그래프 그리기 (3) - 막대그래프

#01. 그래프 생성 준비하기

막대그래프는 데이터의 크기를 막대의 길이로 표현한 시각화 자료로서 "성별-소득차이" 처럼 집단간의 차이를 표현할 때 주로 사용됩니다.

R에서 생성할 수 있는 막대 그래프의 종류에는 빈도 막대 그래프와 평균 막대 그래프로 구분할 수 있습니다.

1) 필요한 패키지 로드

```
# 한국 서버를 통해 라이브러리 로드
REPO_URL <- "https://cran.seoul.go.kr/"

# 그래프 패키지
if (!require(ggplot2)) install.packages("ggplot2", repos=REPO_URL)
library(ggplot2)

# 폰트 설정 패키지
if (!require(extrafont)) install.packages("extrafont", repos=REPO_URL)
library(extrafont)

# 데이터 전처리 패키지
if (!require("dplyr"))     install.packages("dplyr", repos=REPO_URL)
library(dplyr)
```

▶ 출력결과

```
Loading required package: ggplot2
Loading required package: extrafont
Registering fonts with R
Loading required package: dplyr

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

2) 폰트 로드

나눔고딕 폰트를 검색하여 설치하기

```
font_import(pattern = 'NanumGothic')
```

▶ 출력결과

```
Importing fonts may take a few minutes, depending on the number of fonts and the speed of the system.
Continue? [y/n]  y


Scanning ttf files in /Library/Fonts/, /System/Library/Fonts, ~/Library/Fonts/ ...
Extracting .afm files from .ttf files...
/Users/leekh/Library/Fonts/NanumGothic.ttf : NanumGothic already registered in fonts database. Skipping.
/Users/leekh/Library/Fonts/NanumGothicBold.ttf : NanumGothicBold already registered in fonts database. Skip
/Users/leekh/Library/Fonts/NanumGothicExtraBold.ttf : NanumGothicExtraBold already registered in fonts data
```

```
/Users/leekh/Library/Fonts/NanumGothicLight.ttf : NanumGothicLight already registered in fonts database. Sk
Found FontName for 0 fonts.
... 생략 ...
```

설치된 폰트들 로드하기

```
# mac의 경우 `device="win"` 생략
#loadfonts(device="win")
loadfonts()
```

▶ 출력결과

```
D2Coding already registered with pdfFonts().
NanumBarunGothic already registered with pdfFonts().
NanumBarunGothic Light already registered with pdfFonts().
NanumBarunGothic UltraLight already registered with pdfFonts().
NanumBarunpen already registered with pdfFonts().
No regular (non-bold, non-italic) version of NanumBarunpen Bold. Skipping setup for this font.
Nanum Brush Script already registered with pdfFonts().
NanumGothic already registered with pdfFonts().
NanumGothicExtraBold already registered with pdfFonts().
NanumGothic Light already registered with pdfFonts().
```

3) 그래프 기본 크기 및 불필요한 경고 메시지 끄기

```
options(repr.plot.width=15, repr.plot.height=9, warn=-1)
```

#02. 빈도 막대 그래프(geom_bar)

데이터프레임에서 값들에 대한 종류를 판별하여 종류별로 얼마나 자주 등장하는지에 대한 현황을 표현하는 형태

1) 샘플 데이터 가져오기

학생들의 나이와 좋아하는 계절 조사 자료 (임의의 예시 데이터)

```
설문 <- read.csv("http://itpaper.co.kr/demo/r/season.csv", stringsAsFactors=F, fileEncoding="euc-kr")
설문
```

▶ 출력결과

A data.frame: 500 × 3

이름	계절	나이
<chr>	<chr>	<int>
학생1	봄	15
학생2	봄	17
학생3	여름	18
학생4	겨울	19
학생5	가을	15
학생6	봄	16
학생7	여름	19
학생8	여름	18

이름	계절	나이
<chr>	<chr>	<int>
학생9	여름	17
학생10	여름	17
학생11	가을	19
학생12	가을	19
학생13	여름	18
학생14	가을	19
학생15	겨울	15
학생16	겨울	16
학생17	봄	16
학생18	여름	18
학생19	여름	17
학생20	봄	18
학생21	겨울	19
학생22	가을	15
학생23	가을	17
학생24	여름	18
학생25	봄	19
학생26	여름	15
학생27	겨울	16
학생28	겨울	19
학생29	가을	18
학생30	가을	17
⋮	⋮	⋮
학생471	가을	19
학생472	여름	15
학생473	봄	15
학생474	여름	17
학생475	겨울	18
학생476	겨울	19
학생477	가을	15
학생478	가을	16
학생479	가을	19
학생480	봄	18
학생481	봄	17
학생482	여름	17
학생483	겨울	19

이름	계절	나이
<chr>	<chr>	<int>
학생484	가을	19
학생485	봄	18
학생486	여름	19
학생487	여름	15
학생488	여름	17
학생489	여름	18
학생490	가을	19
학생491	가을	15
학생492	여름	16
학생493	가을	19
학생494	겨울	18
학생495	겨울	17
학생496	봄	17
학생497	여름	19
학생498	여름	19
학생499	봄	18
학생500	겨울	19

2) 빈도 막대 그래프 생성

기본 기능으로 그래프 표현하기

`ggplot()` 함수에 빈도 막대 그래프를 의미하는 `geom_bar()` 함수를 연결한다.

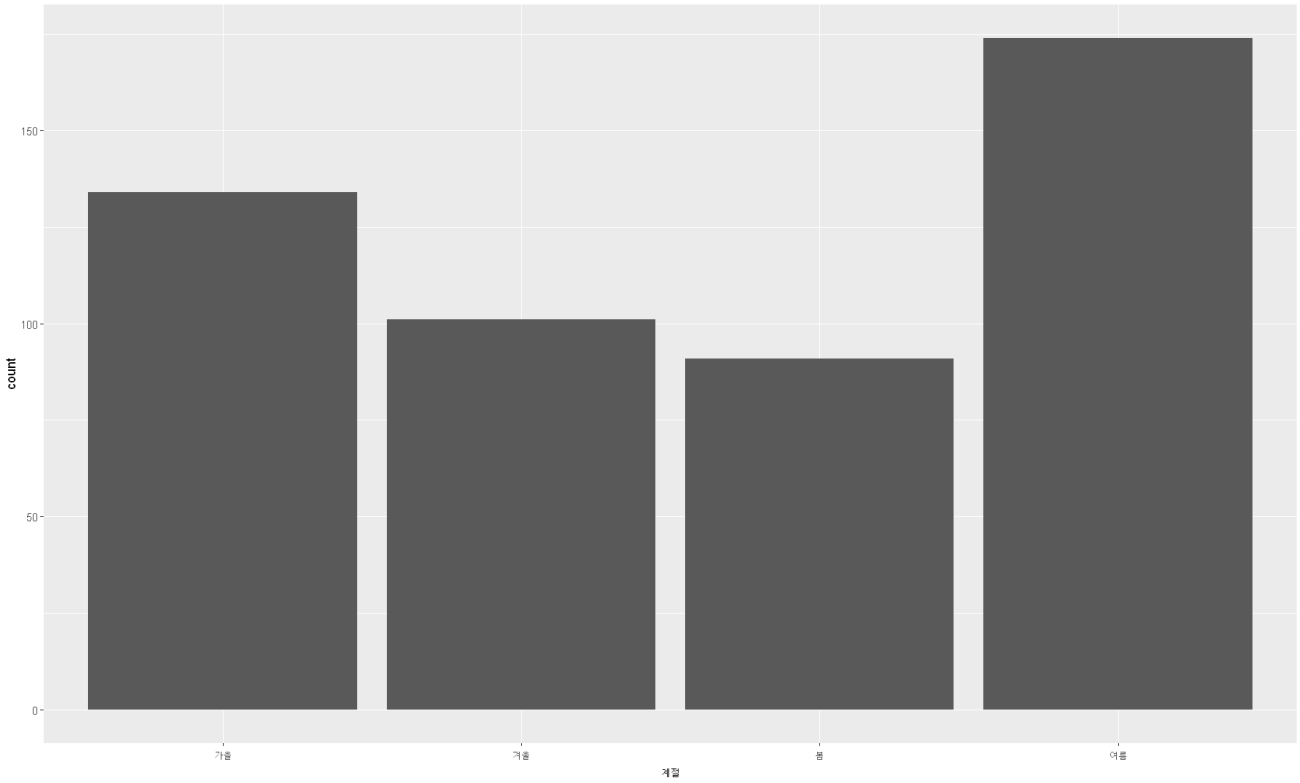
x축으로 사용할 자료만 지정한다.

y축은 x축에 지정된 자료의 빈도수를 계산하여 자동으로 표현된다.

> 윈도우의 경우는 별도의 폰트 설정을 하지 않더라도 한글이 표시되지만 맥의 경우 폰트 설정을 하지 않으면 한글이 깨져서 표시된다.

```
ggplot(data=설문) + geom_bar(aes(x=계절))
```

▶ 출력결과



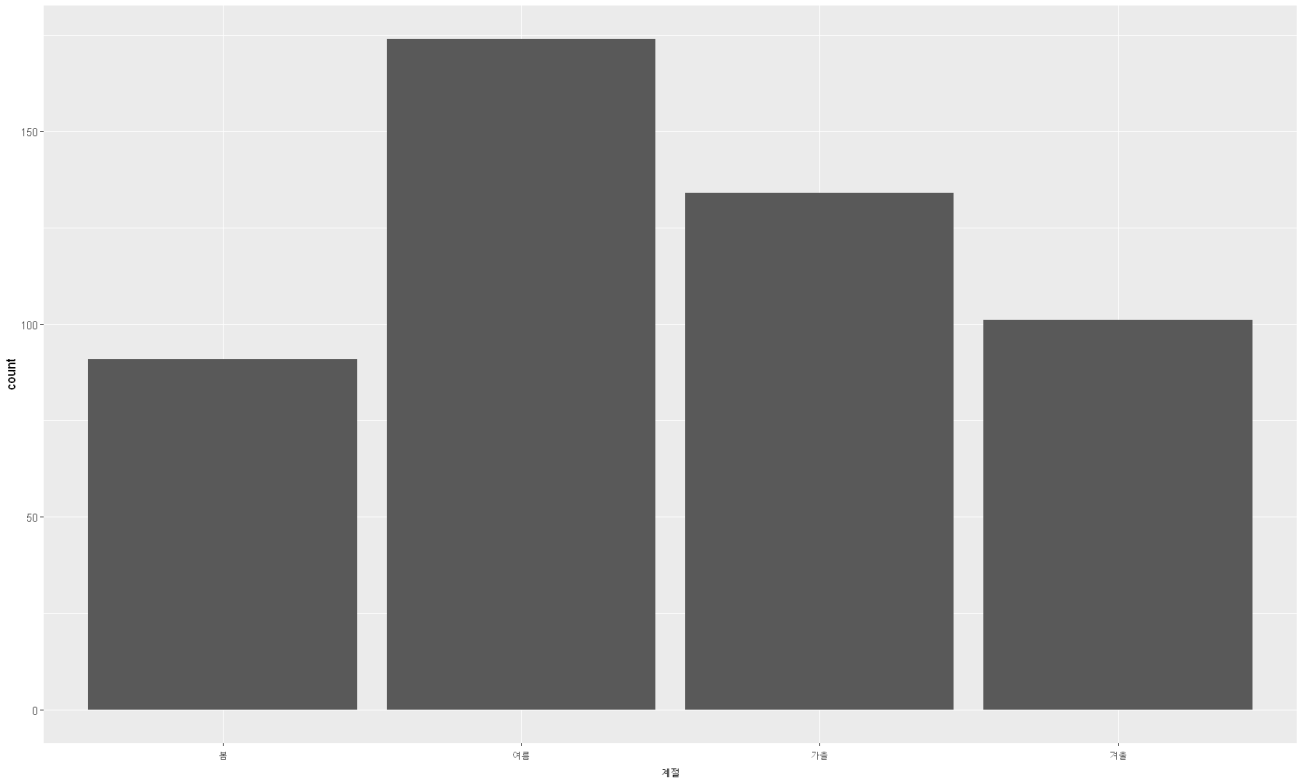
x축 순서 정하기

데이터 프레임에서 x축으로 사용되는 컬럼에 대하여 순서가 정해진 요인을 적용한 후 그래프를 생성한다.

> 이후 실습에서 활용하기 위해 그래프를 변수에 담아서 출력함

```
설문$계절 <- factor(설문$계절, levels=c("봄", "여름", "가을", "겨울"))
ggplot(data=설문) + geom_bar(aes(x=계절))
```

▶ 출력결과



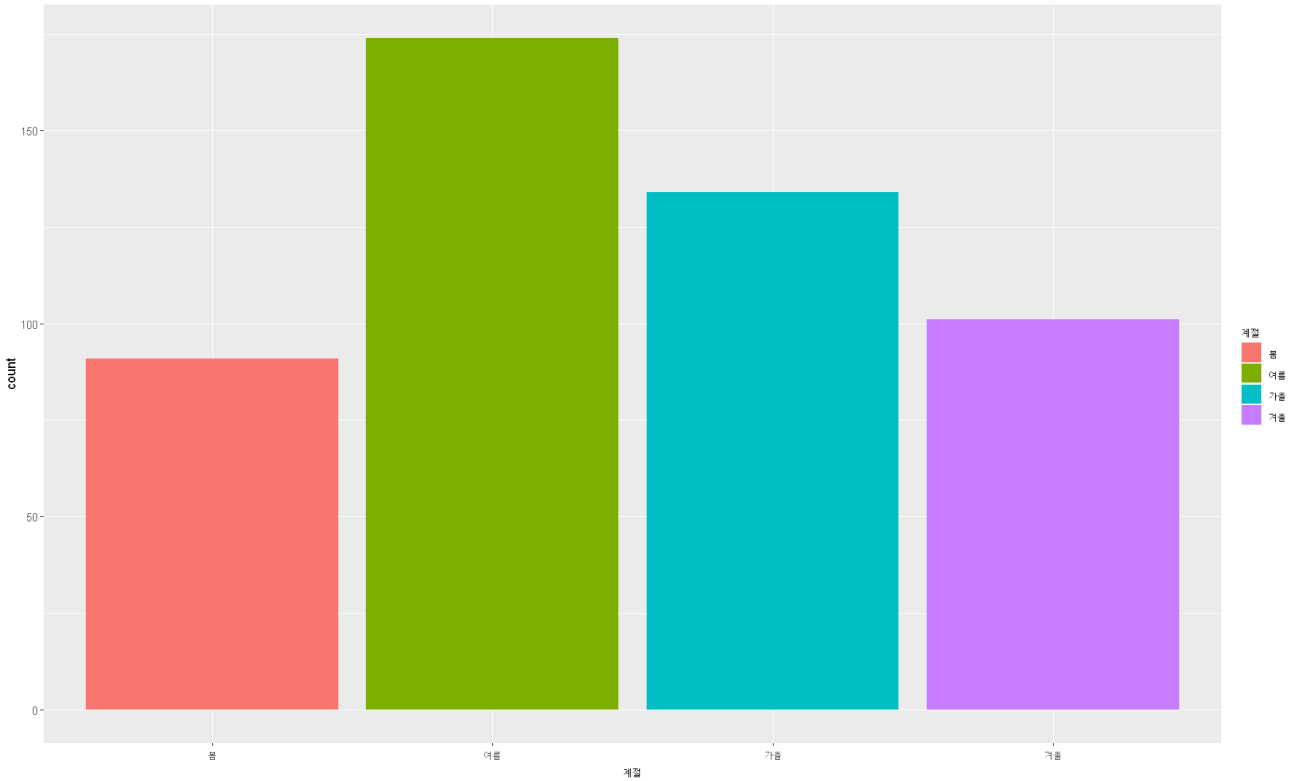
3) 색상별로 구분하기

`geom_bar()` 함수에 설정하는 `ase()` 함수에서 fill 파라미터로 같은 색상으로 구분하고자 하는 컬럼을 지정한다.

하나의 컬럼만으로 표현하는 그래프인 경우 x축에 사용한 컬럼과 동일하게 설정하면 된다.

```
ggplot(data=설문) + geom_bar(aes(x=계절, fill=계절))
```

▶ 출력결과



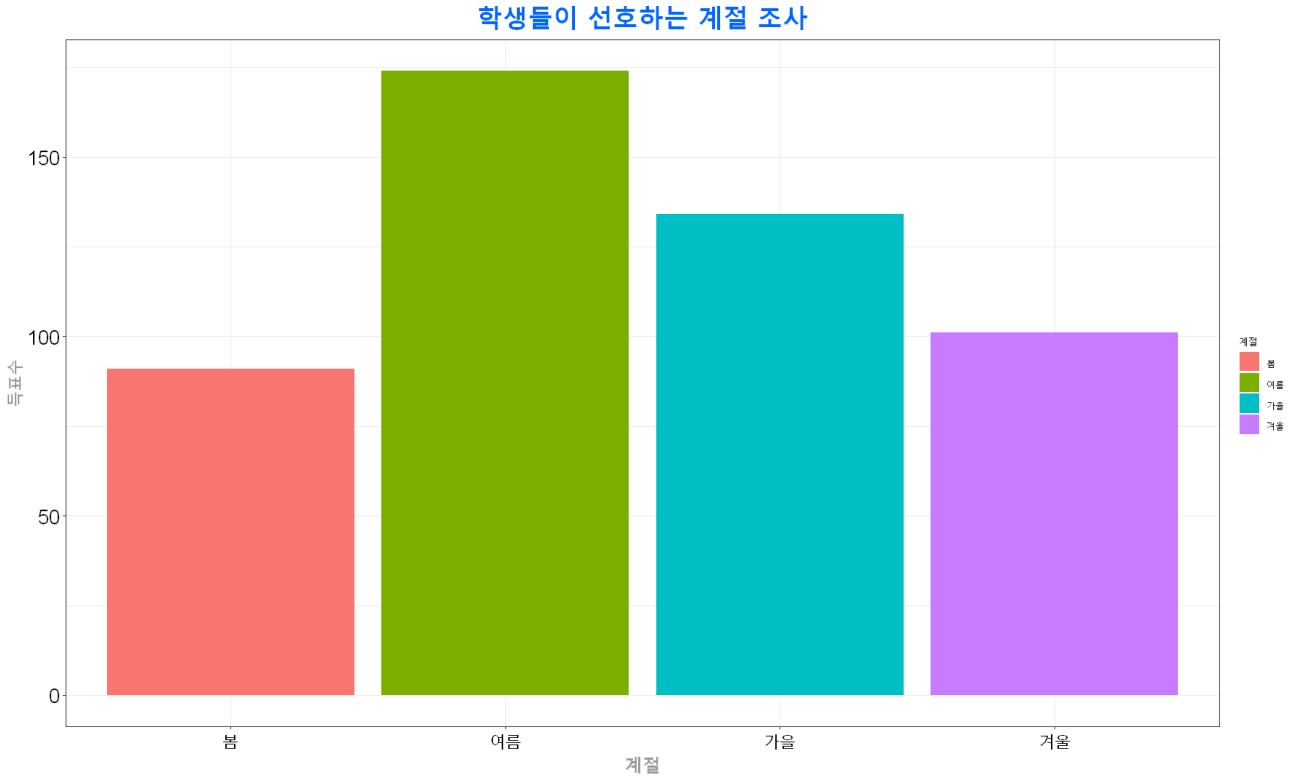
4) 옵션 적용하기

표시되는 범주에 대한 폰트 설정이 지정되지 않았기 때문에 Mac에서는 한글이 깨져서 표시된다.

```
graph <- ggplot(data=설문) + geom_bar(aes(x=계절, fill=계절)) +
    # 배경을 흰색으로 설정
    theme_bw() +
    # 그래프 타이틀 설정
    ggtitle("학생들이 선호하는 계절 조사") +
    # x축 제목 설정
    xlab("계절") +
    # y축 제목 설정
    ylab("득표수") +
    # y축 간격 및 데이터에 대한 세자리 콤마 적용
    scale_y_continuous(breaks=seq(0, 200, 50), labels=scales::comma)  +
    # 각 텍스트의 색상, 크기, 각도, 글꼴 설정
    theme(plot.title=element_text(family="NanumGothic", color="#0066ff",
                                  size=25, face="bold", hjust=0.5),
          axis.title.x=element_text(family="NanumGothic", color="#999999",
                                    size=18, face="bold"),
          axis.title.y=element_text(family="NanumGothic", color="#999999",
                                    size=18, face="bold"),
          axis.text.x=element_text(family="NanumGothic", color="#000000",
                                   size=16, angle=0),
          axis.text.y=element_text(family="NanumGothic", color="#000000",
                                   size=16, angle=0))

graph
```

▶ 출력결과



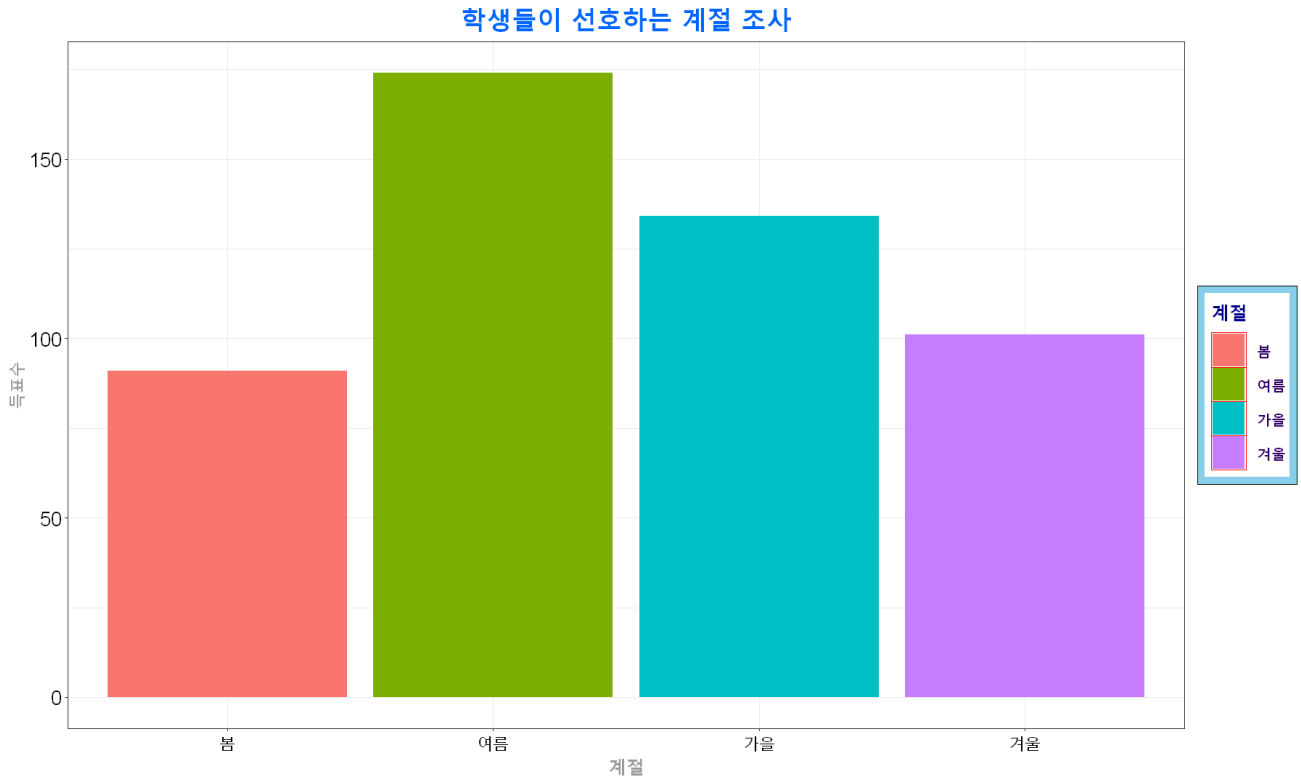
5) 범주 꾸미기

범례도 축이나 타이틀과 같이 `theme()` 함수를 통해 꾸미는 것이 가능하다.

범주의 제목, 텍스트 글자, 배경상자 꾸미기

```
graph +
    # 범주의 제목
    theme(legend.title=element_text(family="NanumGothic", face="bold", size=18, color="darkblue")) +
    # 범주의 각 항목별 텍스트
    theme(legend.text=element_text(family="NanumGothic", face="bold", size=15, color="#330066")) +
    # 범주를 구분하는 각 색상 박스에 대한 테두리와 배경
    theme(legend.key=element_rect(color="red", fill="white")) +
    # 범주의 배경상자 색상 설정
    theme(legend.box.background = element_rect(fill="skyblue")) +
    # 범주의 배경상자 여백
    theme(legend.box.margin = margin(6, 6, 6, 6)) +
    # 범주를 구분하는 각 색상 박스에 대한 크기
    theme(legend.key.size = unit(1,"cm"))
```

▶ 출력결과

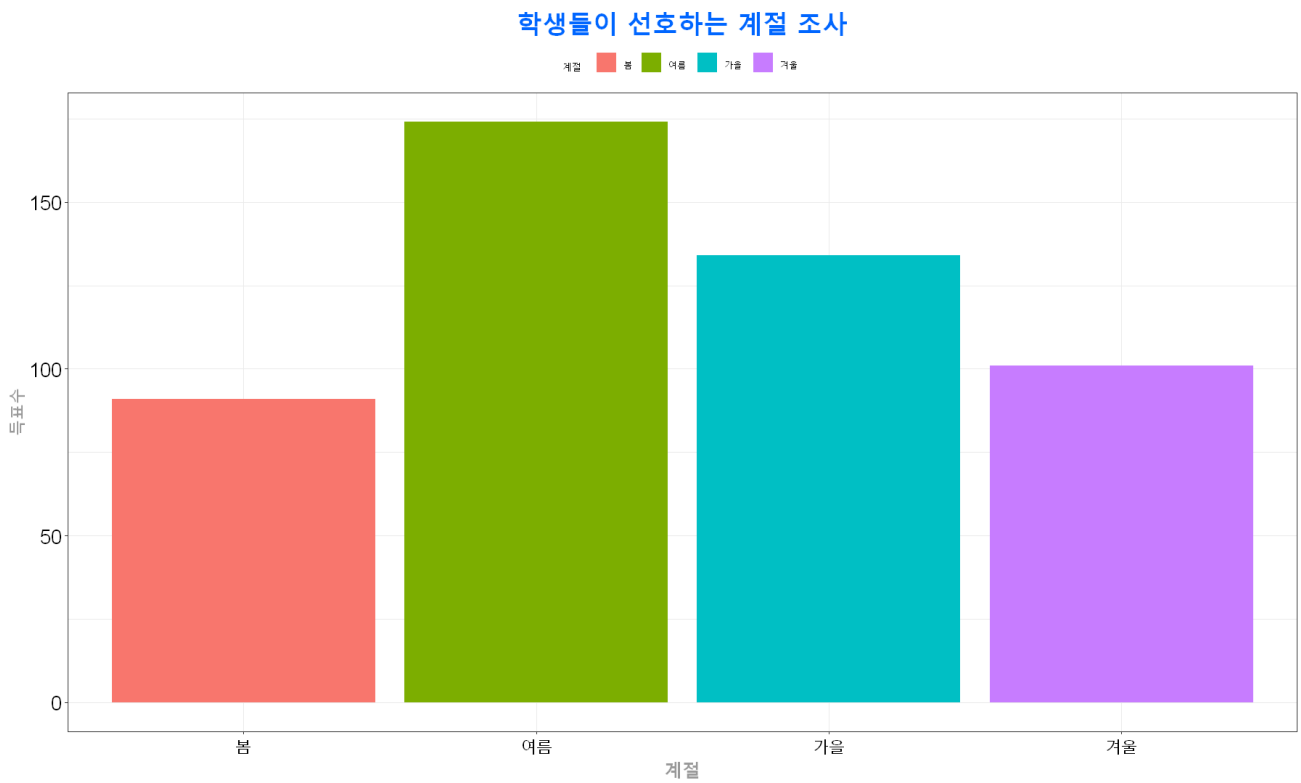


범주의 위치 지정하기

`top`, `right`, `bottom`, `left` 중 하나를 적용하여 위치를 설정할 수 있다.

```
graph + theme(legend.position = "top")
```

▶ 출력결과



범주 위치를 그래프 안으로 넣기

범주 위치는 그래프 외부에서 바꾸는 것 외에 그래프 내부로 가져오는 것도 가능하다.

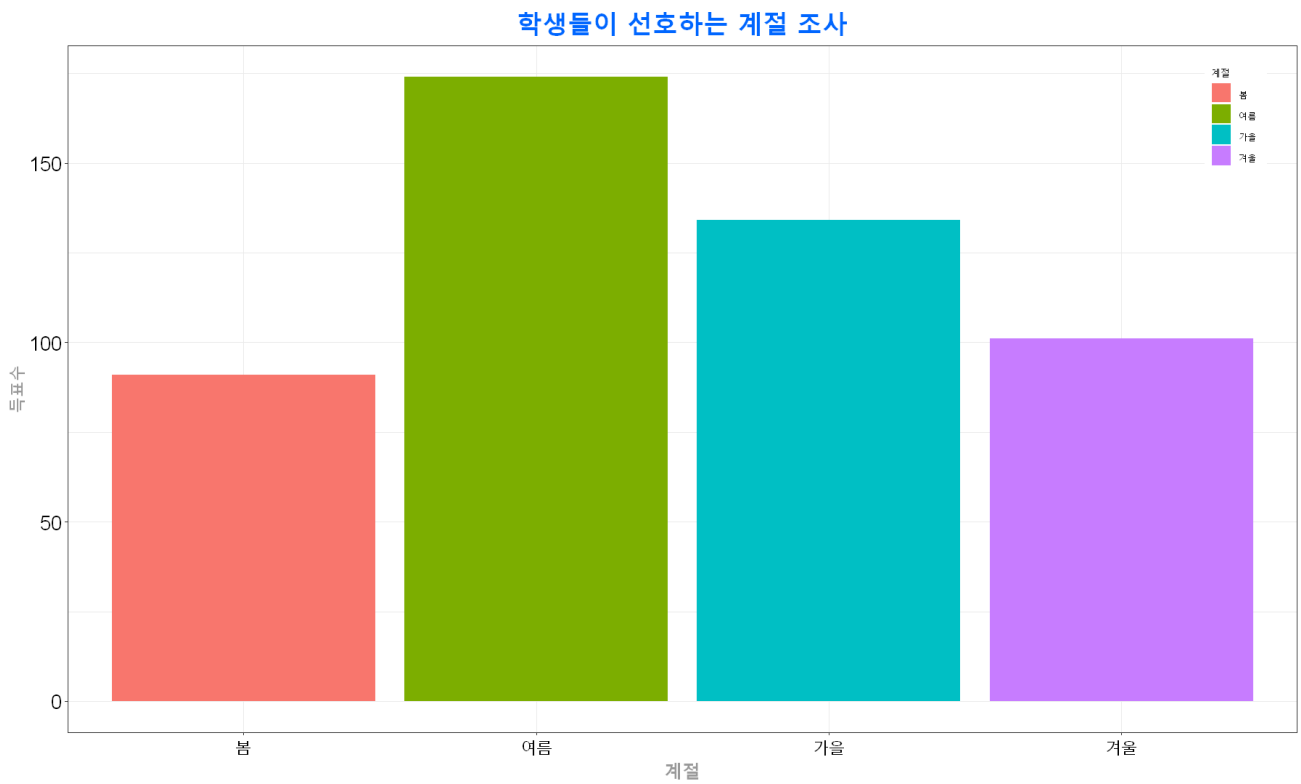
동일하게 `theme()` 함수에서 그래프 내부의 X, Y 좌표를 설정함으로써 그래프 내부로 옮길 수 있다.

수치값은 0이 가장 왼쪽/아래쪽, 1이 가장 위쪽/오른쪽이다.

> 정확한 수치값을 찾기보다는 여러번 수정해 가면서 적정값을 찾도록 하자.

```
graph + theme(legend.position = c(0.95, 0.9))
```

▶ 출력결과



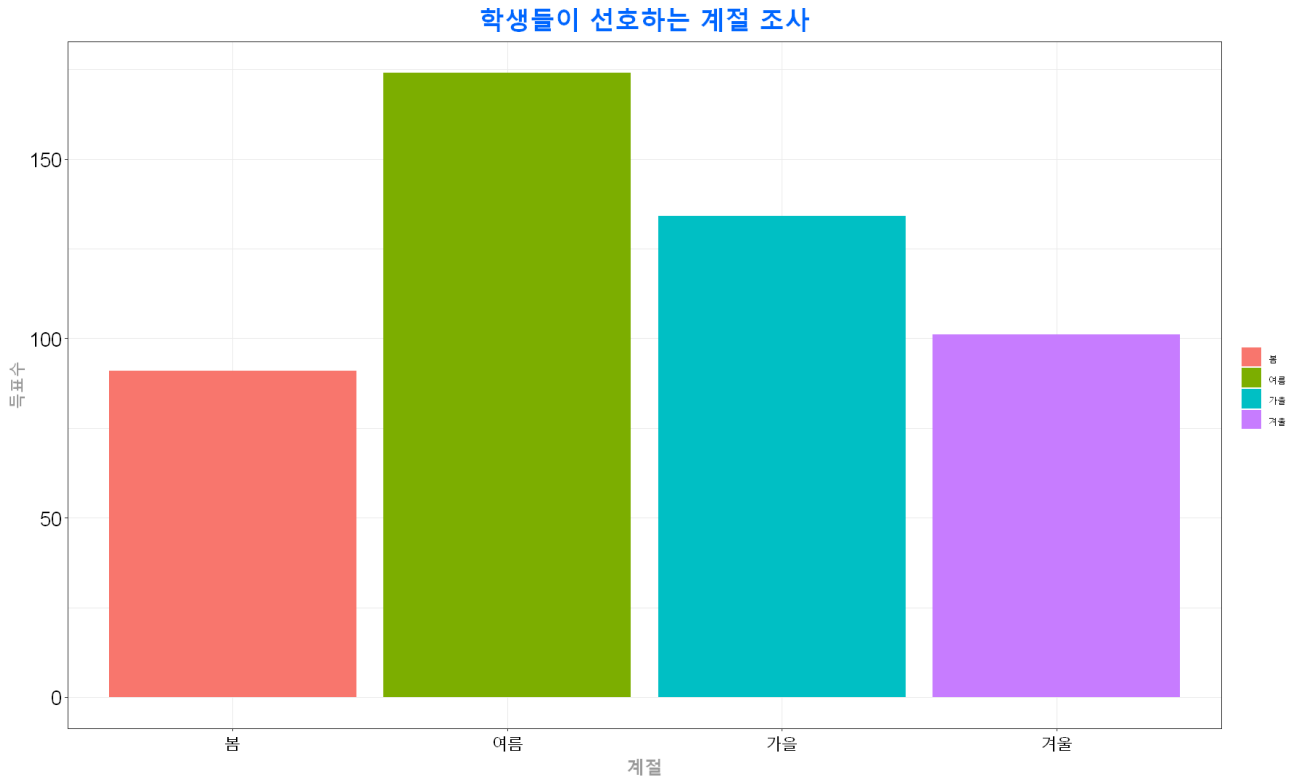
범주 타이틀만 지우기

범례 전체를 지우는 것 외에 범례의 타이틀만 제거하는 것도 가능하다.

범례의 타이틀만 제거할 때에는 범례의 제거와는 달리 theme() 함수에서 element_blank()를 입력해야 한다.

```
graph + theme(legend.title = element_blank())
```

▶ 출력결과

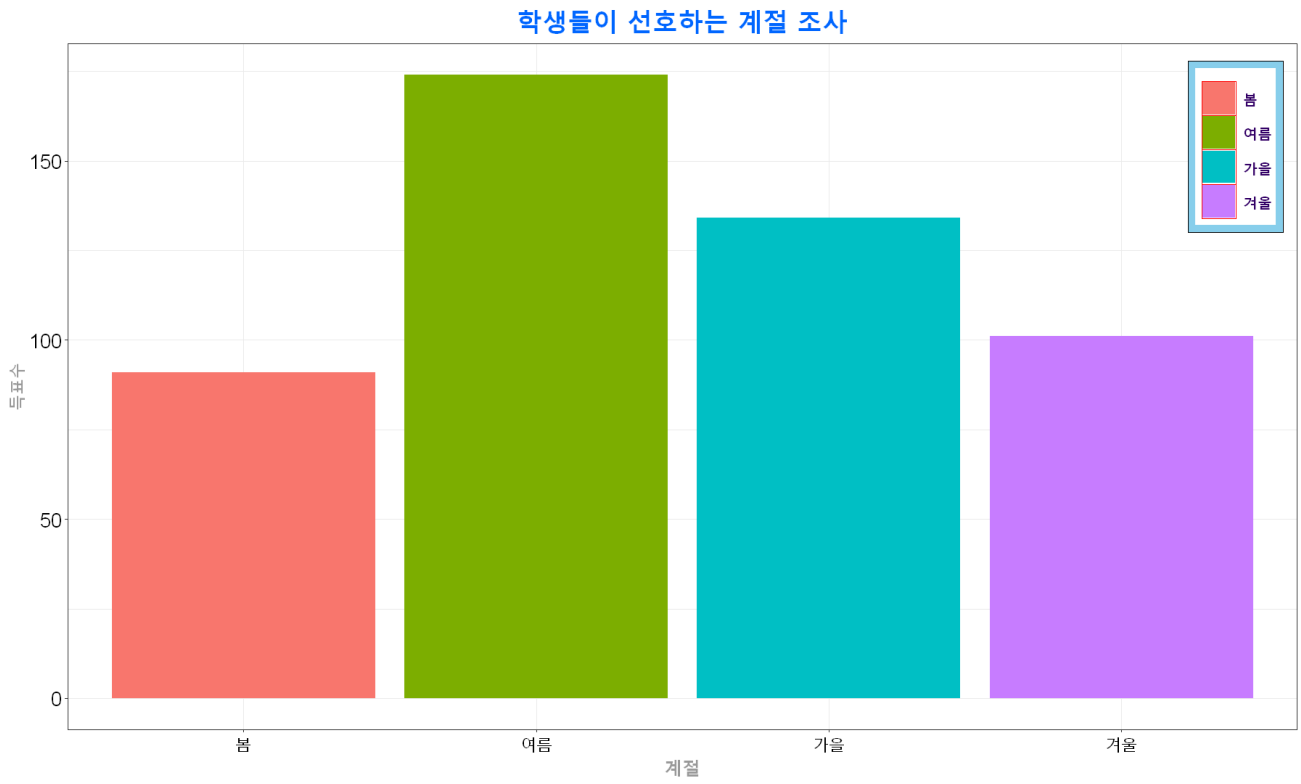


theme 함수 일괄 지정하기

theme 함수를 각 설정 항목마다 나누어서 설정할 수도 있지만 콤마로 구분하여 각 항목을 한번에 지정할 수 도 있다.

```
graph + theme(legend.title = element_blank(),
              legend.text = element_text(face="bold", size=15, color="#330066"),
              legend.key = element_rect(color="red", fill="white"),
              legend.key.size = unit(1,"cm"),
              legend.box.background = element_rect(fill="skyblue"),
              legend.box.margin = margin(6, 6, 6, 6),
              legend.position = c(0.95, 0.85))
```

▶ 출력결과



6) 다중 빈도 막대 그래프

컬럼의 타입 확인

빈도 막대 그래프는 시각화 하고자 하는 컬럼의 스칼라 타입이 종류를 구분할 수 있는 형태인 문자열(`chr`)이거나 요인(`fct`)인 경우만 표현 가능하다.

현재 사용중인 데이터 프레임은 계절 컬럼의 경우 초기에 `chr` 타입에서 값의 순서를 설정하기 위해 요인(`factor`) 타입으로 변경되었지만 나이 컬럼은 정수(`int`) 타입이므로 종류를 구분하는 형식이 아닌 연속성 데이터로 판단되어 빈도 막대 그래프를 작성할 수 없다.

```
설문
```

▶ 출력결과

A data.frame: 500 × 3

이름	계절	나이
<chr>	<fct>	<int>
학생1	봄	15
학생2	봄	17
학생3	여름	18
학생4	겨울	19
학생5	가을	15
학생6	봄	16
학생7	여름	19
학생8	여름	18
학생9	여름	17
학생10	여름	17
학생11	가을	19
학생12	가을	19
학생13	여름	18
학생14	가을	19
학생15	겨울	15
학생16	겨울	16
학생17	봄	16
학생18	여름	18
학생19	여름	17
학생20	봄	18
학생21	겨울	19
학생22	가을	15
학생23	가을	17
학생24	여름	18
학생25	봄	19
학생26	여름	15

이름	계절	나이
<chr>	<fct>	<int>
학생27	겨울	16
학생28	겨울	19
학생29	가을	18
학생30	가을	17
⋮	⋮	⋮
학생471	가을	19
학생472	여름	15
학생473	봄	15
학생474	여름	17
학생475	겨울	18
학생476	겨울	19
학생477	가을	15
학생478	가을	16
학생479	가을	19
학생480	봄	18
학생481	봄	17
학생482	여름	17
학생483	겨울	19
학생484	가을	19
학생485	봄	18
학생486	여름	19
학생487	여름	15
학생488	여름	17
학생489	여름	18
학생490	가을	19
학생491	가을	15
학생492	여름	16
학생493	가을	19
학생494	겨울	18
학생495	겨울	17
학생496	봄	17
학생497	여름	19
학생498	여름	19
학생499	봄	18
학생500	겨울	19

나이 컬럼을 요인으로 변경하기

```
# 데이터프레임의 복사본 생성
설문copy <- 설문

# 나이의 최소값부터 최대값까지를 값의 종류로 삼는 요인으로 변경하기
설문copy$나이 <- factor(설문copy$나이, level=seq(min(설문copy$나이), max(설문copy$나이)))

설문copy
```

▶ 출력결과

A data.frame: 500 × 3

이름	계절	나이
<chr>	<fct>	<fct>
학생1	봄	15
학생2	봄	17
학생3	여름	18
학생4	겨울	19
학생5	가을	15
학생6	봄	16
학생7	여름	19
학생8	여름	18
학생9	여름	17
학생10	여름	17
학생11	가을	19
학생12	가을	19
학생13	여름	18
학생14	가을	19
학생15	겨울	15
학생16	겨울	16
학생17	봄	16
학생18	여름	18
학생19	여름	17
학생20	봄	18
학생21	겨울	19
학생22	가을	15
학생23	가을	17
학생24	여름	18
학생25	봄	19
학생26	여름	15
학생27	겨울	16
학생28	겨울	19
학생29	가을	18

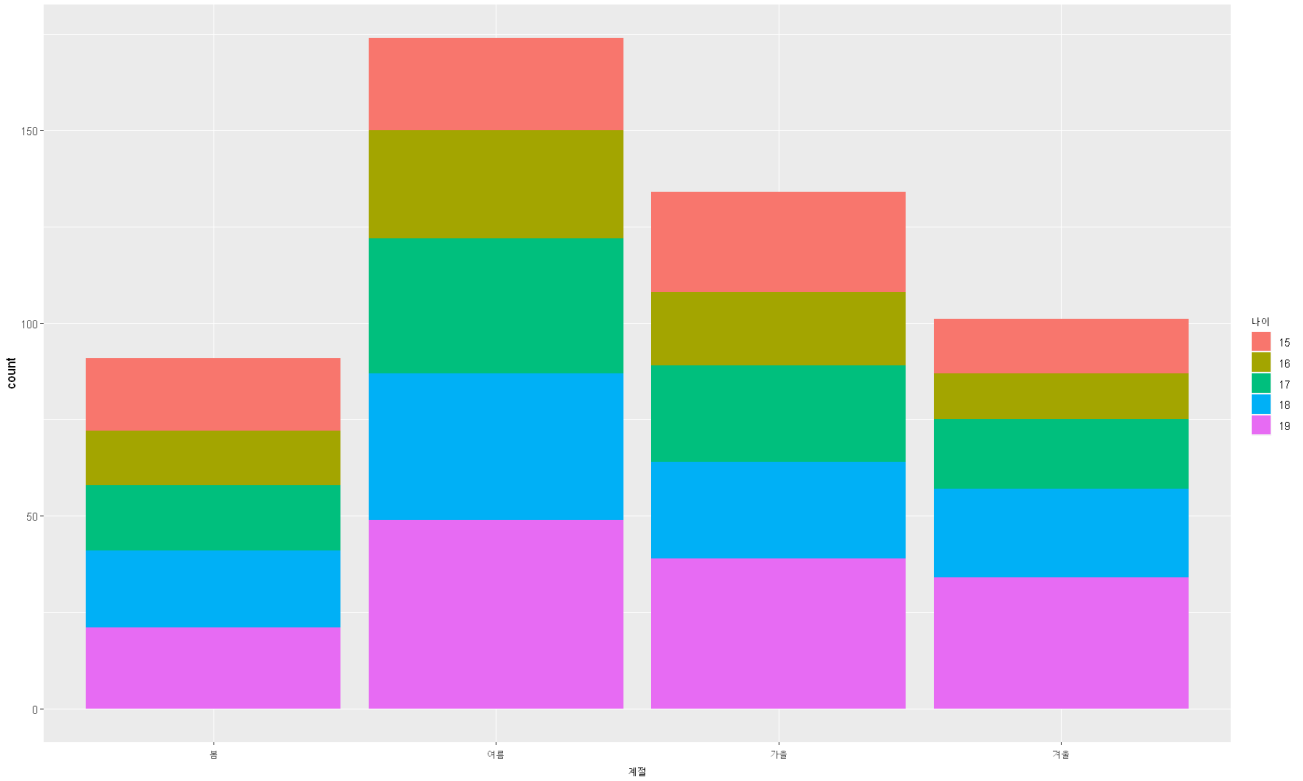
이름	계절	나이
<chr>	<fct>	<fct>
학생30	가을	17
⋮	⋮	⋮
학생471	가을	19
학생472	여름	15
학생473	봄	15
학생474	여름	17
학생475	겨울	18
학생476	겨울	19
학생477	가을	15
학생478	가을	16
학생479	가을	19
학생480	봄	18
학생481	봄	17
학생482	여름	17
학생483	겨울	19
학생484	가을	19
학생485	봄	18
학생486	여름	19
학생487	여름	15
학생488	여름	17
학생489	여름	18
학생490	가을	19
학생491	가을	15
학생492	여름	16
학생493	가을	19
학생494	겨울	18
학생495	겨울	17
학생496	봄	17
학생497	여름	19
학생498	여름	19
학생499	봄	18
학생500	겨울	19

두 개의 컬럼에 대한 빈도 막대 그래프 생성

y축은 각 계절별 합계를 의미하고 각각의 막대에서 색상은 나이별 빈도를 의미한다.

```
ggplot(data=설문copy) + geom_bar(aes(x=계절, fill=나이))
```

▶ 출력결과



빈도 막대 그래프의 형태 변경하기

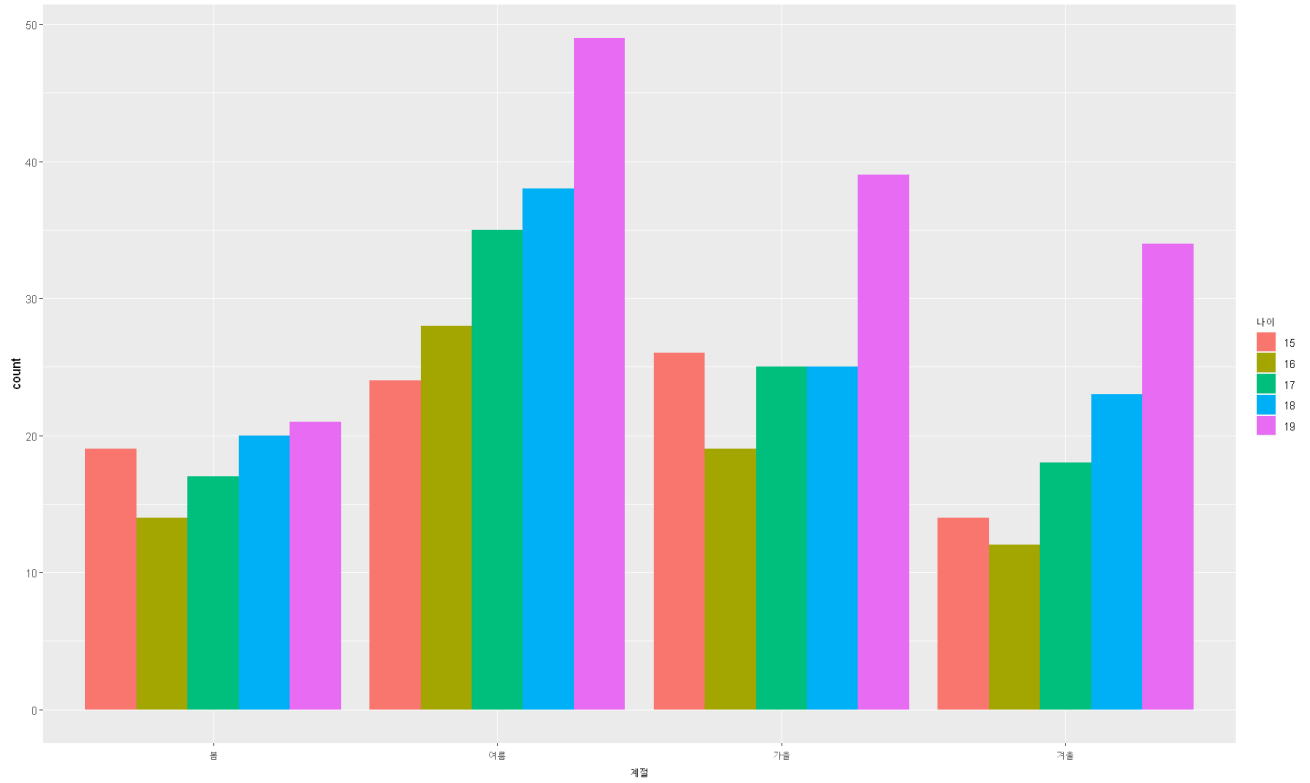
`geom_bar()` 함수에 position 파라미터를 사용하여 막대의 배치를 설정할 수 있다.

position="dodge"

나이를 색상으로 구분하여 각 나이별로 수치를 정확히 표시한다.

```
ggplot(data=설문copy) + geom_bar(aes(x=계절, fill=나이), position="dodge")
```

▶ 출력결과

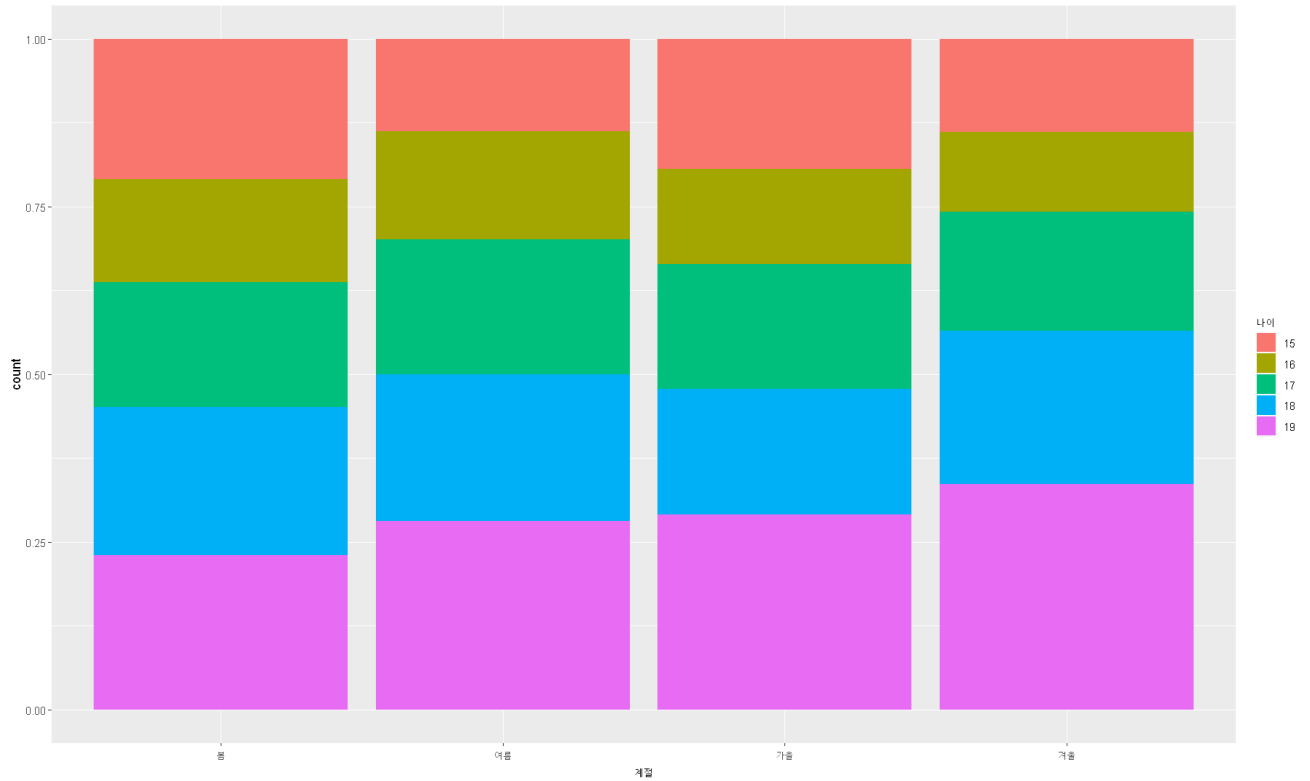


dodge="fill"

각각의 나이별로 차지하는 비중을 시각화 한다.

```
ggplot(data=설문copy) + geom_bar(aes(x=계절, fill=나이), position="fill")
```

▶ 출력결과



#03. 평균 막대 그래프

각 값의 종류별로 평균값 (혹은 다른 수치값)을 미리 산정해 놓은 상태의 DataFrame을 시각화 한다.

빈도 막대 그래프(`geom_bar()`) 는 DataFrame에서 집계가 완료되지 않은 원자료를 R이 집계하여 시각화하지만 **평균 막대 그래프 (`geom_col()`)** 는 분석가가 직접 집계를 수행한 결과를 토대로 시각화를 표현하기 때문에 x축과 y축을 모두 지정해야 한다.

1) 데이터 전처리 (데이터의 빈도를 직접 집계하기)

계절 컬럼에 대한 빈도수 계산

```
계절투표 <- table(설문$계절)
계절투표
```

▶ 출력결과

```
  봄 여름 가을 겨울
  91  174  134  101
```

계산 결과를 데이터 프레임으로 변환

```
계절투표df <- data.frame(계절투표)
계절투표df
```

▶ 출력결과

A data.frame: 4 × 2

Var1	Freq
<fct>	<int>
봄	91
여름	174
가을	134
겨울	101

데이터 프레임의 컬럼이름 재지정

```
계절투표df_이름변경 <- rename(계절투표df, '계절'=Var1, '득표수'=Freq)
계절투표df_이름변경
```

▶ 출력결과

A data.frame: 4 × 2

계절	득표수
<fct>	<int>
봄	91
여름	174
가을	134
겨울	101

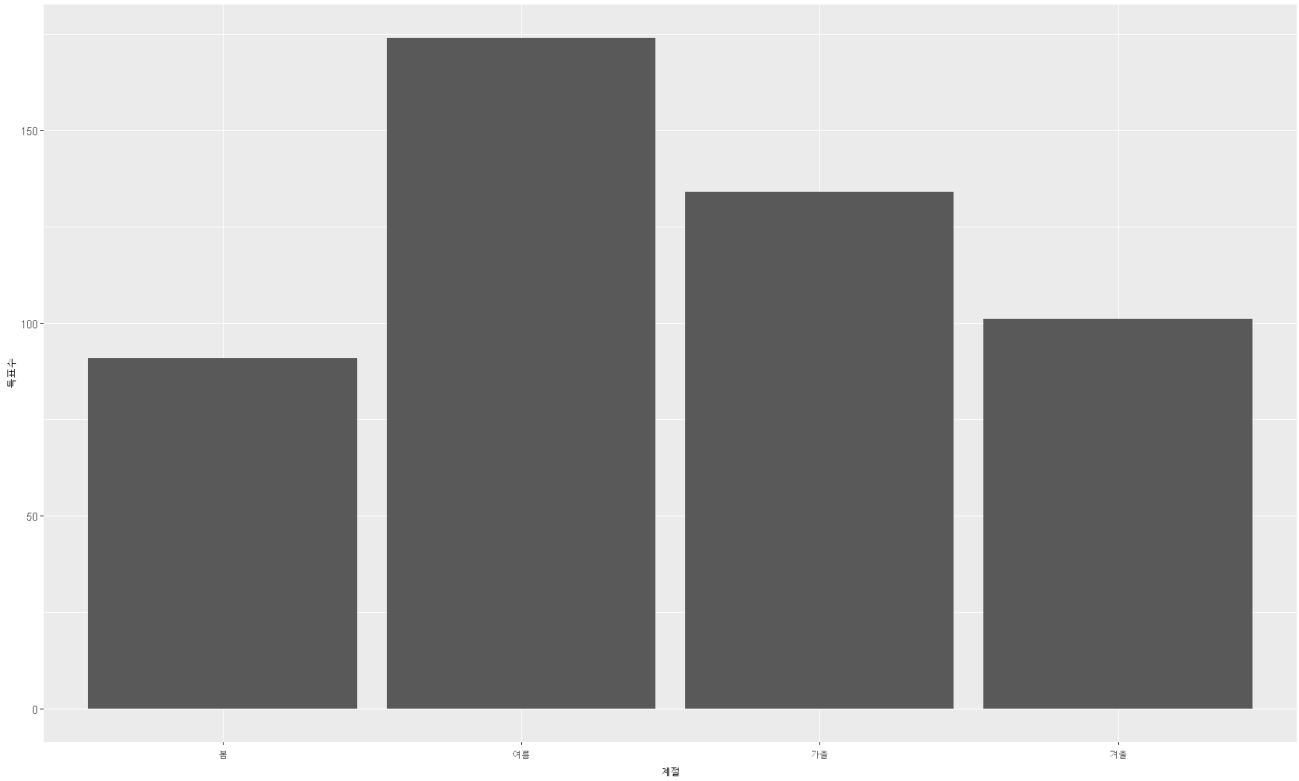
2) 데이터 시각화

기본 그래프 생성하기

그래프를 표현하기 위한 데이터를 `ggplot()` 함수에 설정하고 그래프종류 지정하기 위한 `geom_col()` 함수를 연결한다.

```
ggplot(data=계절투표df_이름변경) + geom_col(aes(x=계절, y=득표수))
```

▶ 출력결과



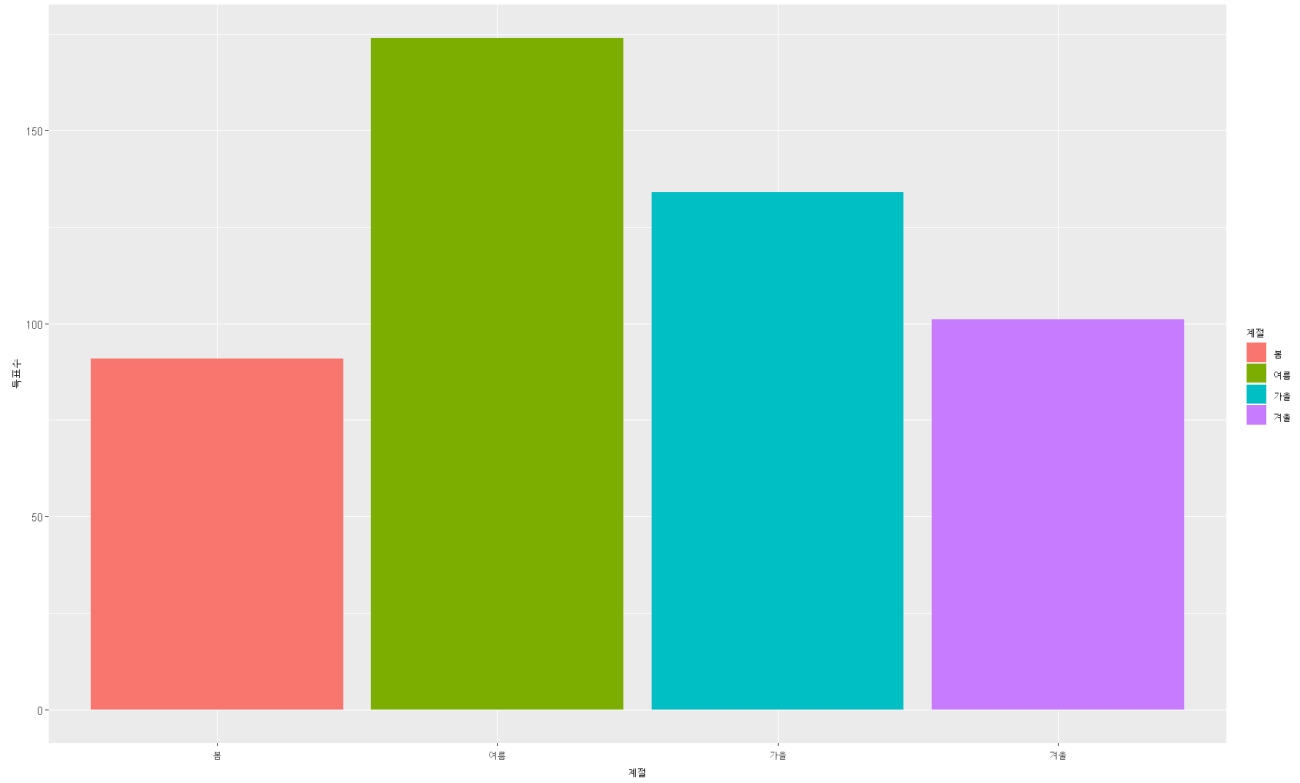
그래프에 색상 적용하기

`geom_col()` 함수에 설정하는 `aes()` 함수에 fill 파라미터를 적용한다.

그래프가 단일 막대인 경우 x축에 설정한 컬럼과 동일하게 적용하면 색상으로 각 항목을 구분할 수 있다.

```
ggplot(data=계절투표df_이름변경) + geom_col(aes(x=계절, y=득표수, fill=계절))
```

▶ 출력결과



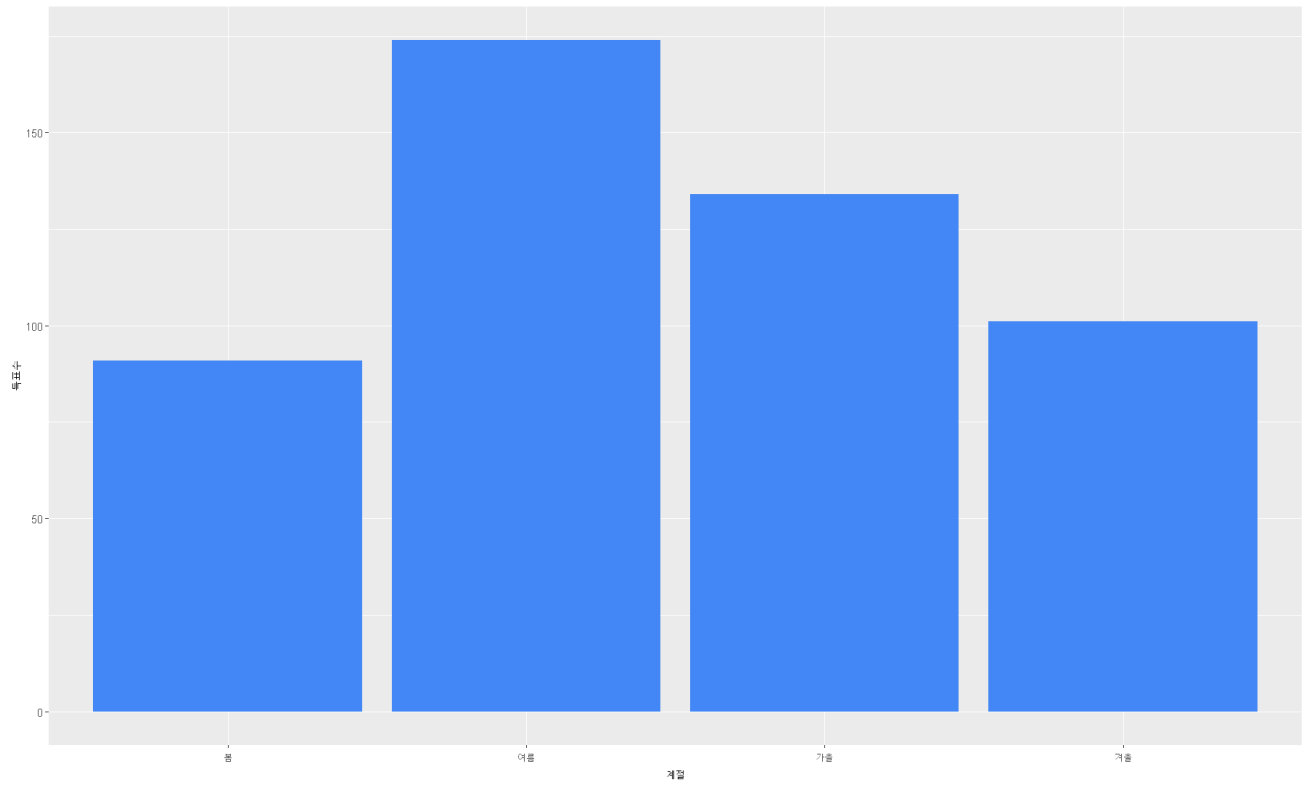
단일 색상 적용하기

`fill` 파라미터에 단일 색상을 적용한다.

단, fill 파라미터가 `ase()` 함수에 전달되는 것이 아니라 `geom_col()` 함수에 직접 전달되어야 한다.

```
ggplot(data=계절투표df_이름변경) + geom_col(aes(x=계절, y=득표수), fill='#4287f5')
```

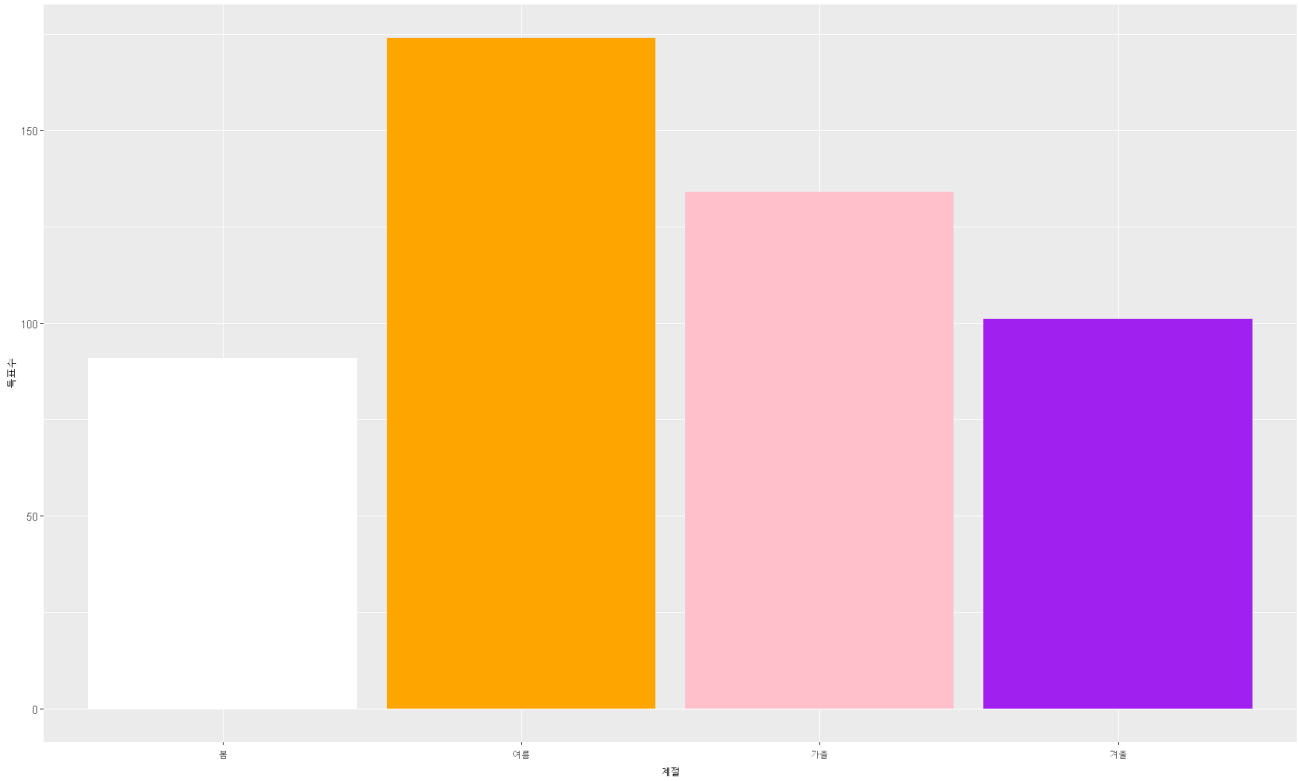
▶ 출력결과



각 막대 별로 개별 색상 지정하기

```
ggplot(data=계절투표df_이름변경) + geom_col(aes(x=계절, y=득표수), fill=c('white', 'orange', 'pink', 'purple'))
```

▶ 출력결과



3) 다중 평균 막대 그래프

데이터 전처리 (나이별로 좋아하는 계절 집계하기)

`n()` 함수는 빈도수를 계산한다. (=갯수를 카운트함)

```
설문집계 <- 설문 %>%
                    group_by(계절, 나이) %>%
                    summarise(득표수=n())
설문집계
```

▶ 출력결과

A grouped_df: 20 × 3

계절	나이	득표수
<fct>	<int>	<int>
봄	15	19
봄	16	14
봄	17	17
봄	18	20
봄	19	21
여름	15	24
여름	16	28
여름	17	35
여름	18	38

계절	나이	득표수
<fct>	<int>	<int>
여름	19	49
가을	15	26
가을	16	19
가을	17	25
가을	18	25
가을	19	39
겨울	15	14
겨울	16	12
겨울	17	18
겨울	18	23
겨울	19	34

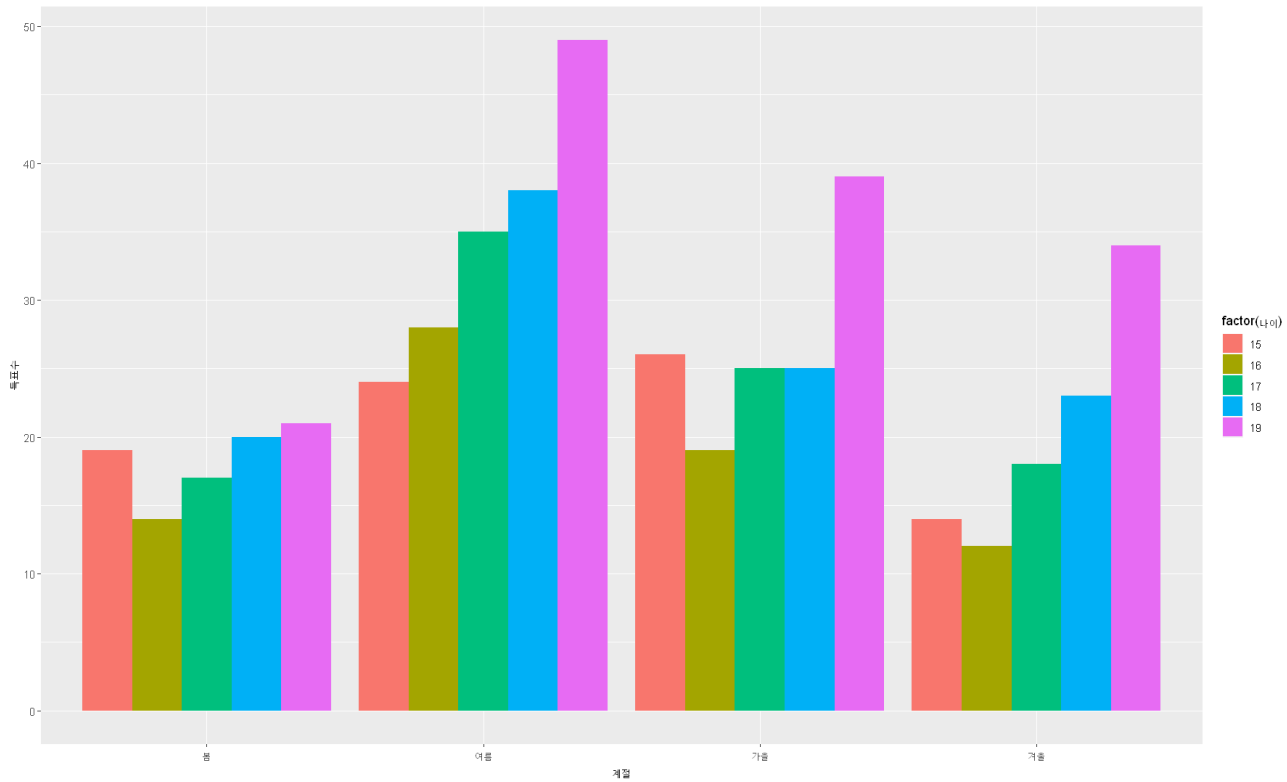
다중 평균 막대 그래프 표현하기

`x` 축과 `fill` 에는 값의 종류를 판별할 수 있는 문자열(`chr`)이나 요인(`fct`) 타입이 사용되어야 하고 `y` 축에는 수치를 판별할 수 있는 정수(`int`) 타입이 사용되어야 한다.

> 아래 코드에서는 **나이** 컬럼을 종류를 판별할 수 있는 요인(factor)로 변환하여 적용함.

```
graph <- ggplot(data=설문집계) +
            geom_col(aes(x=계절, y=득표수, fill=factor(나이)), position='dodge')
graph
```

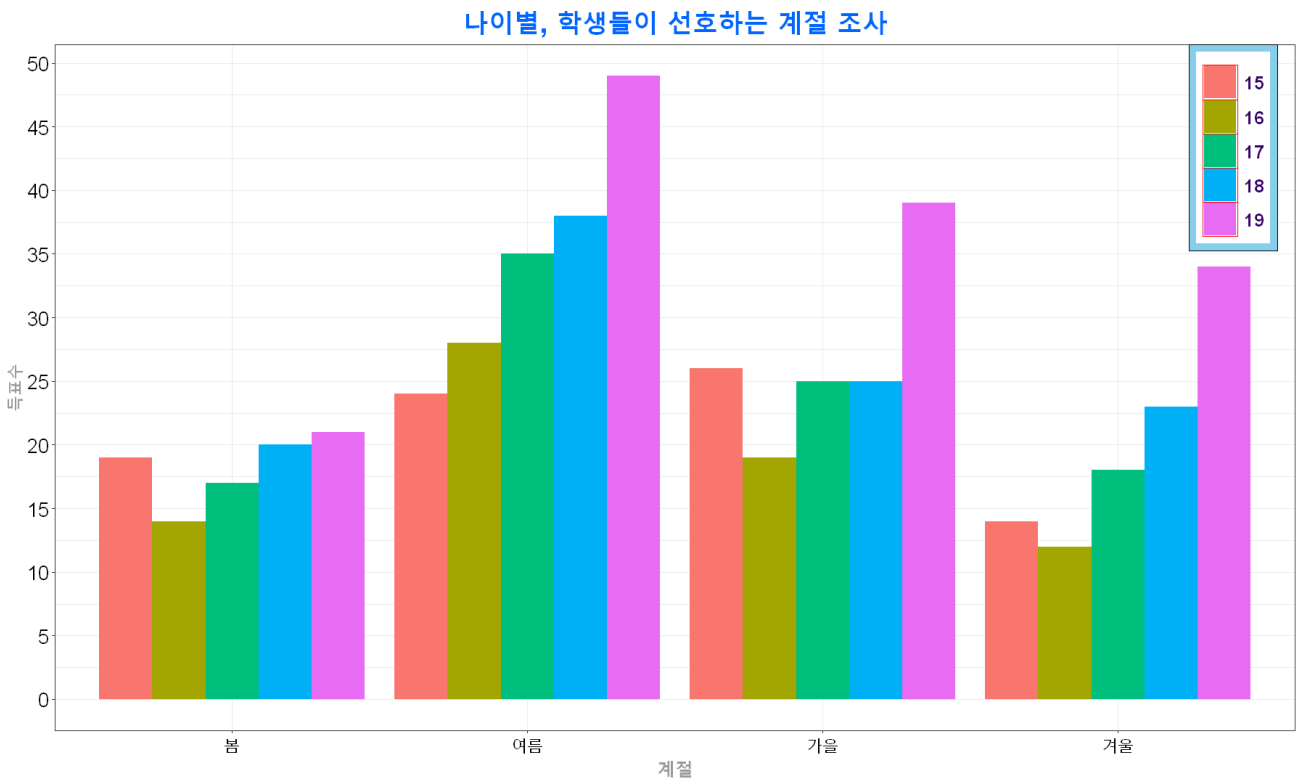
▶ 출력결과



완성된 그래프에 옵션 적용하기

```
graph +
    # 배경을 흰색으로 설정
    theme_bw() +
    # 그래프 타이틀 설정
    ggtitle("나이별, 학생들이 선호하는 계절 조사") +
    # x축 제목 설정
    xlab("계절") +
    # y축 제목 설정
    ylab("득표수") +
    # y축 간격 및 데이터에 대한 세자리 콤마 적용
    scale_y_continuous(breaks=seq(0, 50, 5))  +
    # 각 텍스트의 색상, 크기, 각도, 글꼴 설정
    theme(plot.title=element_text(family="NanumGothic", color="#0066ff",
                                  size=25, face="bold", hjust=0.5),
          axis.title.x=element_text(family="NanumGothic", color="#999999",
                                    size=18, face="bold"),
          axis.title.y=element_text(family="NanumGothic", color="#999999",
                                    size=18, face="bold"),
          axis.text.x=element_text(family="NanumGothic", color="#000000",
                                   size=16, angle=0),
          axis.text.y=element_text(family="NanumGothic", color="#000000",
                                   size=16, angle=0)) +
    # 범주 설정
    theme(legend.title = element_blank(),
              legend.text = element_text(face="bold", size=15, color="#330066"),
              legend.key = element_rect(color="red", fill="white"),
              legend.key.size = unit(1,"cm"),
              legend.box.background = element_rect(fill="skyblue"),
              legend.box.margin = margin(6, 6, 6, 6),
              legend.position = c(0.95, 0.85))
```

▶ 출력결과



4) 서로 다른 종류의 시각화를 일괄 표현하기

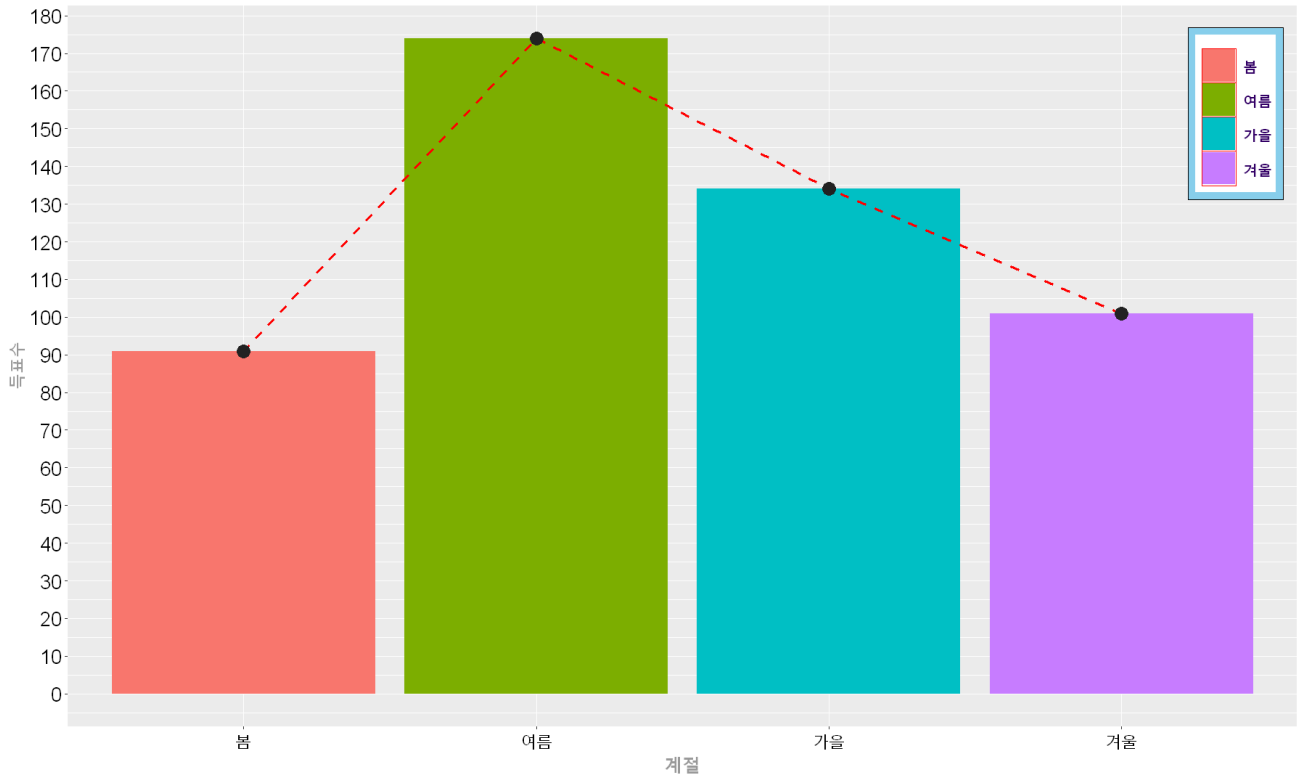
`geom_bar()`, `geom_col()`, `geom_line()`, `geom_point()` 등을 하나로 연결하면 하나의 시각화 자료에 다양한 형태의 그래프를 포함할 수 있다.

`geom_bar()`, `geom_col()` 함수는 x축에 종류를 구분 지을 수 있는 `chr`, `fct` 타입을 적용해야 하지만, `geom_line()`, `geom_point()` 함수는 연속성을 파악할 수 있는 `int` 형을 적용해야 한다ㅏ.

`geom_line()`, `geom_point()` 함수의 `aes()` 옵션에서 **group=TRUE**를 적용하면 `chr`, `fct` 타입으로도 선 그래프 표현이 가능해진다.

```
ggplot(data=계절투표df_이름변경) +
     geom_col(aes(x=계절, y=득표수, fill=계절)) +
     geom_line(aes(x=계절, y=득표수, group=TRUE), color="#ff0000", linetype=2, size=1) +
     geom_point(aes(x=계절, y=득표수, group=TRUE), size=5, color="#222222") +
     # y축 간격 및 데이터에 대한 세자리 콤마 적용
     scale_y_continuous(breaks=seq(0, 200, 10))  +
     # 각 텍스트의 색상, 크기, 각도, 글꼴 설정
     theme(plot.title=element_text(family="NanumGothic", color="#0066ff",
                                   size=25, face="bold", hjust=0.5),
           axis.title.x=element_text(family="NanumGothic", color="#999999",
                                     size=18, face="bold"),
           axis.title.y=element_text(family="NanumGothic", color="#999999",
                                     size=18, face="bold"),
           axis.text.x=element_text(family="NanumGothic", color="#000000",
                                    size=16, angle=0),
           axis.text.y=element_text(family="NanumGothic", color="#000000",
                                    size=16, angle=0)) +
     # 범주 설정
     theme(legend.title = element_blank(),
           legend.text = element_text(face="bold", size=15, color="#330066"),
           legend.key = element_rect(color="red", fill="white"),
           legend.key.size = unit(1,"cm"),
           legend.box.background = element_rect(fill="skyblue"),
           legend.box.margin = margin(6, 6, 6, 6),
           legend.position = c(0.95, 0.85))
```

▶ **출력결과**



5) 가로 막대 그래프

`coord_flip()` 함수를 적용하면 그래프가 옆으로 회전한 형태로 표시된다.

가로로 표시된다고 해서 **x**축이나 **y**축이 변경되지는 않는다.

```
ggplot(data=계절투표df_이름변경) +
     geom_col(aes(x=계절, y=득표수, fill=계절)) +
     # 가로 막대 그래프로 설정
     coord_flip() +
```

```
      # y축 간격 및 데이터에 대한 세자리 콤마 적용
      scale_y_continuous(breaks=seq(0, 200, 10))  +
      # 각 텍스트의 색상, 크기, 각도, 글꼴 설정
      theme(plot.title=element_text(family="NanumGothic", color="#0066ff",
                                    size=25, face="bold", hjust=0.5),
            axis.title.x=element_text(family="NanumGothic", color="#999999",
                                      size=18, face="bold"),
            axis.title.y=element_text(family="NanumGothic", color="#999999",
                                      size=18, face="bold"),
            axis.text.x=element_text(family="NanumGothic", color="#000000",
                                     size=16, angle=0),
            axis.text.y=element_text(family="NanumGothic", color="#000000",
                                     size=16, angle=0)) +
      # 범주 설정
      theme(legend.title = element_blank(),
                legend.text = element_text(face="bold", size=15, color="#330066"),
                legend.key = element_rect(color="red", fill="white"),
                legend.key.size = unit(1,"cm"),
                legend.box.background = element_rect(fill="skyblue"),
                legend.box.margin = margin(6, 6, 6, 6),
                legend.position = c(0.95, 0.85))
```

▶ 출력결과