📚 [LAB-06] 2. 데이터 분포 시각화 (1) - Boxplot, KDE

📘 데이터 분포 시각화 종류

그래프 유형	seaborn 함수	설명 / 목적
박스플롯	`boxplot()`	사분위수 기반 분포 요약
KDE 곡선	`kdeplot()`	연속형 분포의 밀도
히스토그램	`histplot()`	연속형 변수의 분포 확인
히스토그램 + KDE	`displot(kind="hist")` / `histplot(kde=True)`	분포 + 패턴
바이올린 플롯	`violinplot()`	분포 + 중앙값 + 퍼짐
스트립플롯	`stripplot()`	분포의 개별 관측값
스웜플롯	`swarmplot()`	겹치지 않는 스트립(마커) 분포
히트맵	`heatmap()`	복수 변수의 빈도/상관 분석

📘 #01. 준비작업

📝 1. 라이브러리 참조

```
from hossam import load_data
from matplotlib import pyplot as plt
from matplotlib import font_manager as fm
import seaborn as sb
import numpy as np
```

📝 2. 그래프 초기화

```
my_dpi = 200                                    # 이미지 선명도(100~300)
font_path = "./NotoSansKR-Regular.ttf"
        # 한글을 지원하는 폰트 파일의 경로
fm.fontManager.addfont(font_path)               # 폰트의 글꼴을 시스템에 등록
        함
font_prop = fm.FontProperties(fname=font_path) # 폰트의 속성을 읽어옴
```

```
font_name = font_prop.get_name()                # 읽어온 속성에서 폰트의 이름
        만 추출
plt.rcParams['font.family'] = font_name         # 그래프에 한글 폰트 적용
plt.rcParams['font.size']   = 10                # 기본 폰트 크기
plt.rcParams['axes.unicode_minus'] = False      # 그래프에 마이너스 깨짐 방지
```

📝 3. 데이터 가져오기

```
origin = load_data("employee_data_40")
origin.head()
```

```
 [94m[data] [0m https://data.hossam.kr/data/lab06/
employee_data_40.xlsx
 [94m[desc] [0m 어느 기업의 직원 40명을 대상으로 성별과 결혼상태, 나이, 최종학력,
월수입을 조사한 가상의 데이터(인덱스, 메타데이터 없음)
 [91m[!] Cannot read metadata [0m
```

	성별	결혼상태	나이	최종학력	월수입
0	남자	기혼	21	대학교	60
1	남자	기혼	22	대학원	100
2	남자	기혼	33	대학교	200
3	여자	미혼	33	대학교	120
4	남자	미혼	28	대학교	70

📘 #02. Boxplot

📝 1) 연속형 데이터의 분포를 사분위수 기반으로 확인

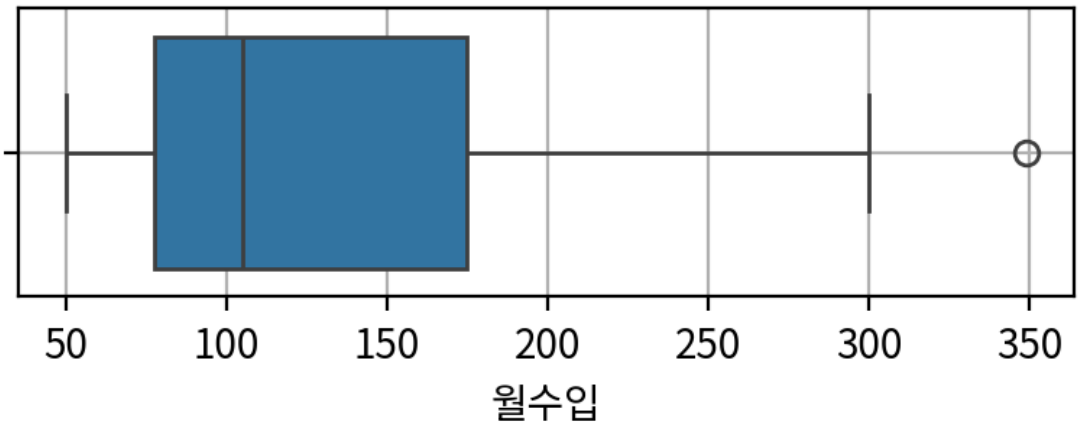
list, ndarray, Series 등 모든 연속형 객체를 `data` 파라미터에 지정한다.

`orient`로 방향을 설정할 수 있다.

- `v` : 세로 (기본값)
- `h` : 가로

```
# 1) 그래프 초기화 (캔바스(fig)와 도화지(ax) 준비하기)
width_px  = 800                         # 그래프 가로 크기
height_px = 350                         # 그래프 세로 크기
```

```
rows = 1                                    # 그래프 행 수
cols = 1                                    # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=origin['월수입'], orient="h")

# 3) 그래프 꾸미기
ax.grid(True)                                       # 격자 표시

# 4) 출력
plt.tight_layout()                                  # 여백 제거
plt.show()                                          # 그래프 화면 출력
plt.close()                                         # 그래프 작업 종료
```



png

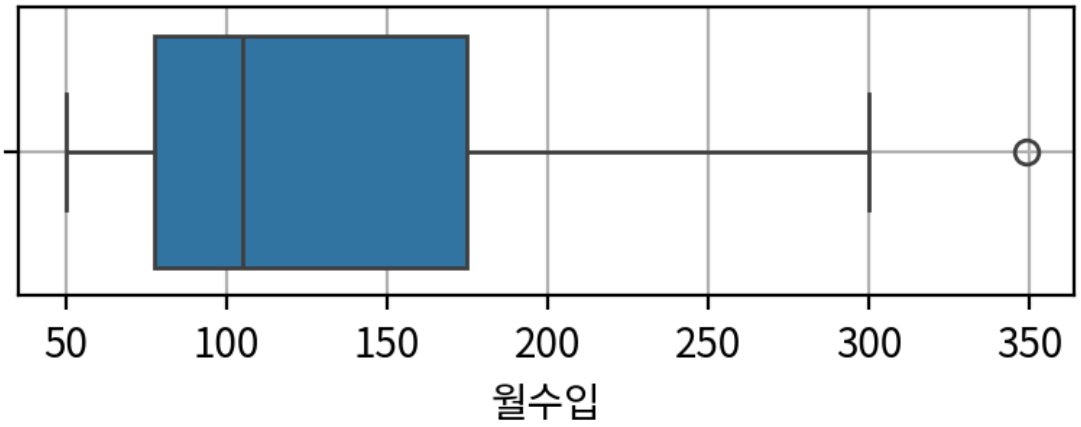
📝 2. 데이터프레임을 통한 상자그림

`data` 파라미터에 데이터 프레임을 설정하고 `y` 파라미터에 표시하고자 하는 변수 이름을 문자열로 설정한다.

`x` 파라미터에 설정할 경우 가로 상자그림으로 표시된다.

```
# 1) 그래프 초기화 (캔바스(fig)와 도화지(ax) 준비하기)
width_px  = 800                             # 그래프 가로 크기
height_px = 350                             # 그래프 세로 크기
rows = 1                                    # 그래프 행 수
cols = 1                                    # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)
```

```
# 2) 그래프 그리기
sb.boxplot(data=origin, x='월수입')

# 3) 그래프 꾸미기
ax.grid(True)                                   # 격자 표시

# 4) 출력
plt.tight_layout()                              # 여백 제거
#plt.savefig("myplot.png", dpi=my_dpi)          # 생략 가능
plt.show()                                      # 그래프 화면 출력
plt.close()                                     # 그래프 작업 종료
```



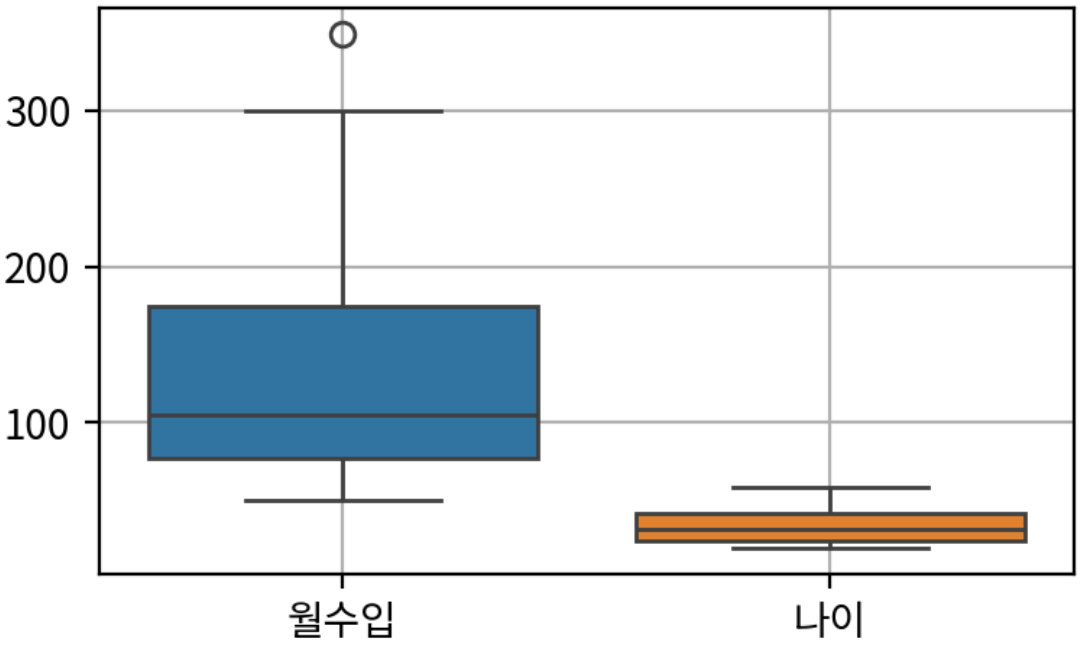
png

📝 3) 복수 변수에 대한 처리

표시하고자 하는 변수를 필터링하여 data 파라미터에 설정한다.

```
# 1) 그래프 초기화 (캔바스(fig)와 도화지(ax) 준비하기)
width_px  = 800                         # 그래프 가로 크기
height_px = 500                         # 그래프 세로 크기
rows = 1                                # 그래프 행 수
cols = 1                                # 그래프 열 수
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(rows, cols, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.boxplot(data=origin[['월수입', '나이']])

# 3) 그래프 꾸미기
ax.grid(True)                                   # 격자 표시

# 4) 출력
```

```
plt.tight_layout()                    # 여백 제거
plt.show()                            # 그래프 화면 출력
plt.close()                           # 그래프 작업 종료
```



png

📘 #03. KDE(커널 밀도 추정) Plot

- 연속형 데이터의 **분포를 부드러운 곡선 형태로 추정**하는 비모수적 방법.
- 데이터의 전체적인 분포 모양, 봉우리(peak), 꼬리(tail) 등을 확인하는 데 유용함.

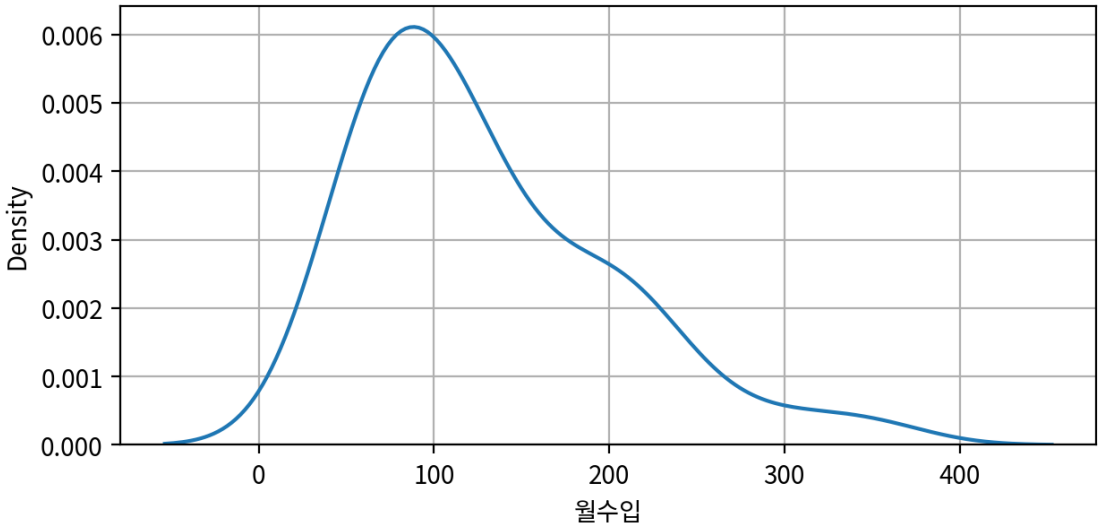
📝 비모수적 방법이란?

- “데이터가 정규분포일 것이다” 같은 가정이 없음
- 대신 데이터 자체를 기반으로 분포의 모양을 추정함
- 필요한 것은 파라미터(평균·분산 등)보다 데이터의 구조와 패턴

📝 1. 연속성 데이터 설정

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)
```

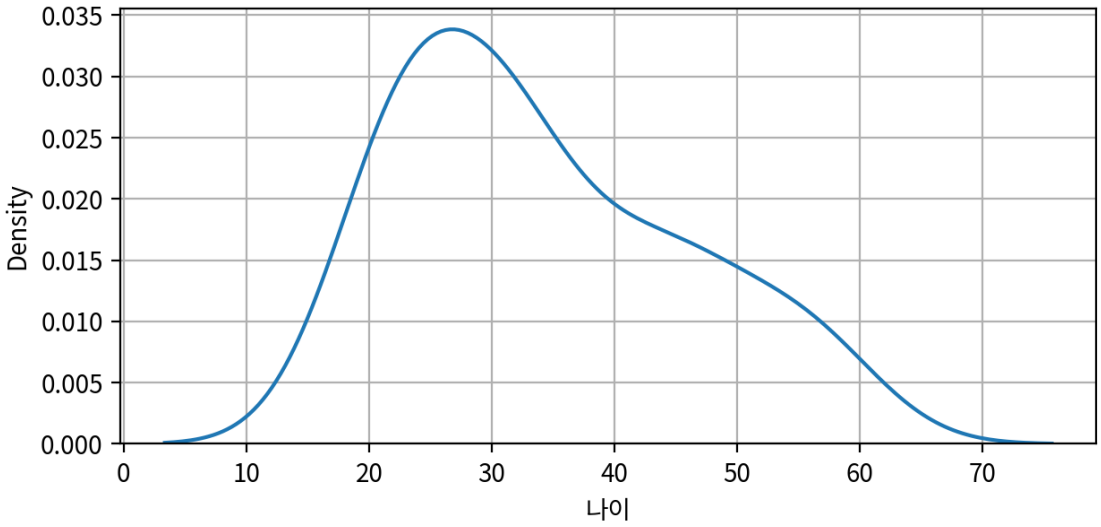
```
# 2) 그래프 그리기
sb.kdeplot(data=origin['월수입'])
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png

📝 2. 데이터 프레임 자체를 적용하기

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='나이')
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```

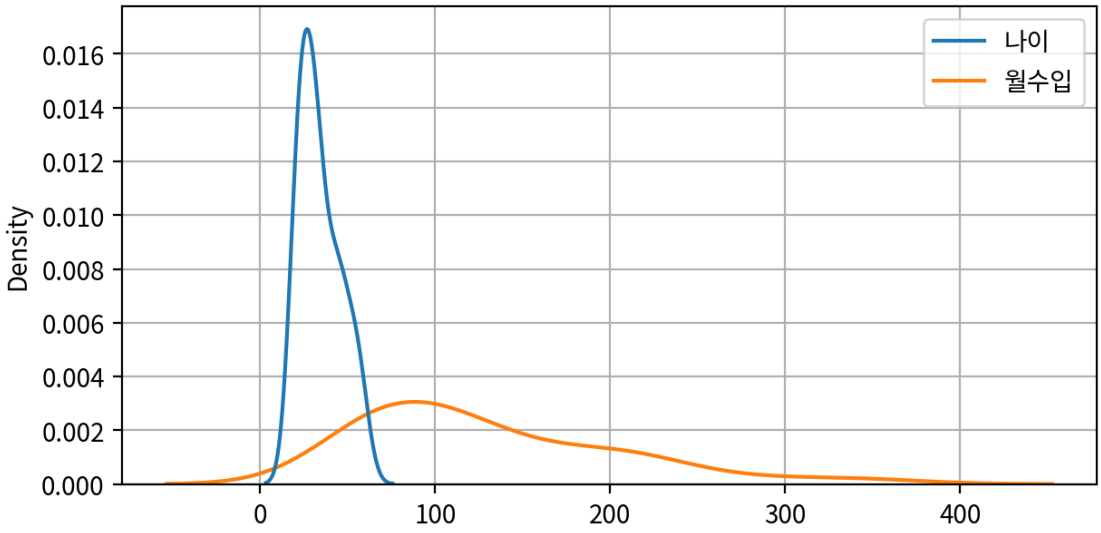


png

📝 3. 다중 분포

데이터프레임을 `data` 파라미터에 적용하면서 `x`나 `y`파라미터를 지정하지 않으면 모든 연속형 변수에 대한 커널 밀도 곡선이 표현된다.

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



png

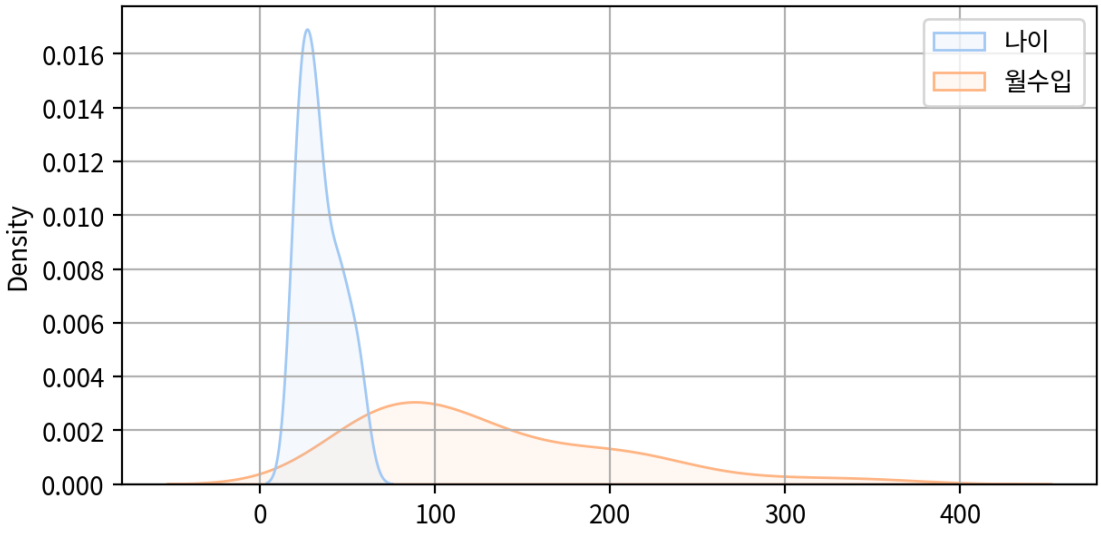
📝 4. 색상 채우기

`fill=True` 파라미터를 설정하면 곡선 내부에 색상이 표시된다.

이 때, `alpha` 파라미터를 `0~1`사이의 값으로 설정하여 면의 투명도를 조절할 수 있다.

`0`=투명, `1`=불투명

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, fill=True, alpha=0.1, palette="pastel")
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```

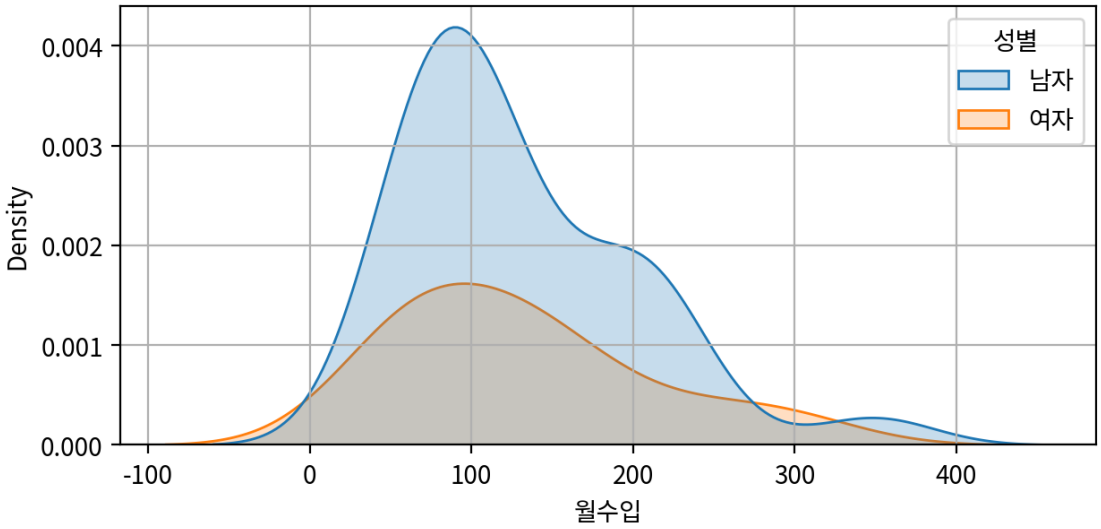


png

📝 5. 범주에 따른 구분

hue 파라미터에 명목형 변수의 이름을 지정하면 범주에 따라 그래프를 분기한다.

```
# 1) 그래프 초기화
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

# 2) 그래프 그리기
sb.kdeplot(data=origin, x='월수입', hue='성별', fill=True)
ax.grid(True)

# 3) 출력
plt.tight_layout()
plt.show()
plt.close()
```



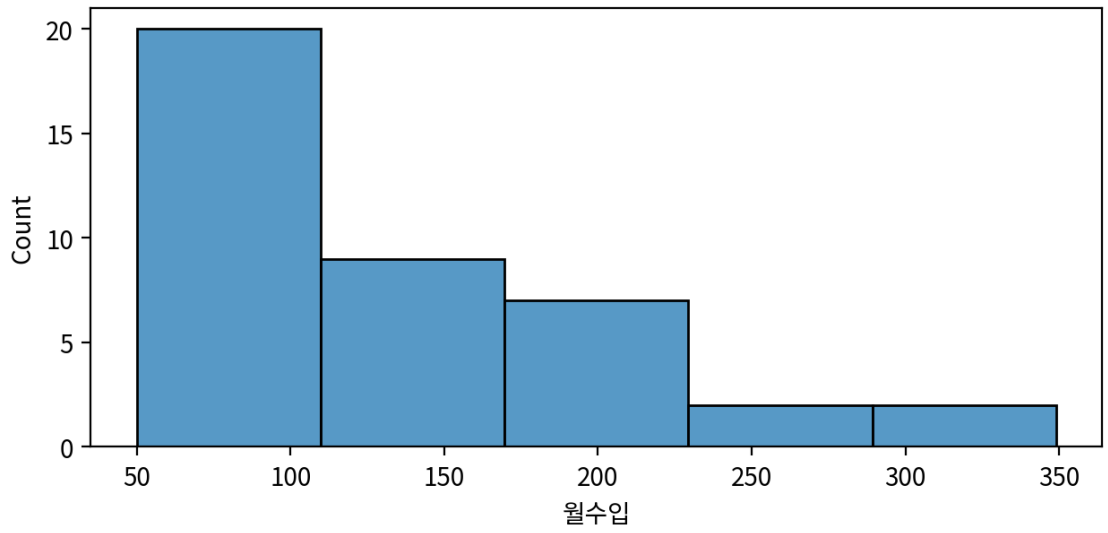
png

📘 #04. Histogram

도수 분포표를 시각화 한 그래프

📝 1. 구간 수 지정하기

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5)
# ax.grid(True) ⟵ 비추

plt.tight_layout()
plt.show()
plt.close()
```



png

📝 2. 구간을 표시하기

```
hist, bins = np.histogram(origin['월수입'], bins=5)
print(hist)
print(bins)
```

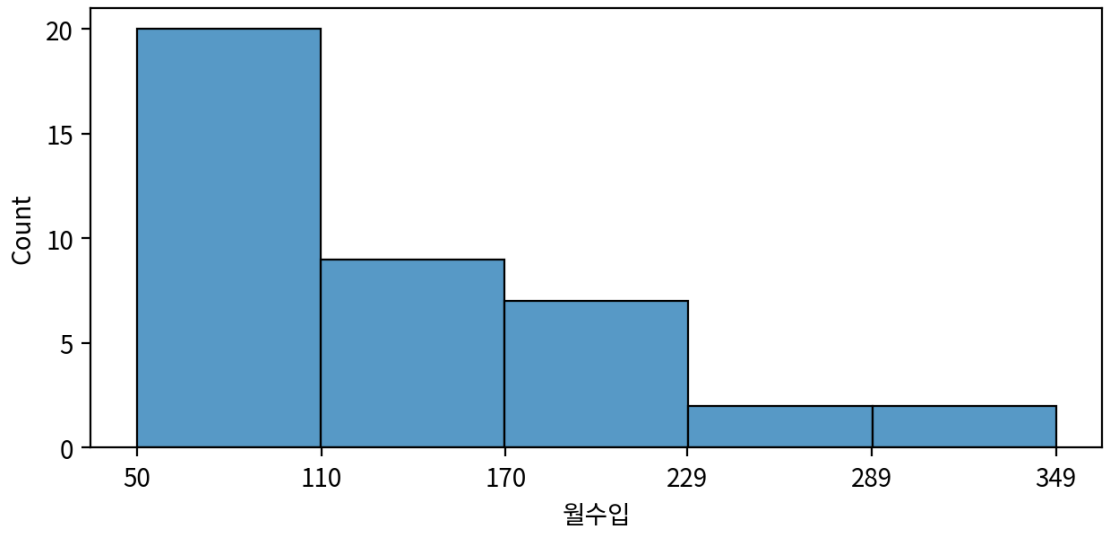
```
[20  9  7  2  2]
[ 50.  109.8 169.6 229.4 289.2 349. ]
```

```
bins = bins.round().astype("int")
bins
```

```
array([ 50, 110, 170, 229, 289, 349])
```

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=5, edgecolor="#000000",
        linewidth=0.8)
ax.set_xticks(bins, bins)
# ax.grid(True) ⟵ 비추

plt.tight_layout()
```

```
plt.show()
plt.close()
```

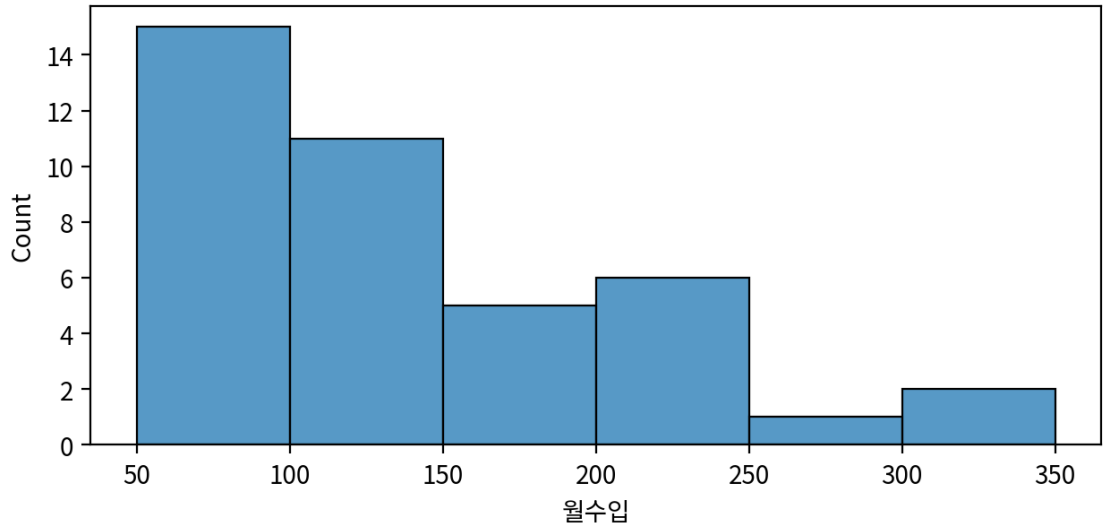


png

📝 3. 구간을 직접 지정하기

`bins` 파라미터에 구간을 의미하는 리스트를 지정한다.

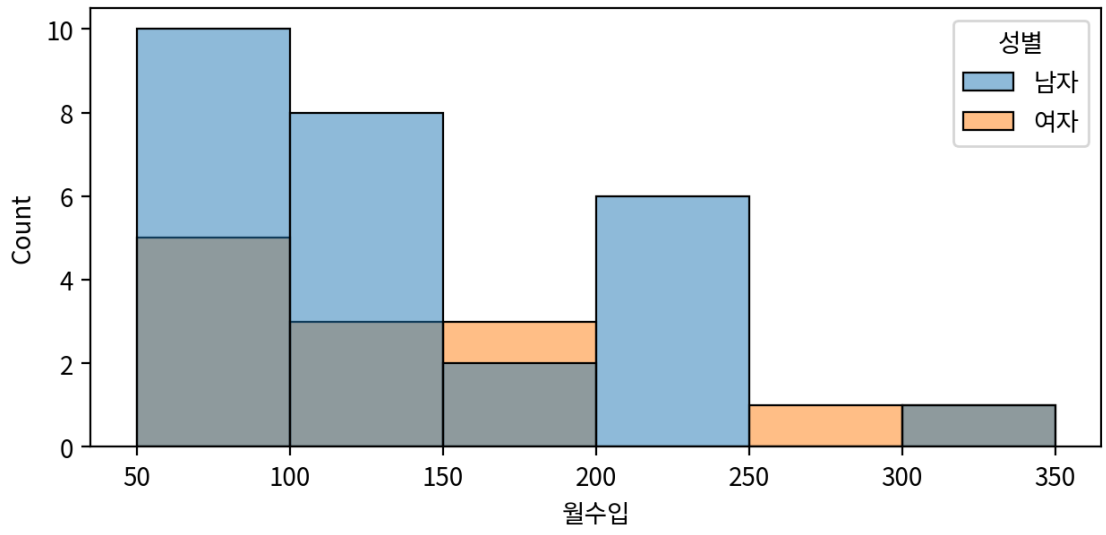
```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin['월수입'], bins=[50, 100, 150, 200, 250, 300,
        350],
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

📝 4. 범주에 따른 구분

```
width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=origin, x='월수입', hue='성별',
            bins=[50, 100, 150, 200, 250, 300, 350],
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



png

📝 5. 히스토그램의 계급의 수와 간격의 크기

우리에게 주어지는 데이터는 종류도 다양하며 관측치의 개수도 다양하다.

또한 그 데이터의 최솟값과 최댓값 또한 다양하다. 따라서 데이터가 주어졌을 때 계급의 개수와 계급의 간격을 설정하는 것에 어려움이 있을 수 있다.

(1) 계급의 수를 구하는 방법

스터지스의 공식(Sturges' formula)

히스토그램의 계급의 수를 결정하는 공식

계급구간수 = $1 + 3.3\log_n$

> n = 관측치의 수

관측치의 수에 따른 계급구간 수 표 활용

통계학에서 일반적으로 활용하는 표

관측치의 수	계급구간의 수
50 미만	5 ~ 7
50 ~ 200	7 ~ 9
200 ~ 500	9 ~ 10
500 ~ 1,000	10 ~ 11
1,000 ~ 5,000	11 ~ 13
5,000 ~ 50,000	13 ~ 17
50,000 초과	17 ~ 20

(2) 간격의 크기를 구하는 방법

계급구간의간격 = 관측치최대값 – 관측치최소값/계급구간의수

여기서 중요한 지점은 계급구간의 첫 시작지점을 어떻게 잡아야 하는지이다.

이 때 필수적으로 알아야 하는 점은 첫 계급구간은 반드시 최소 관측치를 포함해야 한다는 점이다.

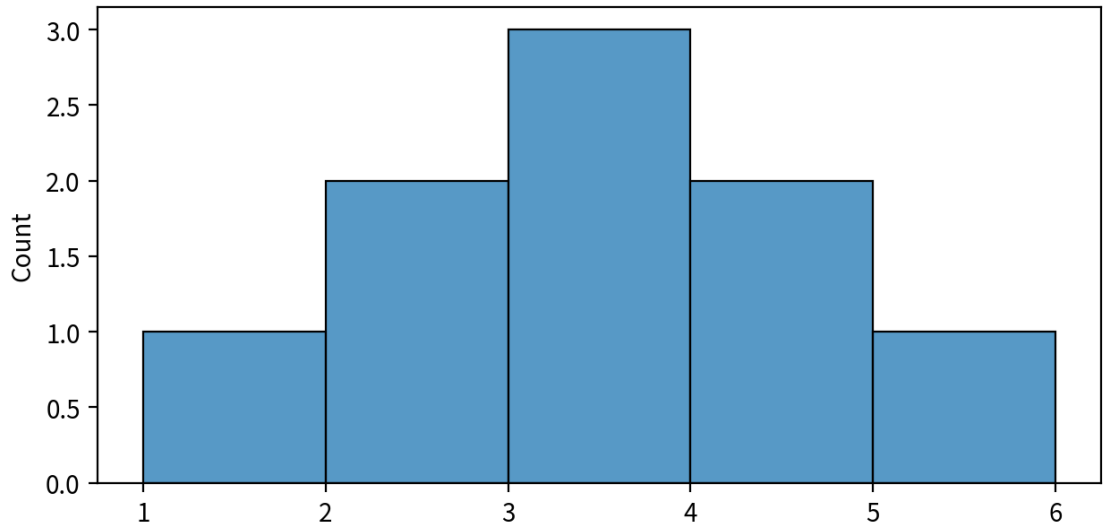
(3) 히스토그램의 모습

대칭성

정중앙 부분에 선을 수직으로 긋고 절반으로 접었을 때 일치하는 그래프

특히 정삼각형에 가까운 형태를 **종모양**이라고 하는데 이 그래프는 정규분포와 밀접한 관련이 있기 때문에 굉장히 중요한 히스토그램의 모습이라고 할 수 있다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 5]

width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

plt.tight_layout()
plt.show()
plt.close()
```



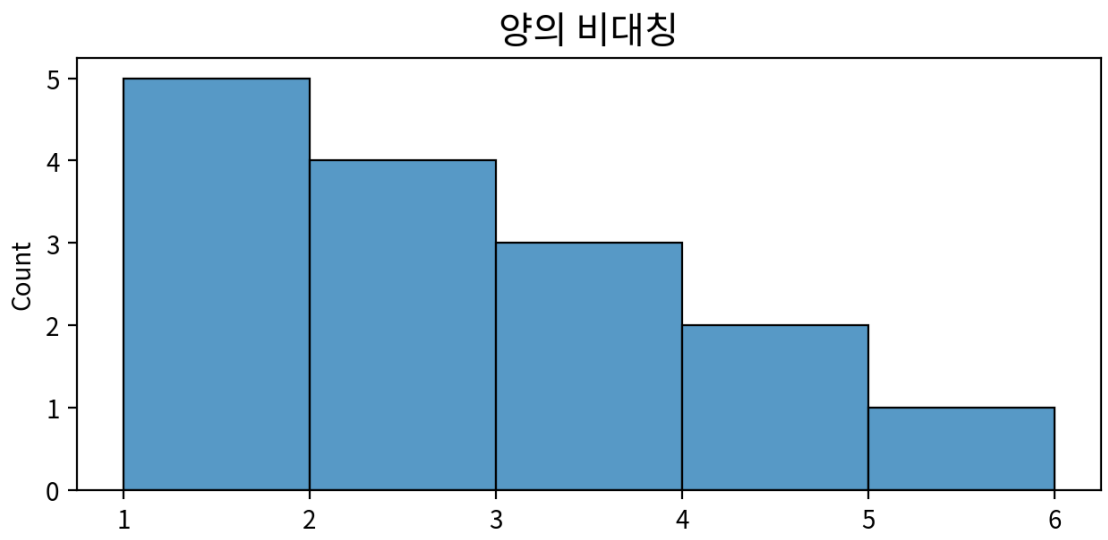
png

비대칭성

그래프의 모양을 보면 점점 작아지는 히스토그램의 모양을 볼 수 있는데, 점점 작아지는 방향을 꼬리라고 칭하며 꼬리가 양의 방향으로 향해 있다면 양의 비대칭이며 꼬리가 음의 방향을 향해 있다면 음의 비대칭이라고 할 수 있다. 밑의 그림은 양과 음의 비대칭 히스토그램의 모습이다.

양의 비대칭은 평균이 중앙값보다 작으며 음의 비대칭은 평균이 중앙값보다 크다.

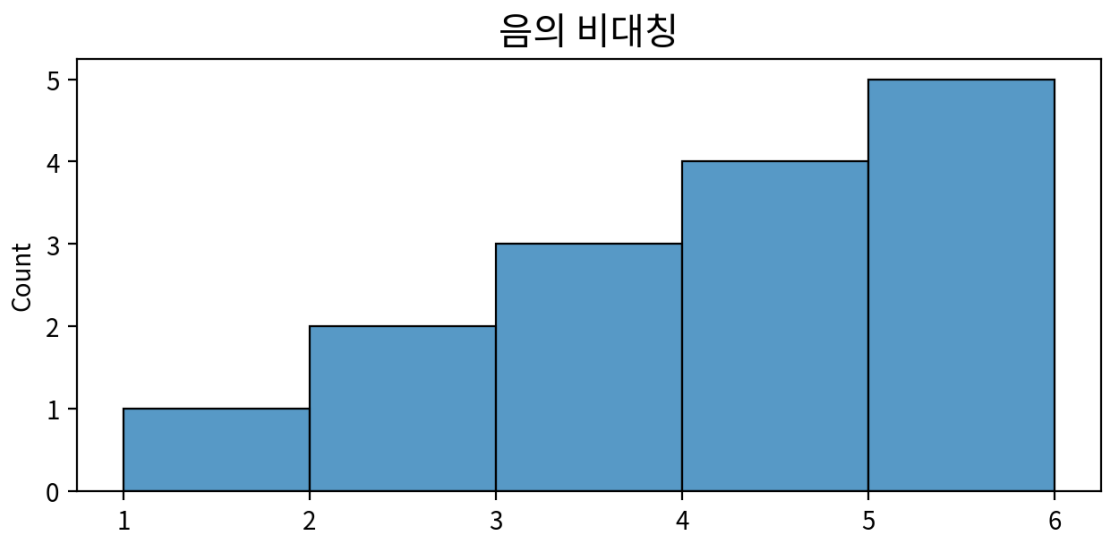
```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 5]

width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()
```



png

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5]

width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)
```

```
ax.set_title("음의 비대칭", fontsize=15)

plt.tight_layout()
plt.show()
plt.close()
```



png

봉우리 계급 구간의 수

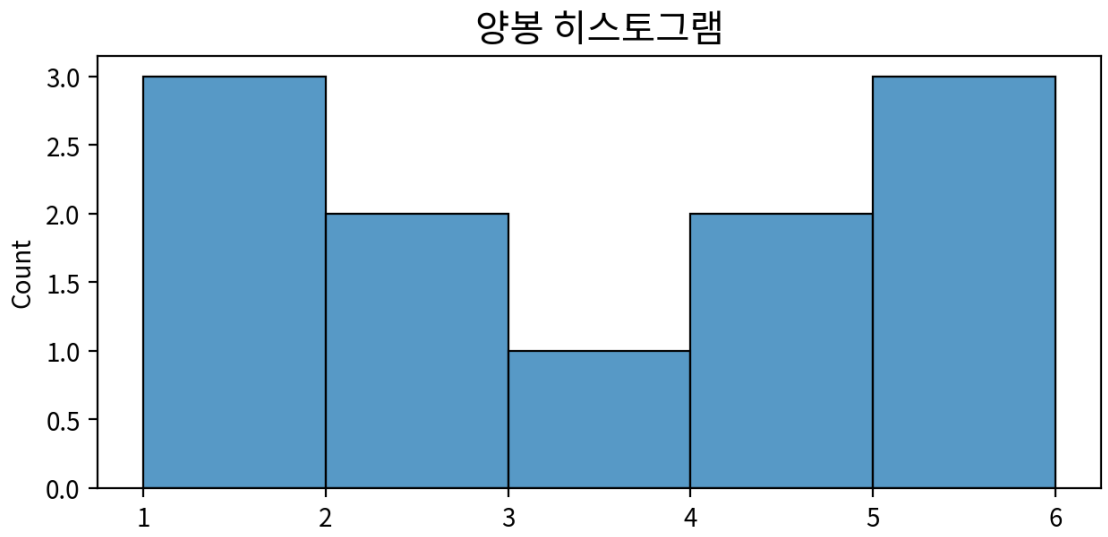
히스토그램에서 가장 높은 도수를 나타내고 있는 수치를 최빈값(mode)이라고 부른다.

최빈계급이란 최대의 관측치 수를 가진 계급이다.

만일 최빈계급이 하나일 경우에는 단봉을 가진 히스토그램 이라고 불리며 최빈계급이 두 개일 경우에는 양봉을 가진 히스토그램이라고 불린다.

```
mybins = [1, 2, 3, 4, 5, 6]
data = [1, 1, 1, 2, 2, 3, 4, 4, 5, 5, 5]

width_px  = 1280
height_px = 640
figsize = (width_px / my_dpi, height_px / my_dpi)
fig, ax = plt.subplots(1, 1, figsize=figsize, dpi=my_dpi)

sb.histplot(data=data, bins=mybins,
            edgecolor="#000000", linewidth=0.8)

ax.set_title("양봉 히스토그램", fontsize=15)

plt.tight_layout()
```

```
plt.show()
plt.close()
```



png

```
```