축에 ..count..가 매핑되어 있고 `geom_density()`에는 ..density.. 가 y 축에 매핑되어 있는 것이다.

다음의 코드를 살펴보자

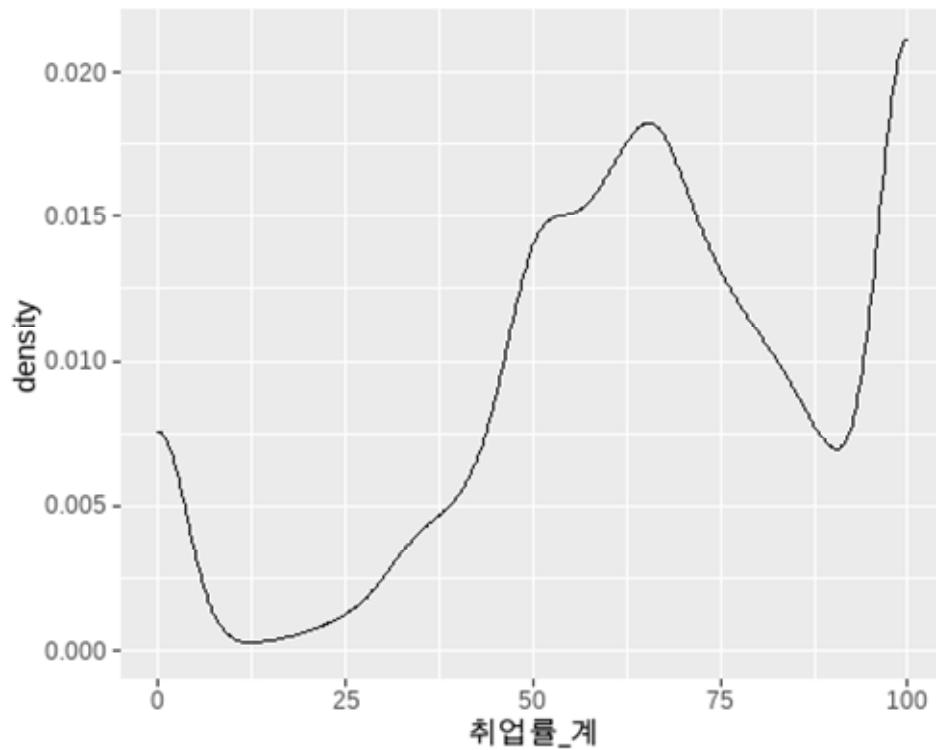
```
## geom_histogram()과 동일한 결과
```

```
p_histogram + geom_histogram(aes(x = 취업률_계, y = ..count..))
```



```
## geom_density()과 동일한 결과
```

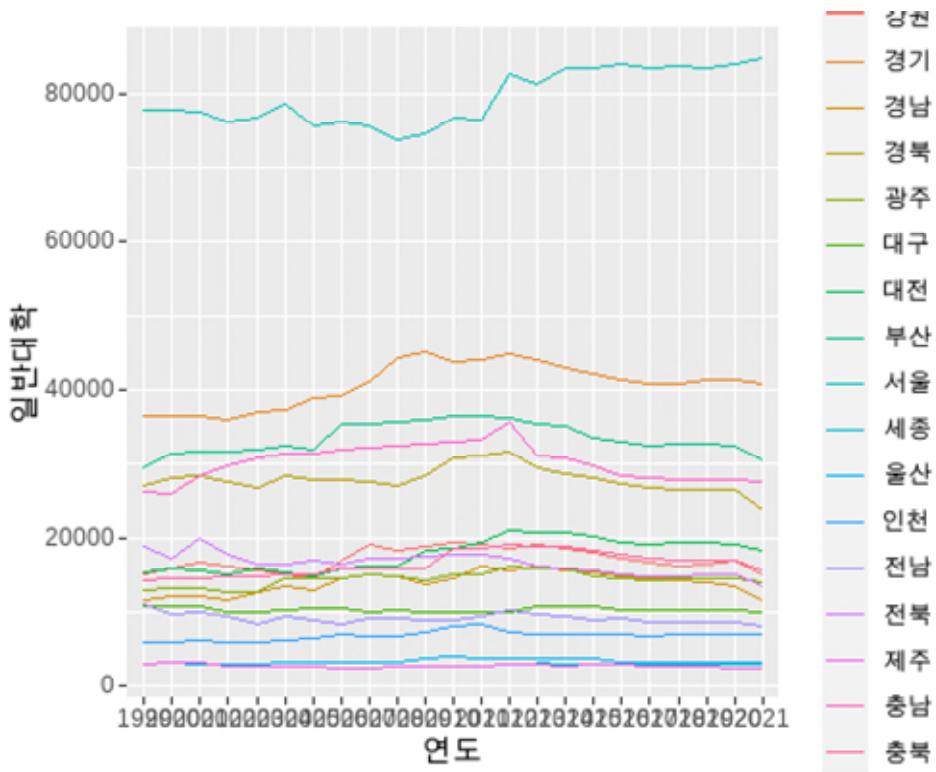
```
p_density + geom_density(aes(x = 취업률_계, y = ..density..))
```



2.4. 분할 요소

팩터와 같은 여러개의 범주(Category)를 가지고 있는 변수를 시각화할 때 하나의 그래프에 표현한다면 동시에 표현되는 기하 요소가 많아지기 때문에 시각화의 효과가 저해되는 상황이 발생한다. 아래의 예를 보자.

```
## geom_histogram()과 동일한 결과  
p_line +  
  geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역))
```



위의 선 그래프는 지역 변수에 포함된 총 17 개 변량에 따라 총 17 개의 선이 표현된다. 맨 위의 선은 겹치는 선이 없으니 잘 보이지만 아래의 선들은 서로 겹쳐서 각각의 변량을 쉽게 확인하기 어렵다. 또 각각의 변량을 구분하기 위해 색으로 구분해 놓았지만 너무 많은 변량이 존재함에 따라 색도 구분하기 어려워진다. 이러한 경우는 각각의 선 그래프를 분할하여 독립된 그래프로 그려주면 데이터를 확인하기가 편해진다. 이러한 경우 사용하는 것이 분할(Facet) 요소이다.

분할 요소는 동시에 표현되는 기하요소의 갯수가 많아져 데이터 시각화가 효과적이지 않은 경우 각각의 기하요소를 분리하여 작은 그래프로 분할하여 그려주는 방법을 말한다. 이 분할 요소로 그려지는 분할 그래프들은 동시에 여러개의 데이터 패턴을 파악할 수 있기 때문에 작지만 매우 강력한 시각화 방법이다. 이 분할 요소는 구현하는 것은 `facet_wrap()`과 `facet_grid()`의 두 가지 방법이 있다.

`facet_wrap()`과 `facet_grid()`는 모두 주어진 변수에 따라 그래프를 분할하는 함수이다. 다만 두 함수의 차이는 분할 변수에 따라 표현되는 서브 그래프의 순서가 어떻게 배열되는지에 따라 달라진다. 배열 순서를 설명하기 위해서는 분할 변수를 지정하는 방법에 대해 먼저 알아야 한다.

분할 변수를 지정할 때는 R에서 독립변수와 종속변수의 관계를 뜻하는 틸드(~)를 사용하여 지정한다. 예를 들어 'A ~ B'라고 표현되면 독립변수 B에 대한 종속변수 A라는 의미이다. `facet_grid()` 이와 같이 독립변수와 종속변수 표현식을 사용하여 분할변수를 지정한다. 반면 `facet_wrap()`은 '~B'와 같이 독립변수만을 지정한다.

'A ~ B'로 표현된 `facet_grid()`은 Y 축으로 A 변수, X 축으로 B 변수의 순서로 표현된다. 만약 'A ~ .'로만 표현된다면 Y 축의 방향으로 A 변수 순서로 표현되고 '. ~ B'로 표현된다면 X 축의 방향으로 B 변수 순서로 표현된다. `facet_wrap()`은 독립변수만을 표현하기 때문에 '~ A'와 같이 분할 변수를 지정한다.

이와 같이 독립변수와 종속변수의 관계를 표현하는 상관식으로 표현하는 것이 일반적이지만 매개변수(rows, cols)와 `vars()`를 이용하는 방법도 있다. `vars()`는 `ggplot` 객체에 포함된 데이터프레임의 일부 변수를 선택하는 함수이다. 사실 `ggplot2`가 계속해서 버전업되면서 현재는 앞의 상관식을 사용하는 것보다는 이 방법을 권장하고 있다. 이 방법을 사용할 때 주의해야 하는 것은 `facet_grid()`과 `facet_wrap()`의 함수 용법이 조금 다르다는 것이다.

2.4.1. `facet_wrap()`

`facet_wrap()`의 사용법과 주요 매개변수는 다음과 같다.

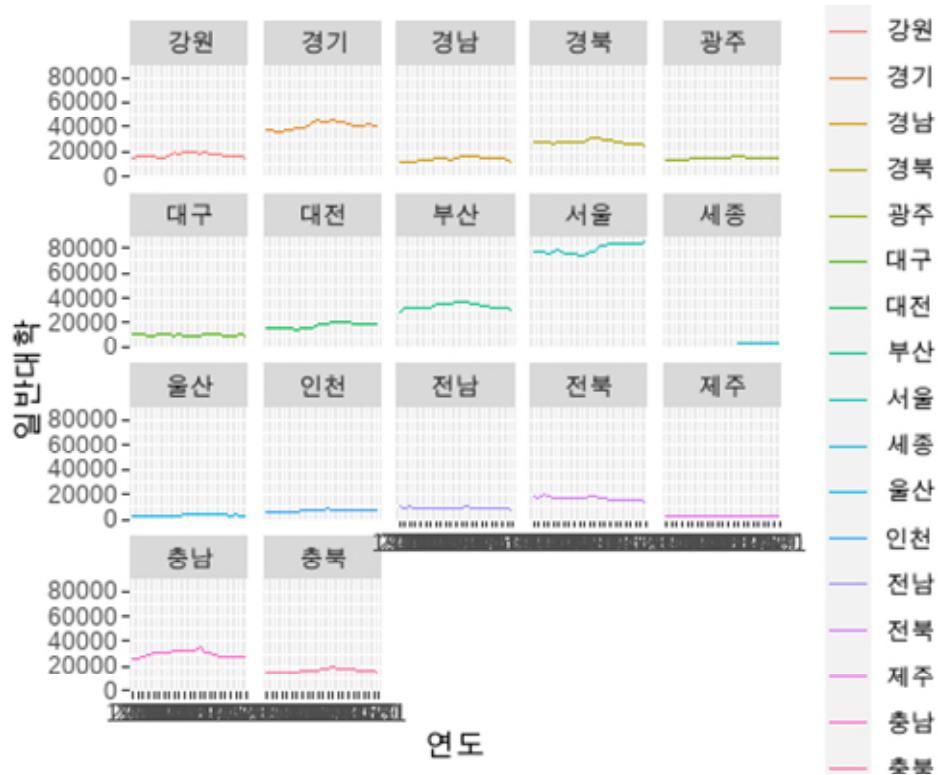
```
facet_wrap(facets, nrow = NULL, ncol = NULL, scales = "fixed", labeller = "label_value", as.table = TRUE, strip.position = "top", ...)
```

- `facets` : 시각화를 분할할 때 사용될 분할 변수, `vars()`를 이용하거나 변수명 앞에 틸드(~)를 붙임.
- `nrow` : 행의 개수를 고정
- `ncol` : 열의 개수를 고정
- `scales` : 각각의 분할 그래프의 X, Y 축의 축척을 전체적으로 고정('fixed')할지 개별 축적으로 설정('free_y', 'free_x')할 것인지 결정
- `labeller` : 서브 그래프의 라벨을 설정
- `as.table` : 서브 그래프의 순서를 bottom-right로 할지 top-right로 할지를 결정하는 논리값(TRUE/FALSE)
- `strip.position` : 서브 그래프 라벨의 위치 설정

`facet_wrap()`은 다음과 같이 사용할 수 있다.

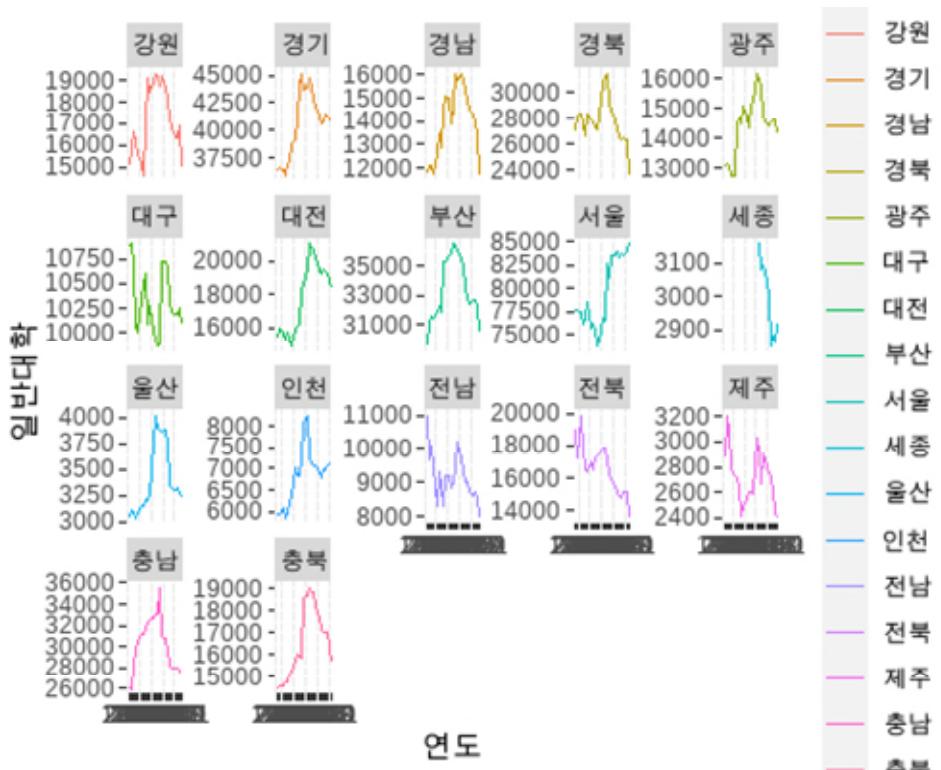
```
## p_line 객체에 x 축은 연도, y 축은 일반대학, group과 color를 지역으로 매핑한 geom_line 레이어를 생성하고 지역별로 시각화를 분할(vars() 사용)
p_line +
```

```
geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역)) +
facet_wrap(vars(지역))
```



위의 예에서 전체 Y 축의 축척(Scale)을 전체적으로 고정시켰기 때문에 전반적인 데이터의 수준은 어느 지역이 높고 어느 지역이 낮은지가 눈에 보이지만 지역별로 데이터의 흐름이 잘 보이지 않는다. 이를 위해 다음과 같이 Y 축의 축척을 풀어줄 수 있다.

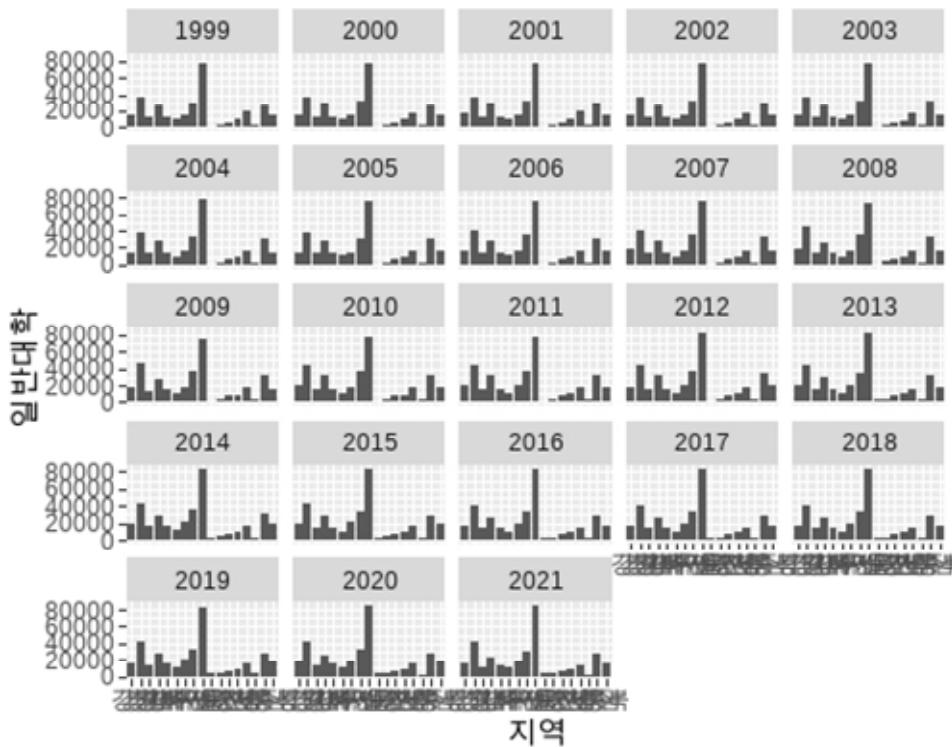
```
## p_line 객체에 x 축은 연도, y 축은 일반대학, group 과 color 를 지역으로 매핑한 geom_
line 레이어를 생성하고 지역별로 시각화를 분할(vars() 사용)
p_line +
  geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역)) +
  facet_wrap(vars(지역), scales = 'free_y')
```



아래는 지역과 연도를 바꾸어 막대그래프로 표현한 분할 시각화이다.

```
## p_col 객체에 x 축은 지역, y 축은 일반대학으로 매핑한 geom_line 레이어를 생성하고 연도별로 시각화를 분할(~ 사용)
```

```
p_col +
  geom_col(aes(x = 지역, y = 일반대학)) +
  facet_wrap(~연도)
```



2.4.2. `facet_grid()`

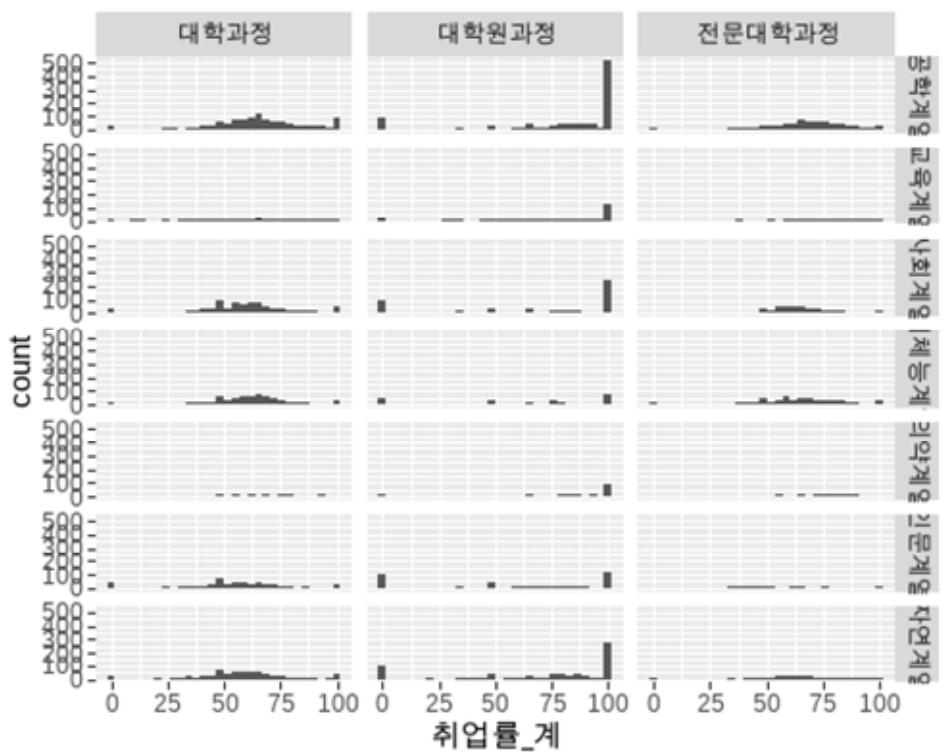
`facet_grid()`의 사용법과 주요 매개변수는 다음과 같다.

```
facet_grid(rows = NULL, cols = NULL, scales = "fixed", labeller = "label_value",
           as.table = TRUE, facets, ...)

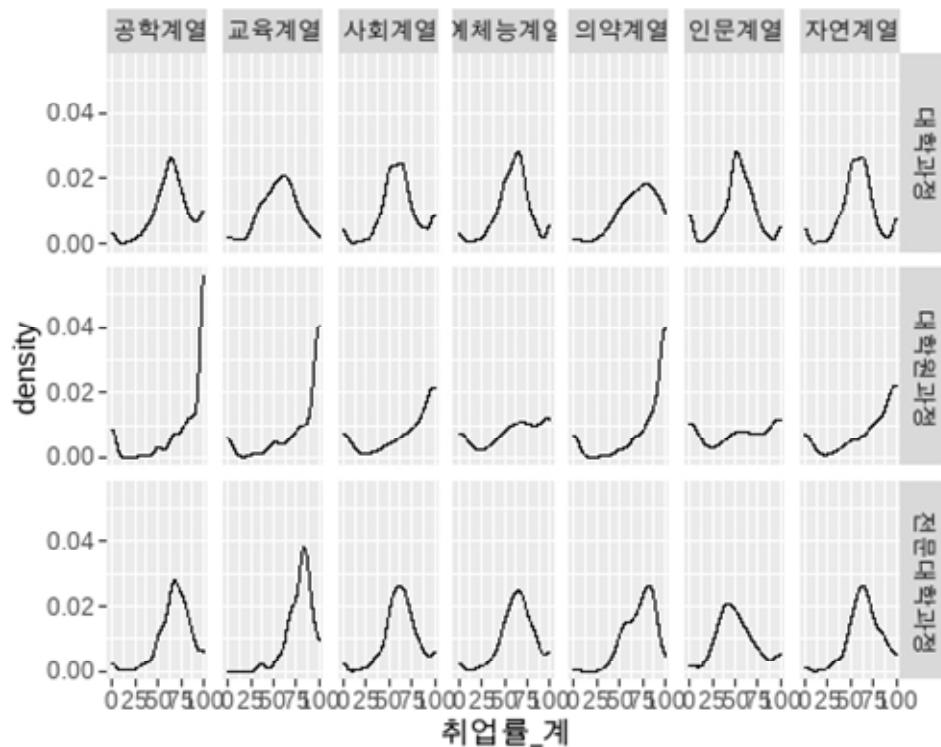
- rows : 행의 방향으로 설정될 변수 설정
- cols : 열의 방향으로 설정될 변수 설정
- scales : 각각의 분할 그래프의 x, y 축의 축척을 전체적으로 고정('fixed')할지 개별
축적으로 설정('free_y', 'free_x')할 것인지 결정
- labeller : 서브 그래프의 라벨을 설정
- as.table : 서브 그래프의 순서를 bottom-right로 할지 top-right로 할지를 결정하는
논리값(TRUE/FALSE)
- facets : 시각화를 분할할 때 사용될 분할 변수식, 현재는 권장되지 않음
```

`facet_grid()`은 다음과 같이 사용할 수 있다.

```
## p_histogram 객체에 x 축은 취업률_계로 맵핑한 geom_histogram 레이어를 생성하고 x 축
방향으로 대계열, y 축방향으로 과정구분으로 시각화를 분할(vars() 사용)
p_histogram +
  geom_histogram(aes(x = 취업률_계)) +
  facet_grid(rows = vars(대계열), cols = vars(과정구분))
```



```
## p_col 객체에 x 축은 지역, y 축은 일반대학으로 매핑한 geom_line 레이어를 생성하고 연
도별로 시각화를 분할(~ 사용)
p_density +
  geom_density(aes(x = 취업률_계)) +
  facet_grid(과정구분 ~ 대계열)
```



2.5. 좌표 요소

좌표요소는 사실 크게 많이 활용되는 요소는 아닌듯하다. 청중들은 X, Y 축으로 구성된 2 차원 좌표인 데카르트(Cartesian) 좌표계에 너무 익숙하기 때문에 다른 좌표계를 사용하는 시각화는 다소 어색하다. 하지만 `ggplot` 에서 핵심으로 제공하는 요소이니만큼 간략하게나마 언급하겠다.

`ggplot` 에서는 앞서 설명한 바와 같이 X, Y 축이 동일한 축척을 가진 좌표계로 이루어진 선형(Linear) 좌표계와 각도(angle)과 반지름(radius)의 좌표계, Log 나 지수와 같은 변환으로 이루어진 비선형 (Non Linear)좌표계를 제공한다.

2.5.1. 선형 좌표계

`ggplot` 에서는 선형 좌표계 함수로 `coord_cartesian()`, `coord_flip()`, `coord_fixed()`의 세 가지를 제공한다.

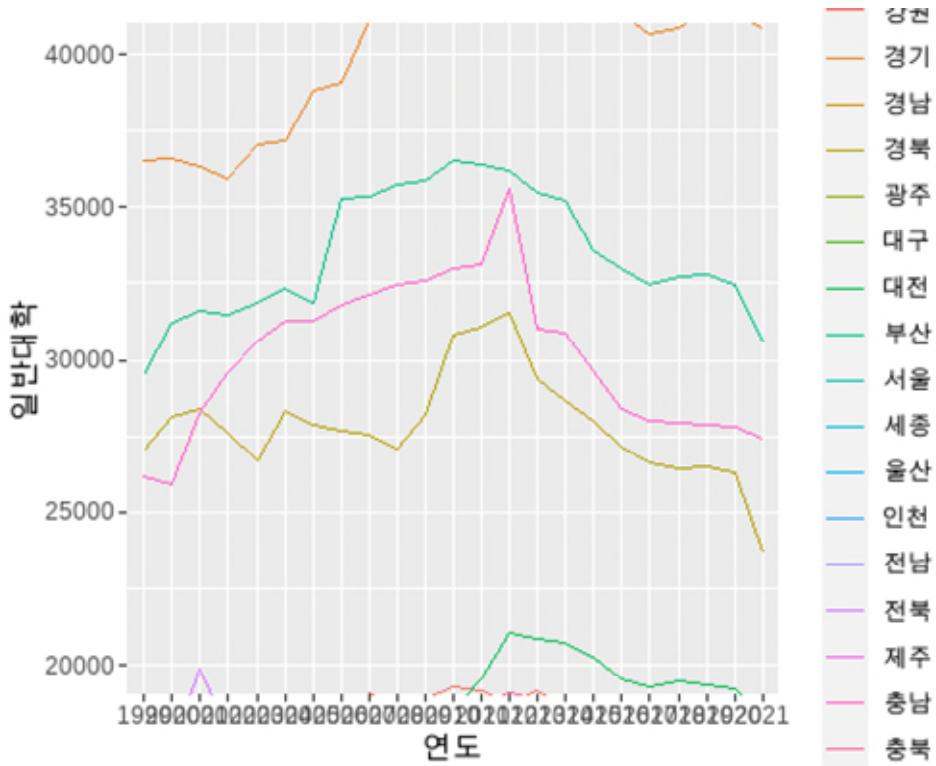
2.5.1.1. coord_cartesian()

`coord_cartesian()`으로 제공되는 좌표계는 가장 일반적이고 대중적으로 사용되는 2 차원 좌표계이다. 이 함수를 사용해서 X 축과 Y 축의 범위를 설정할 수 있고 이 범위의 설정은 Zoom 과 같이 특정 부분을 확대하거나 축소할 수 있다.

```
coord_cartesian(xlim = NULL, ylim = NULL, expand = TRUE, ...)  
  - xlim, ylim : X 축과 Y 축의 범위를 설정, 최소와 최대값의 벡터로 전달  
  - expand : X 축과 Y 축의 범위를 조금 넓게 확장해줄 수 있는지 결정하는 논리값(TRUE/FALSE)
```

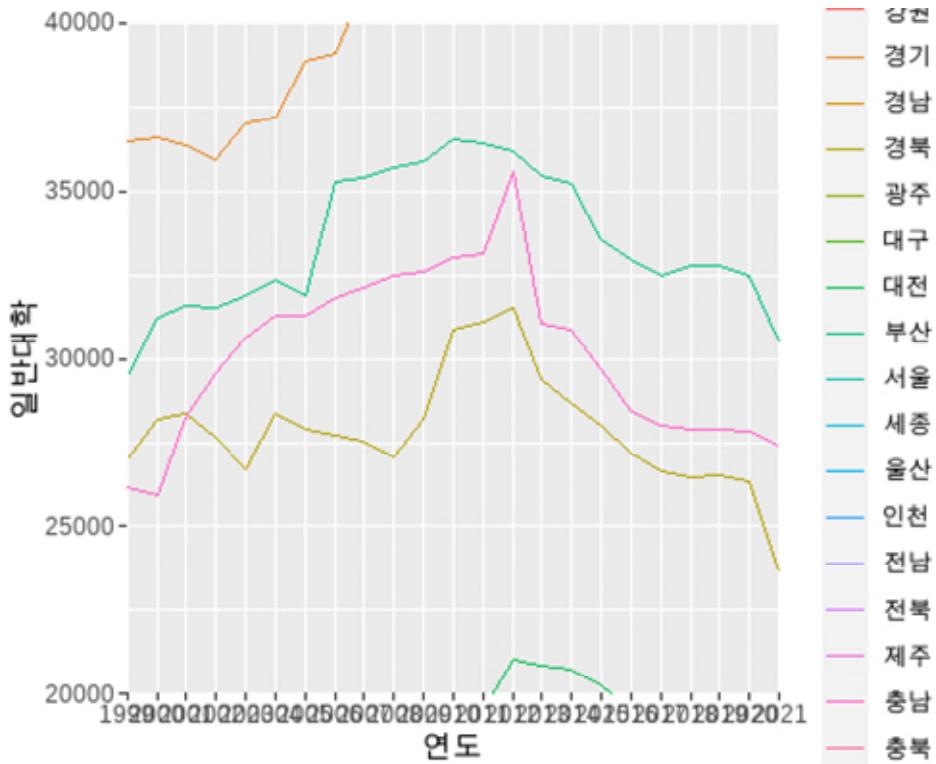
`coord_cartesian()`은 다음과 같이 사용할 수 있다.

```
## p_line 객체에 x 축은 연도, y 축은 일반대학, group 과 color 는 지역으로 매핑된 geom_line 레이어를 생성하는데 y 축의 범위를 20000에서 40000으로 설정  
p_line +  
  geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역)) +  
  coord_cartesian(ylim = c(20000, 40000))
```



```
## p_line 객체에 x 축은 연도, y 축은 일반대학, group 과 color 는 지역으로 매핑된 geom_line 레이어를 생성하는데 y 축의 범위를 20000에서 40000으로 설정하고 expand 를 FALSE 로 설정하여 확장을 제한  
p_line +
```

```
geom_line(aes(x = 연도, y = 일반대학, group = 지역, color = 지역)) +
coord_cartesian(ylim = c(20000, 40000), expand = FALSE)
```



2.5.1.2. coord_flip()

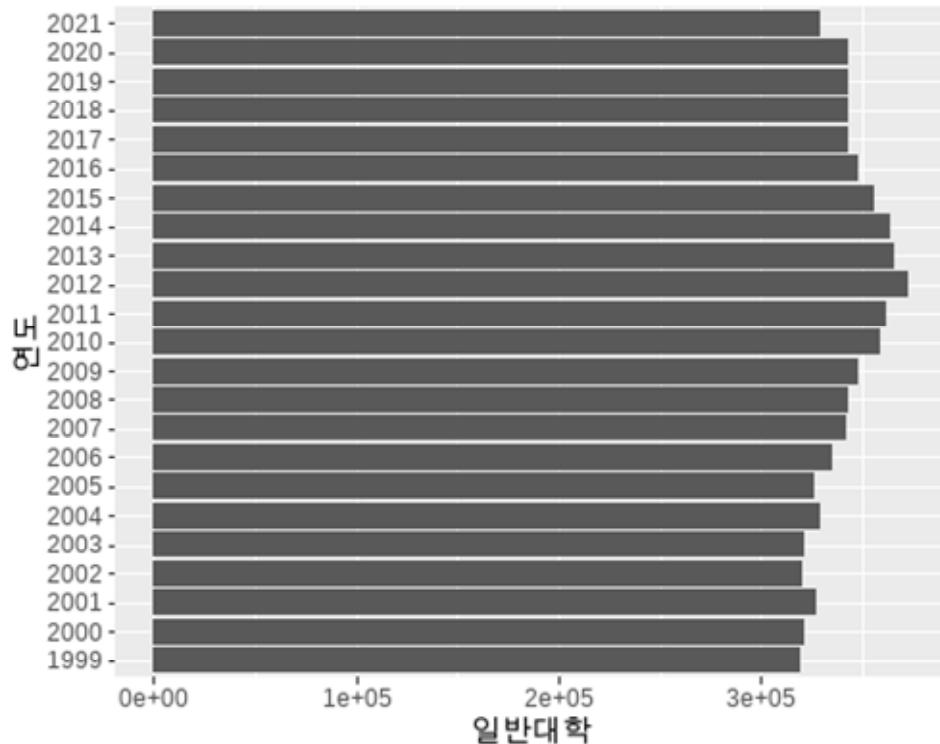
`coord_flip()`에서 제공하는 좌표계는 X 축과 Y 축이 서로 뒤바꾸어 표현되는 좌표계이다. X 축과 Y 축의 표현만이 바뀌는 것이지 맵핑은 바뀌지 않기 때문에 이 부분을 잘 고려해야 한다. 보통 X 축과 Y 축에 맵핑되는 데이터의 경우 주어진 X 값에 따라 산출된 Y 값으로 설정하는 경우가 많다. 하지만 주어진 Y 값에 대해 X 값을 표현해야 하는 경우 `coord_flip()`을 사용하여 축을 틀어 표현하거나 가로 형태로 길게 표현되어야 하는 시각화의 경우 X 축과 Y 축을 바꾸어 표현하는 방법을 사용한다.

```
coord_flip(xlim = NULL, ylim = NULL, expand = TRUE, ...)
  - xlim, ylim : X 축과 Y 축의 범위를 설정, 최소와 최대값의 벡터로 전달
  - expand : X 축과 Y 축의 범위를 조금 넓게 확장해줄 수 있는지 결정하는 논리값(TRUE/FALSE)
```

`coord_flip()`은 다음과 같이 사용할 수 있다.

```
## p_col 객체에 x 축은 연도, y 축은 일반대학이 맵핑된 geom_col 레이어를 생성하는데 축을
# 바꾸어 줌
p_col +
```

```
geom_col(aes(x = 연도, y = 일반대학)) +  
coord_flip()
```



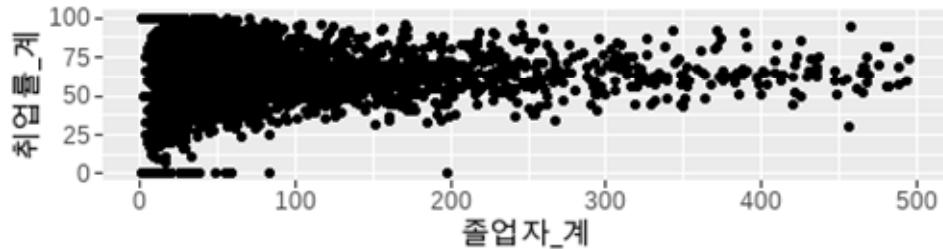
2.5.1.3. coord_fixed()

`coord_fixed()`는 X 축과 Y 축의 비율을 고정시킨 좌표계를 제공한다. 따라서 `coord_fixed()`에서 사용되는 X 축과 Y 축은 유사한 범위를 가진 데이터로 매핑하는 것이 좋다. 예를 들어 X 축을 0 부터 100 까지 데이터가 표현되는 백분률을 표현할 경우 Y 축도 이와 유사한 데이터가 매핑되도록 해야한다. 그런데 둘다 백분률 데이터라고 해도 하나의 데이터는 0 과 100 까지의 범위, 하나의 데이터는 0 과 1 까지의 범위라면 두 데이터의 비율이 1:1 이 아닌 1:0.01 이 될 것이기 때문에 데이터가 정확히 표현되지 않을 것이다. 이런 경우에는 `ratio` 매개변수를 사용하여 비율을 조절할 필요가 있다.

```
coord_fixed(ratio = 1, xlim = NULL, ylim = NULL, expand = TRUE, ...)  
- ratio : X 축과 Y 축의 비율을 설정  
- xlim, ylim : X 축과 Y 축의 범위를 설정, 최소와 최대값의 벡터로 전달  
- expand : X 축과 Y 축의 범위를 조금 넓게 확장해줄 수 있는지 결정하는 논리값(TRUE/FALSE)
```

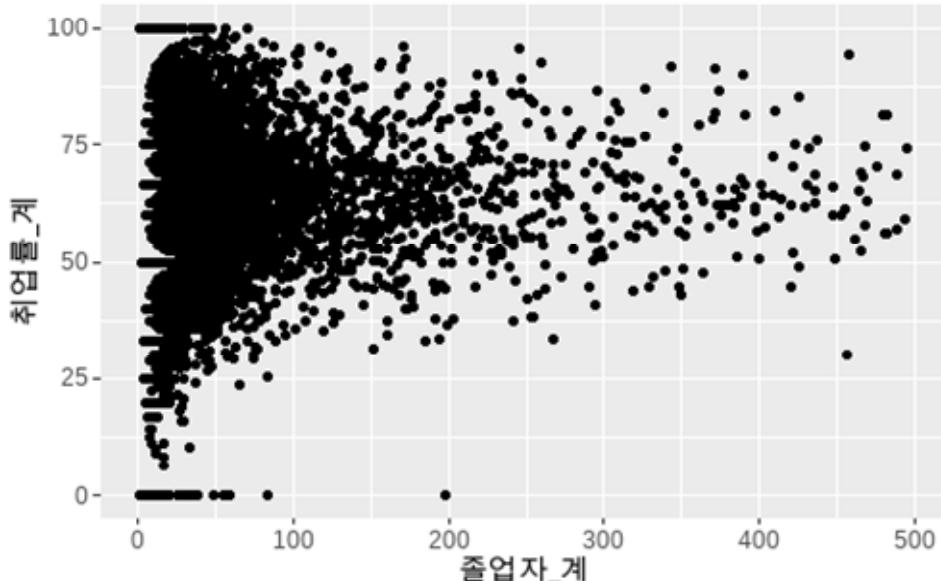
아래의 코드는 `coord_fixed()`의 사용 예제이다. y 축에 매핑된 데이터가 0 부터 100 까지의 백분율 데이터이기 때문에 x 축의 0 부터 100 까지의 길이에 해당하는 만큼의 Y 축만 표현된다.

```
## p_point 객체에 x 축이 졸업자_계, y 축은 취업률_계로 매핑된 geom_point 레이어를 생성하고 양쪽 축의 축척을 1:1로 고정
p_point +
  geom_point(aes(x = 졸업자_계, y = 취업률_계)) +
  coord_fixed()
```



위와 같이 Y 축이 좁아진 것을 넓혀주려면 `ratio` 매개변수를 조절한다.

```
## p_point 객체에 x 축이 졸업자_계, y 축은 취업률_계로 매핑된 geom_point 레이어를 생성하고 양쪽 축의 축척을 3:1로 고정
p_point +
  geom_point(aes(x = 졸업자_계, y = 취업률_계)) +
  coord_fixed(ratio = 3)
```



2.5.2. 비선형 좌표계

비선형 좌표계는 X 축과 Y 축의 직선형태로 표현되는 좌표계가 아닌 둥글게 표현되는 극 좌표계와 X 축과 Y 축으로 표현은 되지만 길이의 축척이 달라지는 축 변환에 의한 좌표계를 말한다. 극 좌표계는 `coord_polar()`를 사용하고 축 변환은 `coord_trans()`이 사용된다.

2.5.2.1. `coord_trans()`

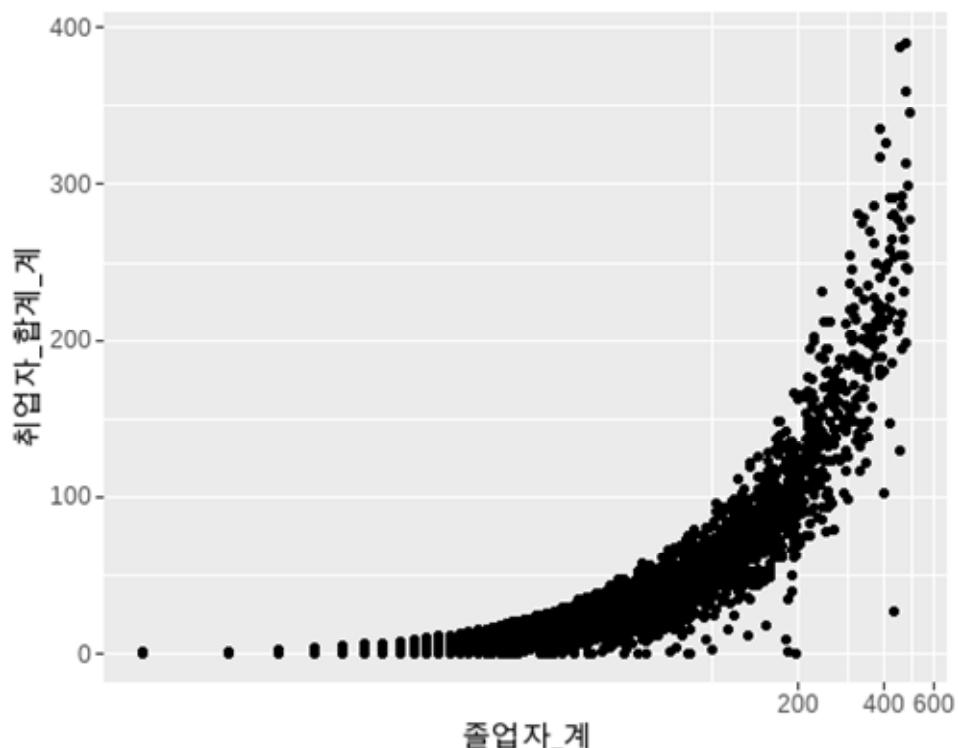
선형이라는 의미는 직선상으로 표현되는 데이터가 어느 위치에서나 같은 데이터 간격은 같은 길이에 표현된다. 따라서 비선형이라는 의미는 직선상에 표현되는 데이터라 하더라도 위치에 따라 같은 데이터 간격이 다른 길이에 표현된다는 것을 의미한다. 비선형 좌표계를 위한 변환은 여러가지가 있지만 가장 흔히 사용되는 비선형 좌표계는 로그변환을 통한 로가리듬(logarithm)이나 지수변환이다. 그외에도 사용자 정의 변환도 사용할 수 있다.

```
coord_trans(x = "identity", y = "identity", xlim = NULL, ylim = NULL, expand = TRUE, ...)
```

- `x, y` : X 축과 Y 축의 변환 이름
- `xlim, ylim` : X 축과 Y 축의 범위 지정
- `expand` : X 축과 Y 축의 범위를 조금 넓게 확장해줄 수 있는지 결정하는 논리값(TRUE/FALSE)

앞에서 시각화했던 `geom_point()`의 결과를 보면 출업자 `coord_trans()`는 다음과 같이 사용할 수 있다.

```
## p_point 객체에 x 축은 출업자_계, y 축은 취업자_합계_계로 매핑된 geom_point 레이어를  
생성하는데 x 축을 'log10' 변환  
p_point +  
  geom_point(aes(x = 출업자_계, y = 취업자_합계_계)) +  
  coord_trans(x = 'log10')
```



위의 예에서 보면 Y 축의 경우 0 부터 100 까지의 거리나 200 에서 300 까지의 거리는 동일하다. 하지만 X 축의 경우 0 부터 100 까지의 거리와 500 부터 600 까지의 거리가 다르다. Y 축은 선형 좌표계이고 X 축은 비선형 좌표계이다.

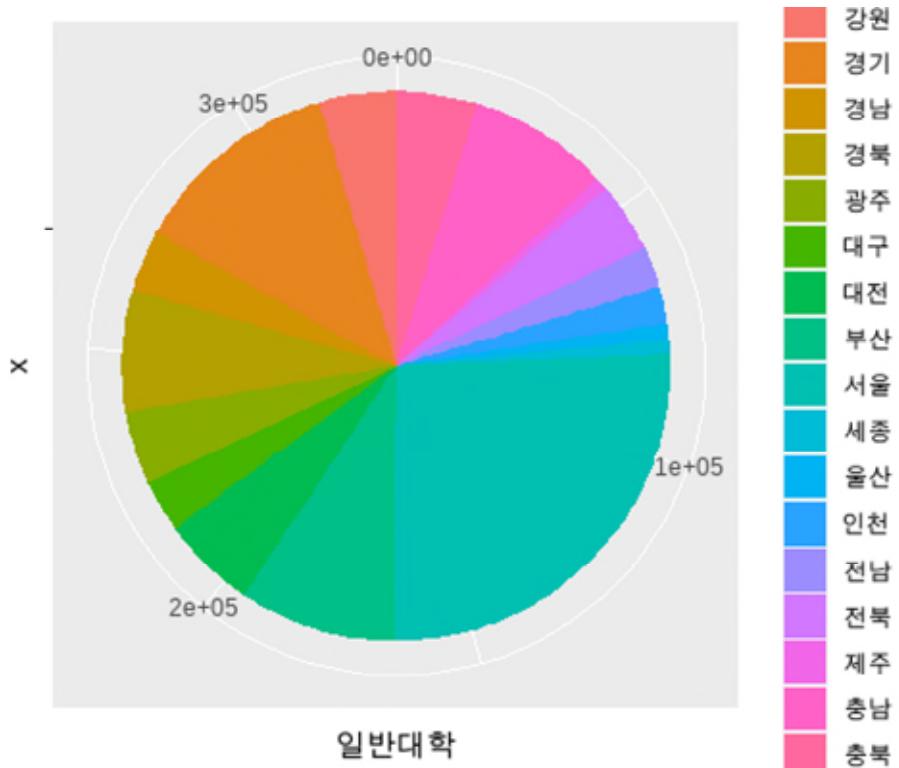
2.5.2.2. `coord_polar()`

`coord_polar()`는 반지름과 각도를 사용하여 좌표를 지정하는 극(Polar) 좌표계를 설정하는 함수이다. 사실 극 좌표계는 모든 기하요소에서 사용하기 보다는 특정한 응용이 필요한 기하요소에서 사용되는 좌표계이다. 극 좌표계에서 표현되는 기하요소는 좌표계의 변형에 따라 같이 변형되기 때문에 막대 그래프가 원형 그래프로 변형된다. `coord_polar()`는 다음과 같이 사용된다.

```
coord_polar(theta = "x", start = 0, direction = 1, ...)
- theta : 각도로 매핑될 변수 설정(x 나 y 중 하나)
- start : 시작각도로 사용할 위치점의 12 시 방향에서부터의 오프셋값
- direction : 1은 시계방향, -1은 반시계방향
```

`ggplot`에서 파이차트를 그리는 방법은 막대 그래프에 극 좌표계를 적용시키는 방법이다.

```
## df_입학자 데이터프레임에서 연도가 '2021'년도, 지역이 전체가 아닌 데이터를 필터링하고
df_입학자 |> filter(연도 == '2021', 지역 != '전체') |>
## ggplot 객체를 생성하는데 x 축을 NULL, y 축을 일반대학, fill 을 지역으로 매핑한 ggplot 객체를 생성하고
ggplot(aes(x = '', y = 일반대학, fill = 지역)) +
## 통계 변환이 없는(identity) geom_col 레이어를 생성하고
geom_col(stat = 'identity') +
## y 축을 기준으로 극 좌표계를 적용
coord_polar(theta = 'y')
```



2.6. 테마요소

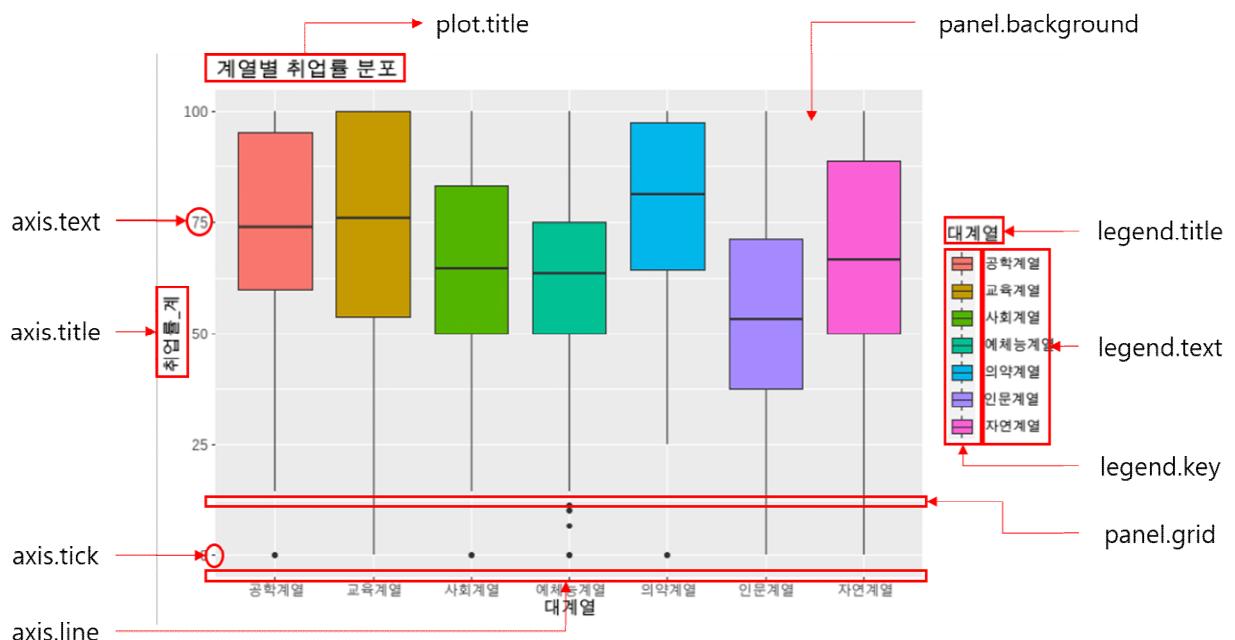
테마요소는 `ggplot` 객체의 데이터로 표현되지 않는 시각화 세부 요소들에 대한 설정을 위한 요소이다. 테마요소는 데이터와 관련 없는 요소들이기 때문에 시각화의 기본적인 속성을 변경하지는 않고 시각화를 접하는 청중이 시각화 내용을 보다 이해하기 쉽게 만들어 주는 폰트, 백그라운드, 패널 보조선, 눈금 등의 설정과 관련된 요소이다. `ggplot`에서의 테마요소 설정에는 다음의 네 가지 요소를 사용하여 설정된다.

2.6.1. `theme()`

테마요소를 설정하기 위해서 사용하는 함수이다. 테마요소들은 모두 `theme()` 안에서 설정되어야 한다. 이를 설정하기 위해 `theme()` 안에서 다음에 설명하는 엘리먼트와 엘리먼트 함수를 사용한다.

2.6.2. 엘리먼트(element)

테마 엘리먼트는 데이터 시각화 세부 요소 중에 데이터와 관련되지 않은 요소들을 지칭한다. 예를 들어 `plot.title`은 전체 시각화의 제목을 지칭하고 `axis.x.ticks`는 X 축의 눈금선을 지칭한다. 이렇게 지칭된 엘리먼트들의 특성을 설정함으로써 테마요소들을 설정하게 된다. 많이 사용되는 테마 엘리먼트의 이름은 다음과 같다.



`ggplot`에서 설정이 가능한 테마 엘리먼트들은 너무 많아 다 설명할 수 없다. 주로 사용되는 엘리먼트는 다음과 같다.

엘리먼트 이름	세부 엘리먼트	설명
plot.title		전체 시각화 제목
axis.title, axis.title.x, axis.title.y	top, bottom, left, right	축 제목의 설정
axis.text, axis.text.x, axis.text.y	top, bottom, left, right	축의 눈금 텍스트 설정
axis.ticks, axis.ticks.x, axis.ticks.y	top, bottom, left, right	축의 눈금자 설정
axis.ticks.length, axis.ticks.length.x, axis.ticks.length.y	top, bottom, left, right	축의 눈금자 길이 설정
axis.line, axis.line.x, axis.line.y		축의 눈금선 설정
legend.background		범례의 배경 설정
legend.key	size, height, width	범례 키 설정
legend.text	align	범례 키 라벨 설정
panel.background		패널의 배경 설정
panel.grid	major, minor	패널의 눈금선 설정

2.6.3. 엘리먼트 함수

위의 엘리먼트 이름을 사용하여 엘리먼트가 지정되면 `element_*`()를 사용하여 세부적인 특성을 설정할 수 있다. 엘리먼트 함수는 엘리먼트의 종류에 따라 `element_line()`, `element_text()`, `element_rect()`, `element_blank()` 등이 있다.

2.6.3.1. `element_line()`

엘리먼트의 표현이 선의 형태로 표현되는 엘리먼트의 특성을 설정할 때 사용하는 함수이다. `element_line()`의 사용법과 주요 매개변수는 다음과 같다.

```
element_line(colour = NULL, size = NULL, linetype = NULL, lineend = NULL, color = NULL, arrow = NULL, ...)

- colour : 선의 색 설정
- size : 선의 두께 설정
```

- `linetype` : 선의 종류 설정
- `lineend` : 선의 끝 스타일 설정(`round`, `butt`, `square`)
- `arrow` : 선 끝의 화살표 설정

`element_line()`을 사용하여 테마요소를 설정하는 방법은 다음과 같다.

```
theme_element <- df_입학자 |> ggplot(aes(x = 연도, y = 일반대학))

theme_element +
  theme(axis.line.x = element_line(linetype = 2, size = 1, color = 'blue'),
        axis.line.y = element_line(linetype = 3, size = 1.5, color = 'skyblue'),
        panel.grid.major.x = element_line(linetype = 1, color = 'red', arrow = grid::arrow(length = unit(0.3, "cm"), ends = "both")),
        panel.grid.major.y = element_line())
```

