데이터시각화이해와실습

Lecture 11. seaborn 라이브러리

matplotlib --> seaborn(pandas의 Series, DataFrame과 호환성이 높음)

동덕여자대학교
데이터사이언스 전공
권 범

목차

01. 데이터 시각화 이해

02. seaborn 라이브러리

01. 데이터 시각화 이해

❖ 시작하기 전에

- 데이터 수가 많아지면, 눈으로 각 데이터를 일일이 살펴보기가 어려워지며 데이터 속에 담긴 의미를 파악하는 것도 어려워짐
- 데이터의 의미를 파악하고 파악한 의미를 보다 효과적으로 전달하기 위해서, 데이터 시각화를 하나의 수단으로 사용할 수 있음
- 실제로, 다양한 분야에서 데이터 시각화를 활용하는 사례가 늘어나고 있음

먼저, 데이터 시각화의 장점과 활용 사례들에 대하여 살펴보자

01. 데이터 시각화 이해

❖ 데이터 시각화 장점

- 기술적인 데이터 분석도 중요하지만, 분석 결과 기반의 가치 창출이 데이터 분석의 핵심임
- 데이터 시각화를 효과적으로 활용하면, 가치 창출에 도움이 될 수 있음
- 데이터 분석을 위해서는 데이터 수집, 가공, 분석 기법을 활용한 기술 역량이 필요함
- 반면, 데이터 시각화에는 데이터 분석 결과를 시각적으로 표현해서 스토리텔링하는 역량이 필요함
- 최근에는 데이터 시각화 활용이 쉬워지면서, 고도의 전문 역량이 없더라도
  데이터를 활용하여 인사이트를 발견하고, 스토리텔링할 수 있게 되었음

데이터 시각화 장점에 대해 알아 보자

01. 데이터 시각화 이해

❖ 데이터 시각화 장점: ① 많은 양의 데이터를 시각적으로 요약 (1/2)

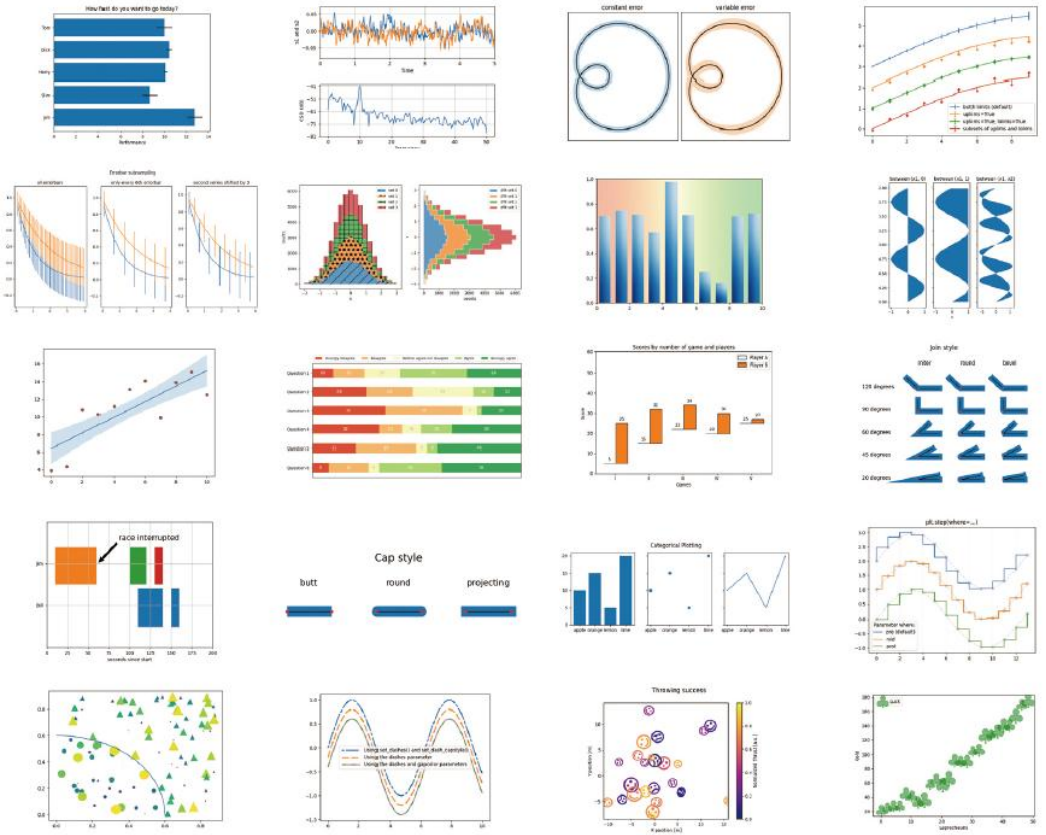
- 컴퓨터 모니터 화면을 벗어날 정도로 많은 양의 데이터는 그 특징을 전달하기 어려움
- 이런 경우, 선, 도형, 색 등의 시각적 요소들을 활용하여 시각화하게 되면,
  많은 양의 데이터를 요약해 표현하는 것이 가능함
- 이렇게 하면 많은 양의 데이터도, 한눈에 볼 수 있게 됨
- 특히, 금융, 교통, 의료 등의 분야에서는 날마다 방대한 양의 데이터가 생성되며,
  시각화 없이는 현상을 파악하고 예측하기 어렵기 때문에
  데이터 시각화가 매우 중요함

01. 데이터 시각화 이해

❖ 데이터 시각화 장점: ① 많은 양의 데이터를 시각적으로 요약 (2/2)

- 데이터 시각화 유형에는 선, 막대, 파이 차트 등의 다양한 시각화 유형이 있음
- 시각화 유형은 어떤 데이터를 어떻게 표현할 것인지에 따라 구분됨
- 차트에 활용되는 도형의 종류, 크기, 위치, 색 등으로 데이터의 크기를 비교하거나 분포를 파악하여 데이터의 의미를 찾을 수 있음

데이터 시각화 유형 예



[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 시각화 이해

❖ 데이터 시각화 장점: ② 시각을 통한 데이터 인사이트 도출 (1/3)

- 시각화 차트를 볼 때, 도형의 형태, 크기, 위치, 색 정보 등의 시각화 요소로부터 시각적 패턴을 찾음
- 선 그래프에서는 선의 높낮이, 기울기 형태로부터 데이터의 변화 추세를 파악하고,
  다른 값들과 구별되는 이상치가 있는지 등을 파악함
- 막대 그래프, 파이 차트 등 대부분의 다른 시각화 차트들도
  시각화 요소로부터 데이터의 의미를 파악하게 됨

01. 데이터 시각화 이해

❖ 데이터 시각화 장점: ② 시각을 통한 데이터 인사이트 도출 (2/3)

- 데이터 시각화에서 주의할 점은 데이터 인사이트의 정확한 전달을 위해, 시각화 형태를 잘 선택해야 한다는 점임
- 데이터 인사이트에 적합한 시각화 유형을 선택하고 해석에 오류가 발생하지 않도록, 시각화 차트를 작성해야 함

시각화 유형을 선택할 때는,
데이터 시각화 목적을 명확히 해야 함

01. 데이터 시각화 이해

❖ 데이터 시각화 장점: ② 시각을 통한 데이터 인사이트 도출 (3/3)

- 데이터 시각화의 기본적인 목적은 데이터의 크기를 비교하는 것임
- 데이터 비교 시각화 차트에는 누적 막대 그래프, 그룹 막대 그래프, 선 그래프, 영역 차트 등이 있음
- 구성의 비중을 보는 차트에는 파이 차트, 100% 누적 막대 그래프, 폭포 차트 등이 있음
  - ◆ 구성 비중 차트는 항목이 많아지게 되면 항목간 비교가 어려워지기 때문에, 항목이 4~5개 정도일 때 효과적으로 사용될 수 있음
- 데이터 분포를 확인하는 차트에는 산점도, 히스토그램 등이 있음

01. 데이터 시각화 이해

❖ **데이터 시각화 장점: ③ 더 정확한 데이터 분석 결과 도출**

- 데이터 시각화는 데이터 분석 결과를 전달하기 위한 목적으로도 사용됨
- 즉, 데이터 분석에서 시각화는 데이터의 정확한 이해를 돕고, 데이터 인사이트 발견을 위한 필수 요소임
- 이 외에도, 데이터 시각화는 데이터 탐색을 위한 방법으로도 활용될 수 있음
- 즉, 변수, 수치 계산 방식, 차트 유형 등의 조건을 변경하면서 데이터를 시각화 차트로 표현하고, 차트에 표현되는 시각적 패턴을 토대로 데이터를 다양한 각도에서 탐색함
- 데이터 탐색을 통해 발견한 내용을 토대로, 데이터 분석을 올바르게 수행할 수 있으며 이는 정확한 데이터 분석 결과 도출에 도움이 됨

01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ① 시각화 대시보드를 통한 공유 데이터 활용 (1/3)

- 보고서 또는 프레젠테이션을 통해 데이터 인사이트를 전달 및 공유할 때,
  데이터 시각화 차트를 활용하면 전달하고자 하는 메시지를 효과적으로 전달 할 수 있게 됨
- 시각화 차트를 효과적으로 활용하면, 전달하고자 하는 스토리텔링에 큰 공감을 일으킬 수 있음

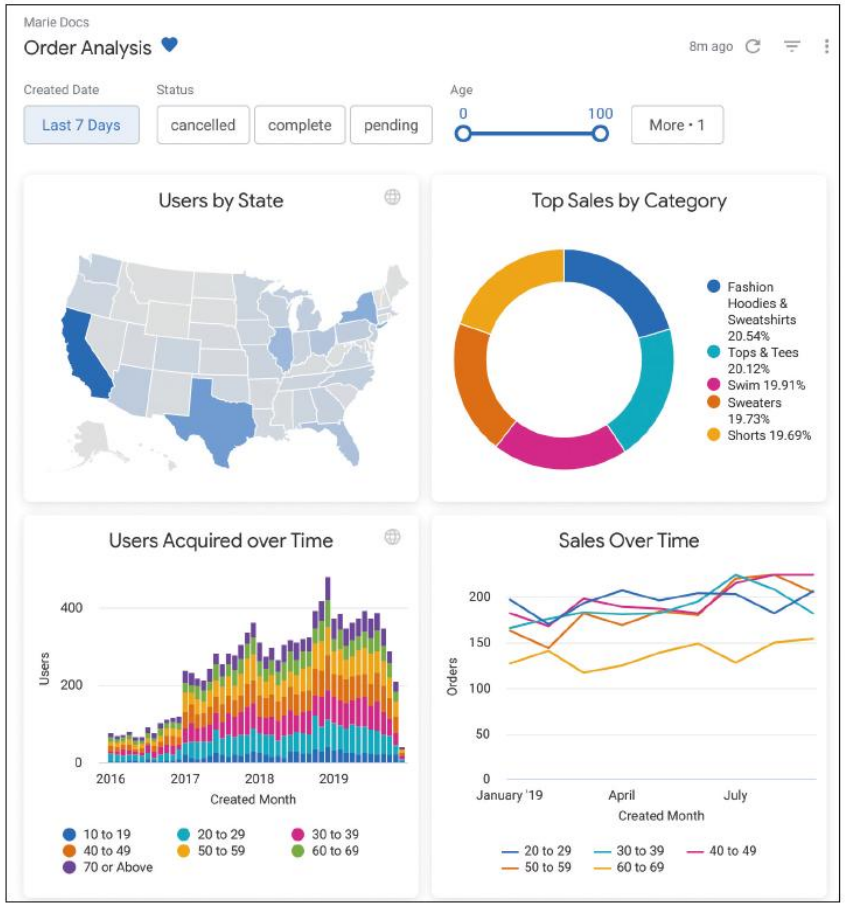
데이터 시각화 결과를 공유하는 방식 중 하나인
시각화 대시보드(Dashboard)에 대해 알아 보자

01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ① 시각화 대시보드를 통한 공유 데이터 활용 (2/3)

- 시각화 대시보드는 시각화 차트와 표 등으로 구성된 하나의 보드로, 시각화 대시보드를 이용하면 주요한 데이터 지표를 모니터링하고 탐색할 때 유용함
- 데이터 조회 기간을 선택하거나, 특정 기준으로 데이터를 조회할 수 있는 필터와 같은 상호작용 기능을 제공함
- 이러한 기능을 이용하면, 복잡한 데이터 분석 기술 없이도 편리하게 데이터를 조회하고 탐색할 수 있게 됨

시각화 대시보드 예



[사진출처] 데이터 분석을 위한 전처리와 시각화 with 파이썬 (출판사: 길벗캠퍼스)

01. 데이터 시각화 이해

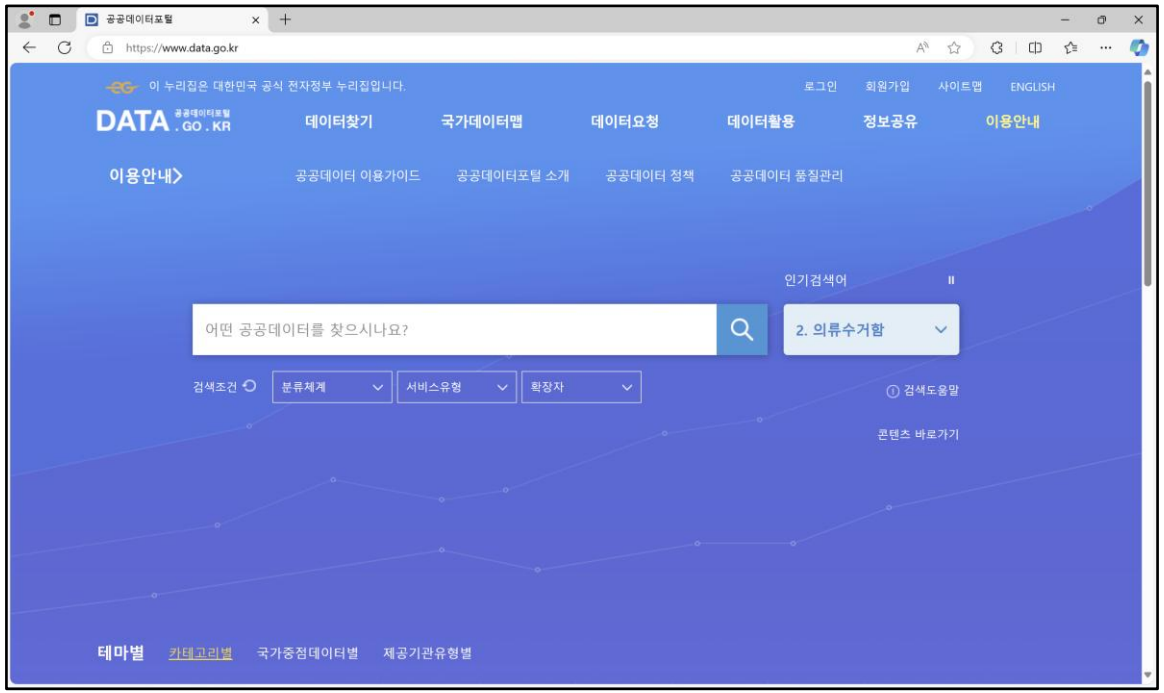
❖ 데이터 시각화 활용 사례: ① 시각화 대시보드를 통한 공유 데이터 활용 (3/3)

- 시각화 대시보드는 기업 및 조직에서 유용하게 활용되는데,
  전문 데이터 분석가가 아니어도 쉽게 데이터를 활용할 수 있음
- 개인마다 서로 다른 관점에서 데이터를 탐색할 수 있기 때문에
  더욱 다양한 데이터 인사이트를 도출할 수 있음
- 이는 조직 전체의 관점에서 의사결정의 근거로 활용되어,
  새로운 기회 도출 및 가치 창출에 큰 역할을 함

01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ② 다양한 분야에서의 시각화 활용 (1/5)

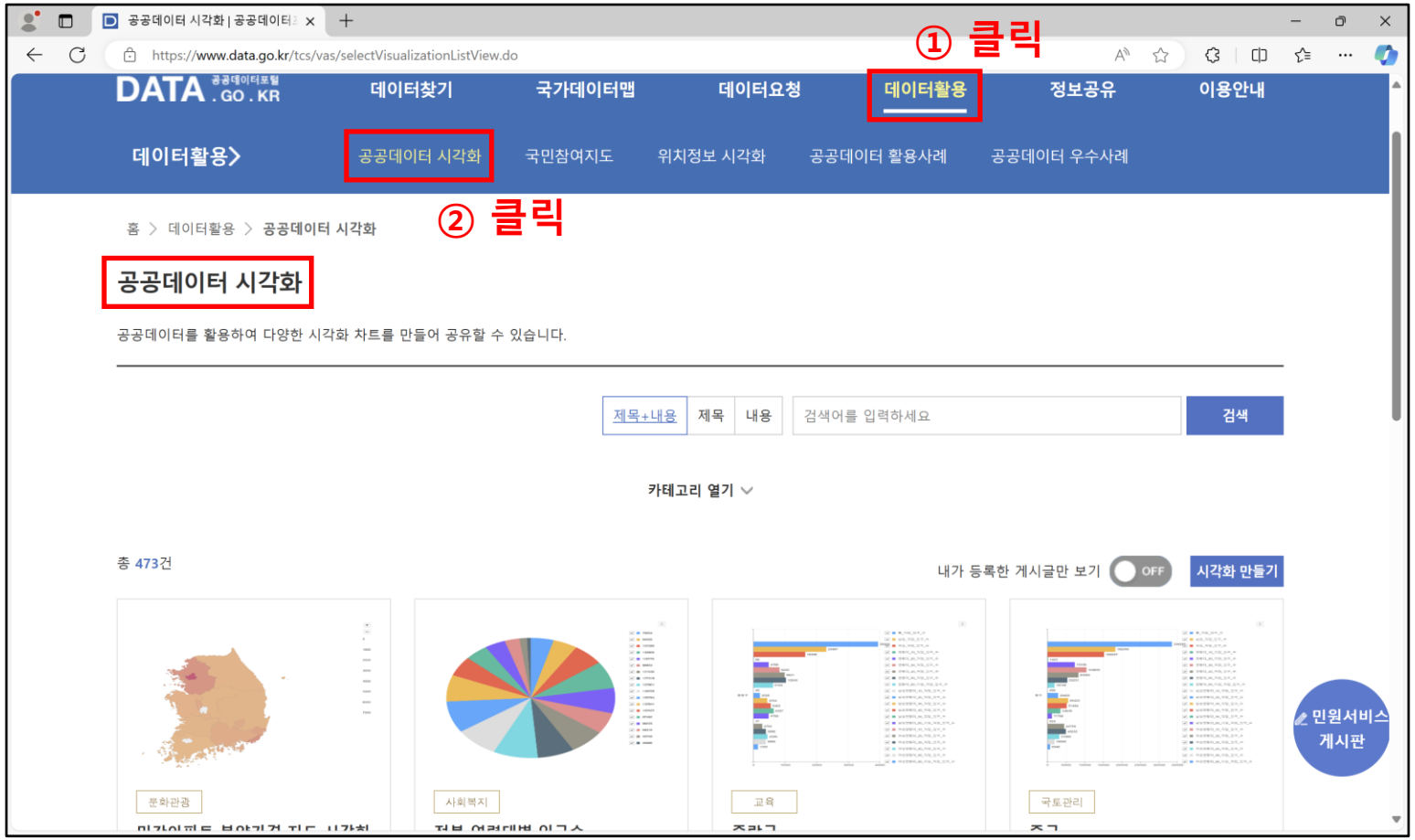
- 데이터 시각화는 데이터가 존재하는 모든 분야에서 활용될 수 있음
- 우리 나라 정부에서도 『공공데이터의 제공 및 이용 활성화에 관한 법률』에 의거하여, 국가에서 보유하고 있는 다양한 데이터의 제공 및 이용 활성화를 위해 관련 서비스를 제공하고 있음
- 대표 사례로, 공공데이터포털 사이트(https://www.data.go.kr/)가 있음



01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ② 다양한 분야에서의 시각화 활용 (2/5)

- 공공데이터포털에서는 제공하는 시각화 서비스를 살펴보자
- [공공데이터포털] – [데이터활용] – [공공데이터 시각화]를 클릭



01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ② 다양한 분야에서의 시각화 활용 (3/5)

- 공공 부문 외에 민간 부문에서도 데이터 시각화를 활용하고 있음
- 다양한 산업의 여러 기업들은 각자 보유한 데이터의 활용도를 높이기 위해, 데이터 시각화를 적극 도입하고 있음
- 시각화 대시보드를 구축하여, 대시보드를 의사결정 목적으로 활용하고 있음

01. 데이터 시각화 이해

❖ 데이터 시각화 활용 사례: ② 다양한 분야에서의 시각화 활용 (4/5)

- 금융 분야에서도 데이터 분석을 활용하는 사례가 점점 늘고 있음
- 수익 창출을 위해서, 데이터 분석을 활용하여 고객을 분석하는 비중이 점점 증가하는 추세임
- 즉, 고객 행동 예측, 고객 만족도 개선, 금융 상품 마케팅 등에 데이터 분석을 활용하고 있음
- 특히, 조직 단위별 영업 실적 모니터링, 고객 특성에 따른 비정형 분석과 같은 내용을 그래프로 시각화는 사례가 늘어나고 있음

01. 데이터 시각화 이해

❖ **데이터 시각화 활용 사례: ② 다양한 분야에서의 시각화 활용 (5/5)**

- 언론 분야에서도 데이터 분석을 기반으로 스토리텔링 기사를 내보낼 때, 시각화 차트를 이용하고 있음
- 단순히 수치를 활용하는 것 외에도, 데이터 분석을 통해 찾아낸 새로운 사실을 보도하는 데이터 저널리즘을 실현하는데 데이터 시각화를 활용하고 있음

그럼 이제, 시각화 라이브러리 중 하나인 seaborn 라이브러리에 대해 알아 보자

02. seaborn 라이브러리

01. 데이터 시각화 이해

02. seaborn 라이브러리

❖ 시작하기 전에 (1/4)

- seaborn(시본) 라이브러리는 matplotblit(맷플롯립) 라이브러리를 기반으로 다양한 테마와 통계용 차트 등의 동적인 기능을 추가한 시각화 라이브러리임
- seaborn 라이브러리는 matplotlib 라이브러리와 다르게 통계와 관련된 차트를 제공하기 때문에 데이터프레임으로 다양한 통계 지표를 시각화할 수 있어, 널리 사용되고 있음

팔레트(palette) -> 색상

02. seaborn 라이브러리

❖ 시작하기 전에 (2/4)

- seaborn 라이브러리로 그리는 그래프들은 세 가지 범주로 분류할 수 있음
  - ◆ 관계형 그래프
  - ◆ 분포형 그래프
  - ◆ 카테고리형 그래프
- 실제 분석에서는 matplotlib 라이브러리와 seaborn 라이브러리 두 가지를 함께 사용함
- seaborn 라이브러리를 사용할 때 주의할 점은
  seaborn 라이브러리가 matplotlib 라이브러리에 의존적이기 때문에
  matplotlib 라이브러리도 반드시 함께 임포트(Import) 해야 한다는 점임

02. seaborn 라이브러리

❖ 시작하기 전에 (3/4)

- seaborn 라이브러리의 주요 특징은 다음과 같음
  - ◆ 뛰어난 시각화 효과
  - ◆ 간결한 구문 제공
  - ◆ 판다스 데이터프레임에 최적화
  - ◆ 쉬운 데이터프레임 집계 및 차트 요약 --> groupby() + sum(), mean() -> seaborn

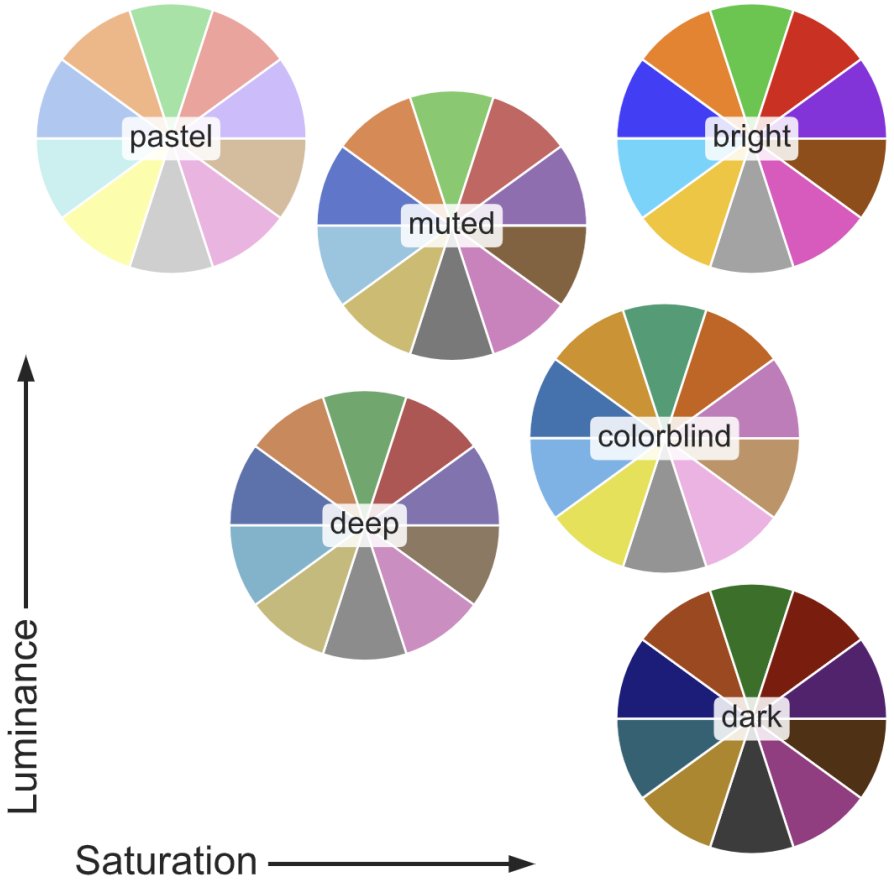
matplotlib VS seaborn

- plot
- hist
- bar
- boxplot

02. seaborn 라이브러리

❖ 시작하기 전에 (4/4)

- seaborn 라이브러리는 기본적으로 matplotlib 라이브러리보다 제공하는 색상이 더 다양하기 때문에 색 표현력이 좋음
- seaborn 라이브러리에서는 6개의 기본 팔레트(Palette)를 제공함
  - bright
  - colorblind
  - dark
  - deep
  - muted
  - pastel



기본 팔레트 외에도 다양한 색상 팔레트가 존재하며, 이는 아래 seaborn 사이트에서 확인할 수 있음

[사진출처] https://seaborn.pydata.org/tutorial/color_palettes.html

02. seaborn 라이브러리

❖ 데이터 시각화 준비하기: ① 라이브러리 및 데이터 읽어오기 (1/2)

- 'health_screenings.xlsx' 파일에는 2020년 건강검진 데이터의 일부가 저장되어 있음
- 건강검진 데이터를 .xlsx 파일로부터 읽고, 컬럼명을 확인해 보자

```
1 import pandas as pd
2
3 df = pd.read_excel("health_screenings.xlsx")
4 df.columns
```

실행결과

```
Index(['year', 'city_code', 'gender', 'age_code', 'height', 'weight', 'waist',
       'eye_left', 'eye_right', 'hear_left', 'hear_right', 'systolic',
       'diastolic', 'blood_sugar', 'cholesterol', 'triglycerides', 'HDL',
       'LDL', 'hemoglobin', 'urine_protein', 'serum', 'AST', 'ALT', 'GTP',
       'smoking', 'drinking', 'oral_check', 'dental_caries', 'tartar',
       'open_date'],
      dtype='object')
```

성별, 흡연 상태, 음주 여부 데이터를 이용해 seaborn 라이브러리 활용법을 살펴보자

02. seaborn 라이브러리

❖ 데이터 시각화 준비하기: ① 라이브러리 및 데이터 읽어오기 (2/2)

- 데이터프레임에서 성별(gender), 흡연 상태(smoking), 음주 여부(drinking) 컬럼에 저장된 값들을 확인하기 위해서, 첫 5행을 출력해 확인해 보자

```
1 df[[“gender”, “smoking”, “drinking”]].head()
```

실행결과

	gender	smoking	drinking
0	1	1	0
1	2	1	0
2	2	1	0
3	1	1	0
4	2	1	0

각 컬럼에 기록된 값들의 의미는 무엇일까?

02. seaborn 라이브러리

❖ 데이터 시각화 준비하기: ② 데이터 전처리 (1/4)

- 성별(gender), 흡연 상태(smoking), 음주 여부(drinking) 컬럼에 기록된 값들의 의미를 살펴 보자
- 각 컬럼에 숫자 자료형으로 저장되어 있는 데이터를, 가독성을 높이기 위해서 문자열로 변경해 보자

Before **After**

No.	Column Name	Meaning	Value	String
1	"gender"	남성	1	"Man"
		여성	2	"Woman"
2	"smoking"	비흡연	1	"Non-smoking"
		흡연 끊음	2	"Non-smoking"
		흡연	3	"Smoking"
3	"drinking"	비음주	0	"Non-drinking"
		음주	1	"Drinking"

02. seaborn 라이브러리

❖ 데이터 시각화 준비하기: ② 데이터 전처리 (2/4)

- 우선, 숫자 형식을 문자 형식으로 변환하자

```
1 df[“gender”] = df[“gender”].astype(str)
2 df[“smoking”] = df[“smoking”].astype(str)
3 df[“drinking”] = df[“drinking”].astype(str)
```

자료형을 int64에서 str로 변환하지 않은 상태에서 진행하면, 아래와 같은 FutureWarning 경고창이 뜰 수 있음

```
FutureWarning: Setting an item of incompatible dtype is deprecated and will
raise an error in a future version of pandas. Value ‘Man’ has dtype
incompatible with int64, please explicitly cast to a compatible dtype first.
  df.loc[df[“gender”]==1, [“gender”]] = “Man”
```

02. seaborn 라이브러리

❖ 데이터 시각화 준비하기: ② 데이터 전처리 (3/4)

- 그다음 부울린 인덱싱(Boolean Indexing)을 활용해, 지정한 문자열로 바꿔보자

```
df.loc[df["gender"]=='1', ["gender"]] = "Man"
df.loc[df["gender"]=='2', ["gender"]] = "Woman"

df.loc[(df["smoking"]=='1') | (df["smoking"]==2), ["smoking"]] = "Non-smoking"
df.loc[df["smoking"]=='3', ["smoking"]] = "Smoking"

df.loc[df["drinking"]=='0', ["drinking"]] = "Non-drinking"
df.loc[df["drinking"]=='1', ["drinking"]] = "Drinking"

df[["gender", "smoking", "drinking"]].head(3)
```

실행결과

	gender	smoking	drinking
0	Man	Non-smoking	Non-drinking
1	Woman	Non-smoking	Non-drinking
2	Woman	Non-smoking	Non-drinking

❖ 데이터 시각화 준비하기: ② 데이터 전처리 (4/4)

- 앞 소스 코드에서 비트 연산자 | 대신에, 논리 연산자 or를 사용하면 안 될까? **안 됨**

```
1 import numpy as np
2
3 print(np.array([True]) | np.array([True]))
4 print(np.array([True, False]) | np.array([True, True]))
5 print(np.array([True]) or np.array([True]))
6 print(np.array([True, False]) or np.array([True, True]))
```

실행결과

```
[ True]
[ True  True]
[ True]
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-4789745ca78e> in <cell line: 6>()
      4 print(np.array([True, False]) | np.array([True, True]))
      5 print(np.array([True]) or np.array([True]))
----> 6 print(np.array([True, False]) or np.array([True, True]))
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any()
or a.all()
```

**논리 연산자 or은 두 개의 피연산자의 조건에 따라서 True or False 값 하나만 반환됨.
지금 상황에서는 행별로 True or False 값을 반환 받아야 하므로 비트 연산자를 사용하는 것이 적절함**

02. seaborn 라이브러리

❖ 막대 그래프: ① 데이터 준비하기 (1/4)

- 우선, 데이터프레임에서 성별(gender), 흡연 상태(smoking)의 그룹별 개수(인원)를 구해보자

```
1 smoking = df.groupby(["gender", "smoking"])["smoking"].count()
2 smoking
```

실행결과

```
gender  smoking
Man     Non-smoking    321
        Smoking        161
Woman   Non-smoking    500
        Smoking         18
Name: smoking, dtype: int64
```

02. seaborn 라이브러리

❖ 막대 그래프: ① 데이터 준비하기 (2/4)

- 그룹별 개수(인원)의 시리즈(Series) 자료형을 to_frame( ) 메소드를 이용해, 데이터프레임으로 변환하자

```
1 smoking = smoking.to_frame(name="count")
2 smoking
```

실행결과

		count
gender	smoking	
Man	Non-smoking	321
	Smoking	161
Woman	Non-smoking	500
	Smoking	18

❖ 막대 그래프: ① 데이터 준비하기 (3/4)

- 그룹화된 인덱스를 reset_index( ) 함수를 이용해, 초기화하자

```
1 smoking = smoking.reset_index()
2 smoking
```

실행결과

	gender	smoking	count
0	Man	Non-smoking	321
1	Man	Smoking	161
2	Woman	Non-smoking	500
3	Woman	Smoking	18

❖ 막대 그래프: ① 데이터 준비하기 (4/4)

- 동일한 과정을 데이터프레임에서 성별(gender), 음주 여부(drinking)에 대해 수행해 보자

```
1 drinking = df.groupby(["gender", "drinking"])["drinking"].count()
2 drinking = drinking.to_frame(name="count")
3 drinking = drinking.reset_index()
4 drinking
```

실행결과

	gender	drinking	count
0	Man	Drinking	356
1	Man	Non-drinking	126
2	Woman	Drinking	213
3	Woman	Non-drinking	305

흡연 상태 및 음주 여부 데이터가 모두 준비됨!

02. seaborn 라이브러리

❖ 막대 그래프: ② matplotlib vs. seaborn 비교 (1/5)

- seaborn 라이브러리와 비교를 위해서, 우선 matplotlib 라이브러리를 활용해 성별에 따른 음주 여부 및 흡연 상태 막대 그래프를 그려 보자

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 fig = plt.figure(figsize=(17, 6))
5 fig.suptitle("matplotlib Bar Graph", fontweight="bold")
6 index = np.arange(0, 4, 1)
```

✓ 기본값은 "normal"임
✓ "bold"로 지정하면, 제목을 굵게 표시 가능함

add_subplot( ) 함수를 활용해 성별에 따른 흡연 상태, 음주 여부를 막대 그래프로 시각화 해보자

add_subplot( ) 함수

✓ add_subplot( ) 함수의 인자를 통해, 서브플롯 개수를 조정함
✓ add_subplot(1, 2, 1)은 1행 2열의 서브플롯을 생성한다는 의미임
✓ 세 번째 인자 1인 생성된 두 개의 서브플롯 중 첫 번째 서브플롯을 의미함
✓ 마찬가지로 add_subplot(1, 2, 2)는 1행 2열의 서브플롯에서 두 번째 서브플롯을 의미함

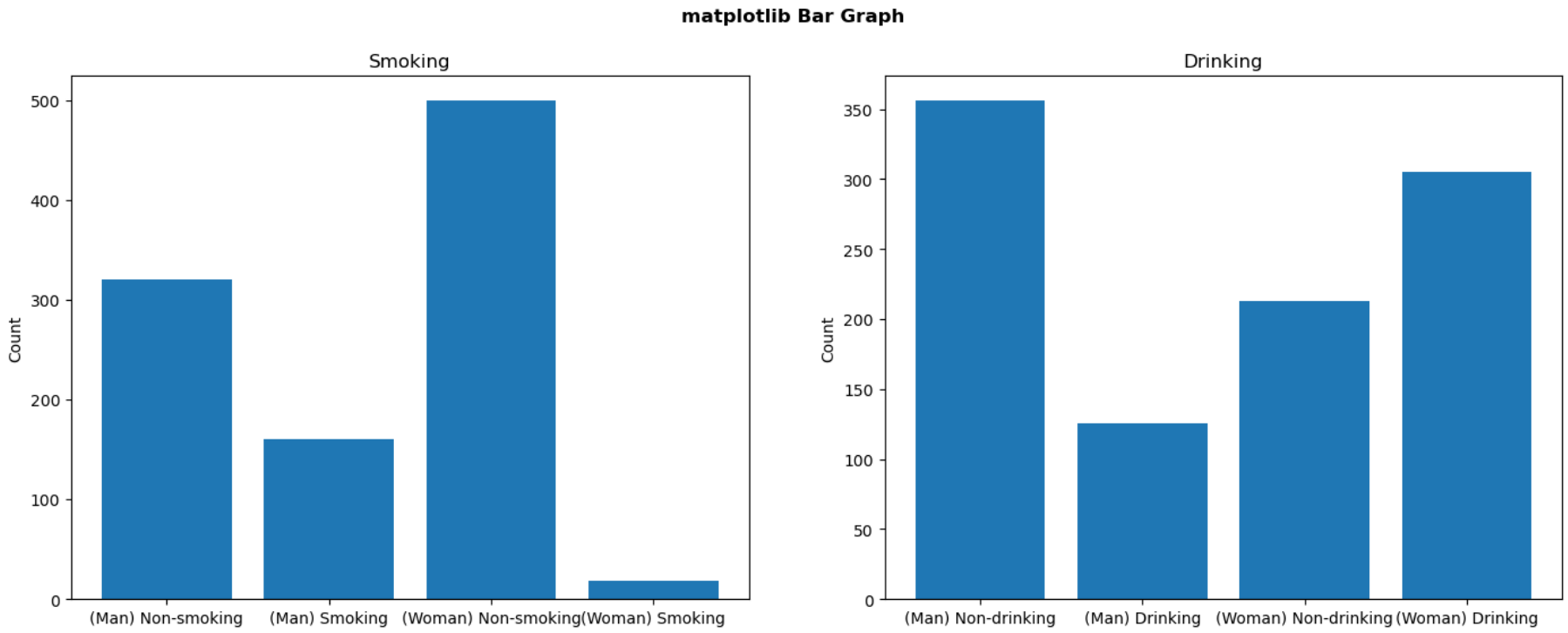
❖ 막대 그래프: ② matplotlib vs. seaborn 비교 (2/5)

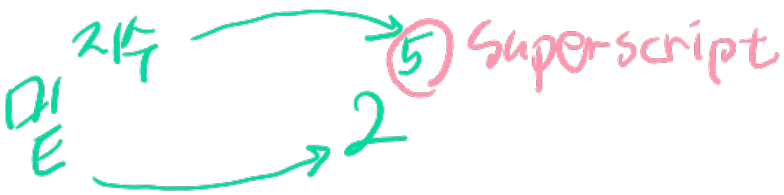
```
7  fig.add_subplot(1, 2, 1)
8
9  plt.bar(index, smoking["count"])
10 plt.title("Smoking")
11 plt.ylabel("Count")
12 plt.xticks(index, ["(Man) Non-smoking", "(Man) Smoking",
13                    "(Woman) Non-smoking", "(Woman) Smoking"])
14
15 fig.add_subplot(1, 2, 2)
16
17 plt.bar(index, drinking["count"])
18 plt.title("Drinking")
19 plt.ylabel("Count")
20 plt.xticks(index, ["(Man) Non-drinking", "(Man) Drinking",
21                    "(Woman) Non-drinking", "(Woman) Drinking"])
22
23 plt.show()
```

- ✓ 첫 번째 인자에는 변경시키고자 하는 xtick의 위치를 배열 형태로 전달함
- ✓ 두 번째 인자에는 앞서 전달된 xtick 위치의 대응되게, 변경하고자 하는 레이블(Labels)을 전달함

❖ 막대 그래프: ② matplotlib vs. seaborn 비교 (3/5)

실행결과





02. seaborn 라이브러리

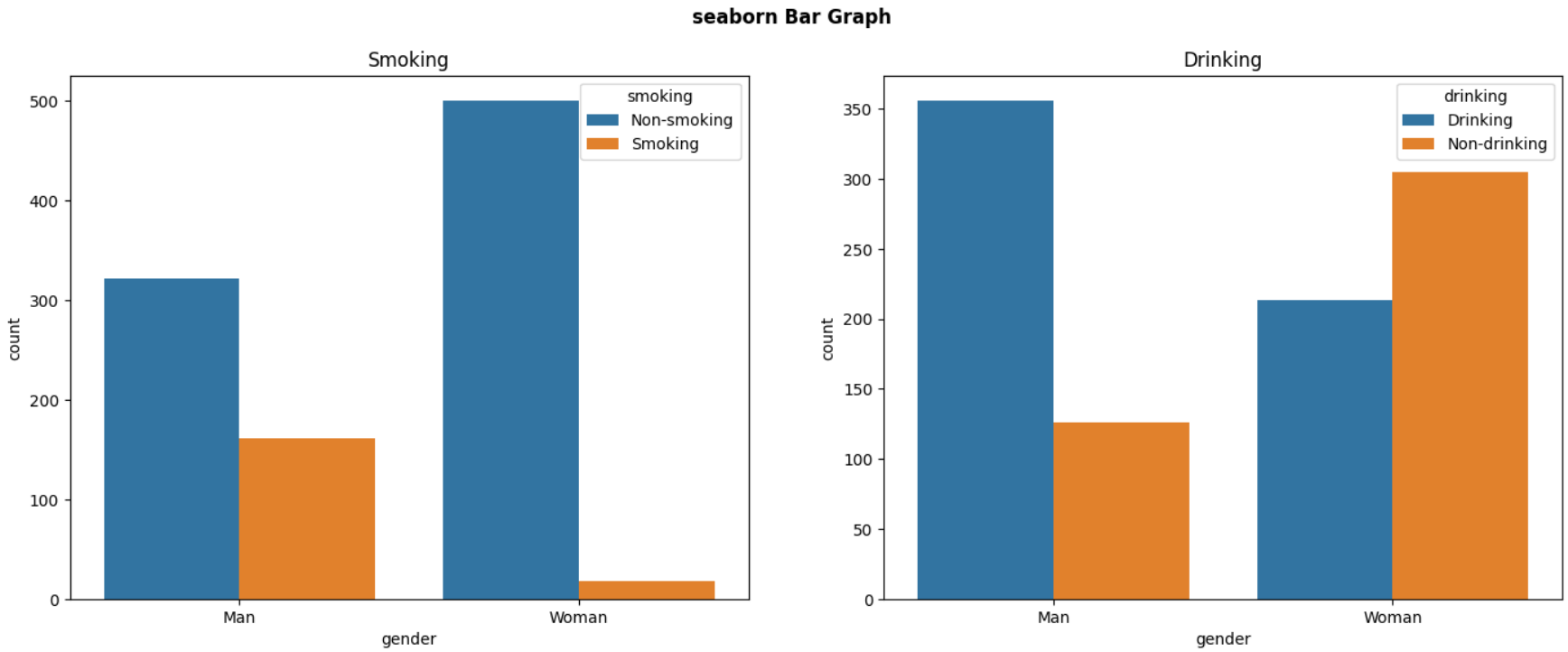
❖ 막대 그래프: ② matplotlib vs. seaborn 비교 (4/5)

- 이번에는 seaborn 라이브러리를 활용해 성별에 따른 음주 여부 및 흡연 상태 막대 그래프를 그려 보자

```
1  import seaborn as sns
2
3  fig = plt.figure(figsize=(17, 6))
4  fig.suptitle("seaborn Bar Graph", fontweight="bold")
5
6  area1 = fig.add_subplot(1, 2, 1)
7  area2 = fig.add_subplot(1, 2, 2)
8
9  sns.barplot(data=smoking, x="gender", y="count", hue="smoking", ax=area1)
10 sns.barplot(data=drinking, x="gender", y="count", hue="drinking", ax=area2)
11
12 area1.set_title("Smoking")
13 area2.set_title("Drinking")
14
15 plt.show()
```

❖ 막대 그래프: ② matplotlib vs. seaborn 비교 (5/5)

실행결과



02. seaborn 라이브러리

❖ 산점도: ① 데이터 준비하기

- 데이터프레임에서 남성 및 여성의 성별(gender), 몸무게(weight), 허리둘레(waist) 데이터를 읽어와 m_data, w_data에 저장하자

```
1 m_data = df.loc[df[“gender”]==“Man”, [“gender”, “weight”, “waist”]]
2 w_data = df.loc[df[“gender”]==“Woman”, [“gender”, “weight”, “waist”]]
```

02. seaborn 라이브러리

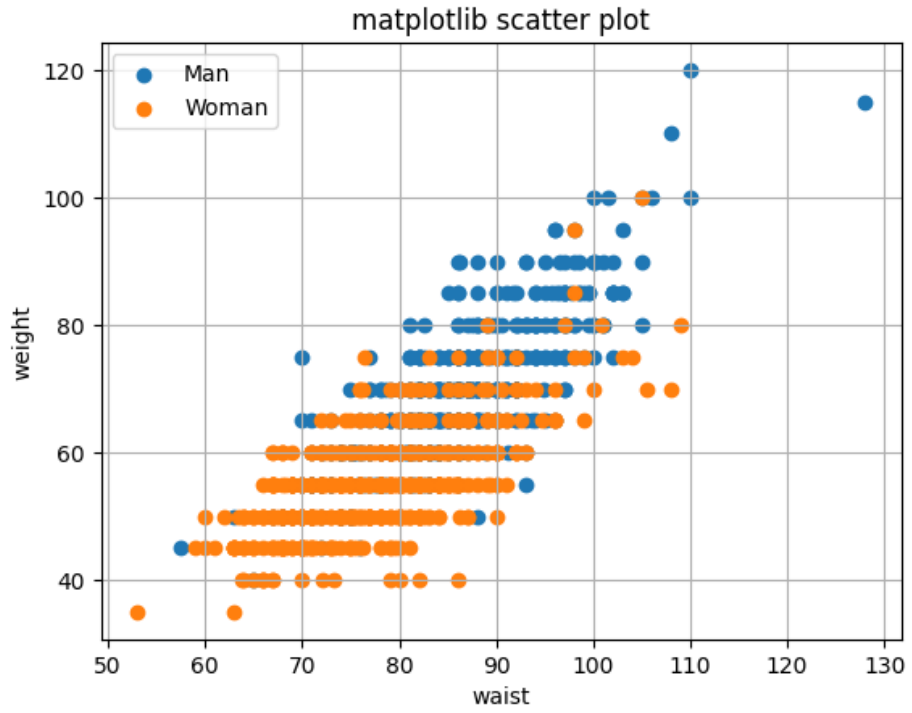
❖ 산점도: ② matplotlib vs. seaborn 비교 (1/4)

- seaborn 라이브러리와 비교를 위해서, 우선 matplotlib 라이브러리를 활용해 성별에 따라 x축을 허리둘레(waist), y축을 몸무게로 하는 산점도를 그려 보자

```
1 plt.figure()
2 plt.title("matplotlib scatter plot")
3 plt.scatter(m_data["waist"], m_data["weight"], label="Man")
4 plt.scatter(w_data["waist"], w_data["weight"], label="Woman")
5 plt.xlabel("waist")
6 plt.ylabel("weight")
7 plt.legend()
8 plt.grid()
9 plt.show()
```

❖ 산점도: ② matplotlib vs. seaborn 비교 (2/4)

실행결과



02. seaborn 라이브러리

❖ 산점도: ② matplotlib vs. seaborn 비교 (3/4)

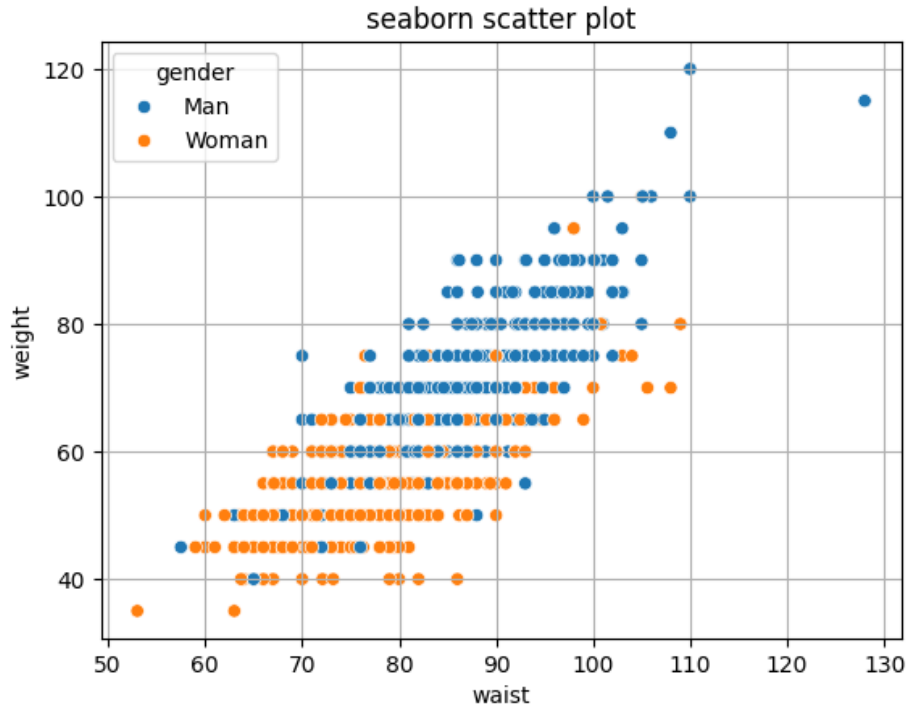
- 이번에는 seaborn 라이브러리를 활용해 산점도를 그려 보자

```
1 plt.figure()
2 plt.title("seaborn scatter plot")
3 sns.scatterplot(data=df, x="waist", y="weight", hue="gender")
4 plt.grid()
5 plt.show()
```

02. seaborn 라이브러리

❖ 산점도: ② matplotlib vs. seaborn 비교 (4/4)

실행결과



02. seaborn 라이브러리

❖ 히스토그램: ① matplotlib 히스토그램 그리기 (1/2)

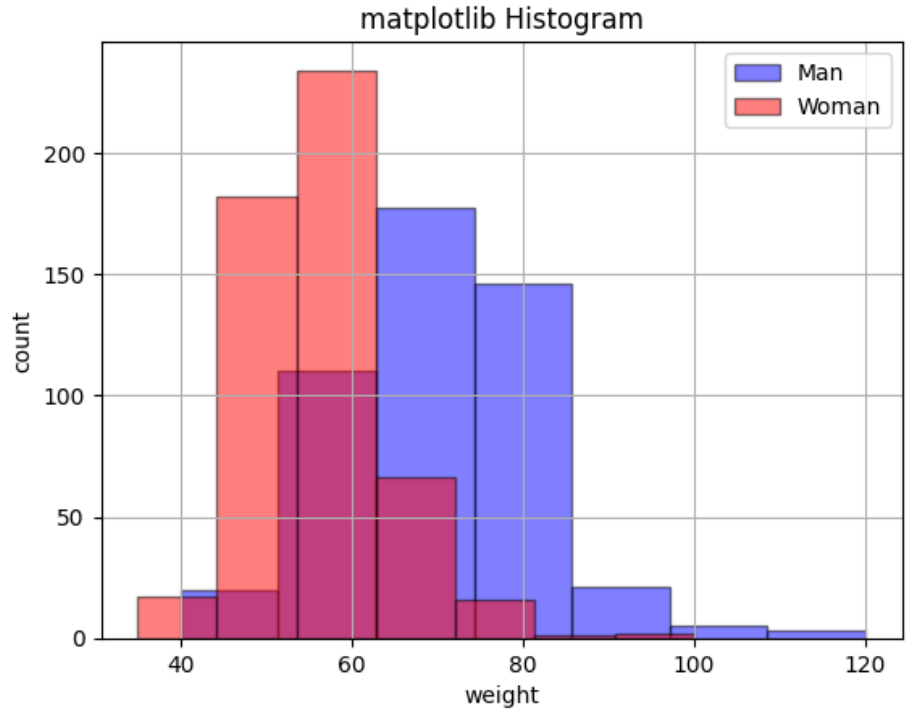
- matplotlib 라이브러리를 활용해 남성과 여성의 몸무게(weight) 히스토그램을 그려 보자

```
1 plt.figure()
2 plt.title("matplotlib Histogram")
3 plt.hist(m_data["weight"], bins=7, alpha=0.5, label="Man", color='b', edgecolor='k')
4 plt.hist(w_data["weight"], bins=7, alpha=0.5, label="Woman", color='r', edgecolor='k')
5 plt.xlabel("weight")
6 plt.ylabel("count")
7 plt.legend()
8 plt.grid()
9 plt.show()
```

- ✓ alpha 속성값의 범위는 0부터 1까지임
- ✓ 0에 가까울 수록 투명해짐
- ✓ 1에 가까울 수록 불투명해짐
- ✓ edgecolor에는 막대기 테두리의 색상을 지정함

❖ 히스토그램: ① matplotlib 히스토그램 그리기 (2/2)

실행결과



02. seaborn 라이브러리

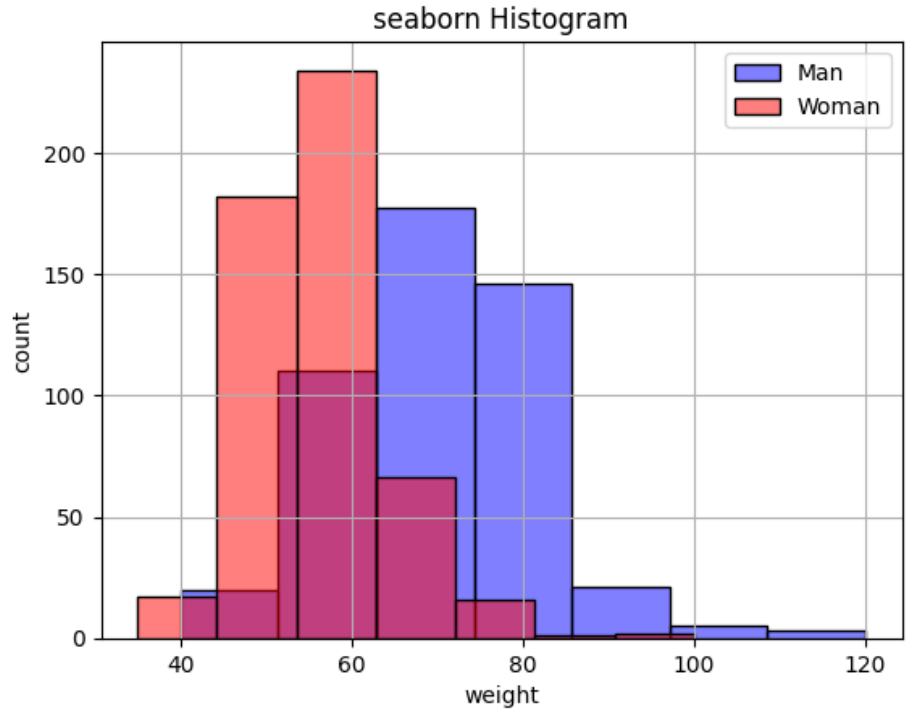
❖ 히스토그램: ② seaborn 히스토그램 그리기 (1/2)

- 이번엔 seaborn 라이브러리를 활용해 남성과 여성의 몸무게(weight) 히스토그램을 그려 보자

```
plt.figure()
plt.title("seaborn Histogram")
sns.histplot(m_data["weight"], bins=7, alpha=0.5, label="Man", color='b')
sns.histplot(w_data["weight"], bins=7, alpha=0.5, label="Woman", color='r')
plt.xlabel("weight")
plt.ylabel("count")
plt.legend()
plt.grid()
plt.show()
```

❖ 히스토그램: ② seaborn 히스토그램 그리기 (2/2)

실행결과



02. seaborn 라이브러리

❖ 히스토그램: ③ seaborn 히스토그램의 커널 밀도 추정 속성 지정하기 (1/2)

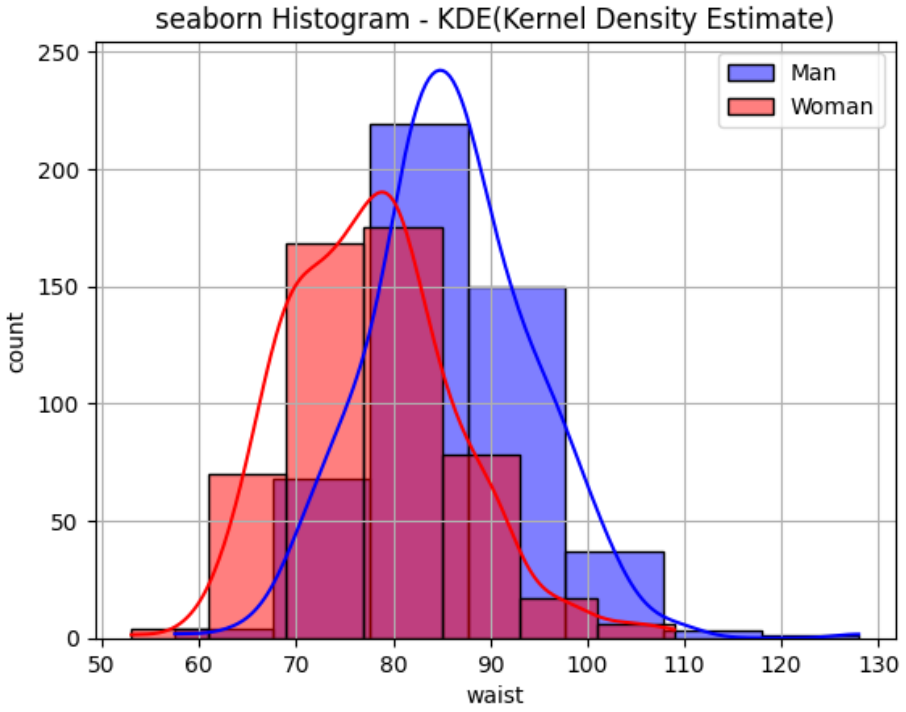
- seaborn 라이브러리의 histplot( ) 함수는 데이터 분포와 밀도를 시각화할 수 있음
- kde 매개변수를 True로 지정하여, 커널 밀도 추정(Kernel Density Estimate) 곡선을 추가해 보자

```
1 plt.figure()
2 plt.title("seaborn Histogram - KDE(Kernel Density Estimator)")
3 sns.histplot(m_data["waist"], bins=7, alpha=0.5, label="Man", color='b', kde=True)
4 sns.histplot(w_data["waist"], bins=7, alpha=0.5, label="Woman", color='r', kde=True)
5 plt.xlabel("weight")
6 plt.ylabel("count")
7 plt.legend()
8 plt.grid()
9 plt.show()
```

kde를 True로 지정하면, 히스토그램보다 부드러운 형태의 분포 곡선을 추가해 보여줌

❖ 히스토그램: ③ seaborn 히스토그램의 커널 밀도 추정 속성 지정하기 (2/2)

실행결과



02. seaborn 라이브러리

❖ 상자 그림: ① 데이터 준비하기

- 음주 여부(drinking) 및 성별(gender)를 기준으로 몸무게(weight) 데이터를 추출해 보자

```
m_non_drink = df.loc[(df[“drinking”]==“Non-drinking”) & (df[“gender”]==“Man”), [“weight”]]
w_non_drink = df.loc[(df[“drinking”]==“Non-drinking”) & (df[“gender”]==“Woman”), [“weight”]]

m_drink = df.loc[(df[“drinking”]==“Drinking”) & (df[“gender”]==“Man”), [“weight”]]
w_drink = df.loc[(df[“drinking”]==“Drinking”) & (df[“gender”]==“Woman”), [“weight”]]
```

02. seaborn 라이브러리

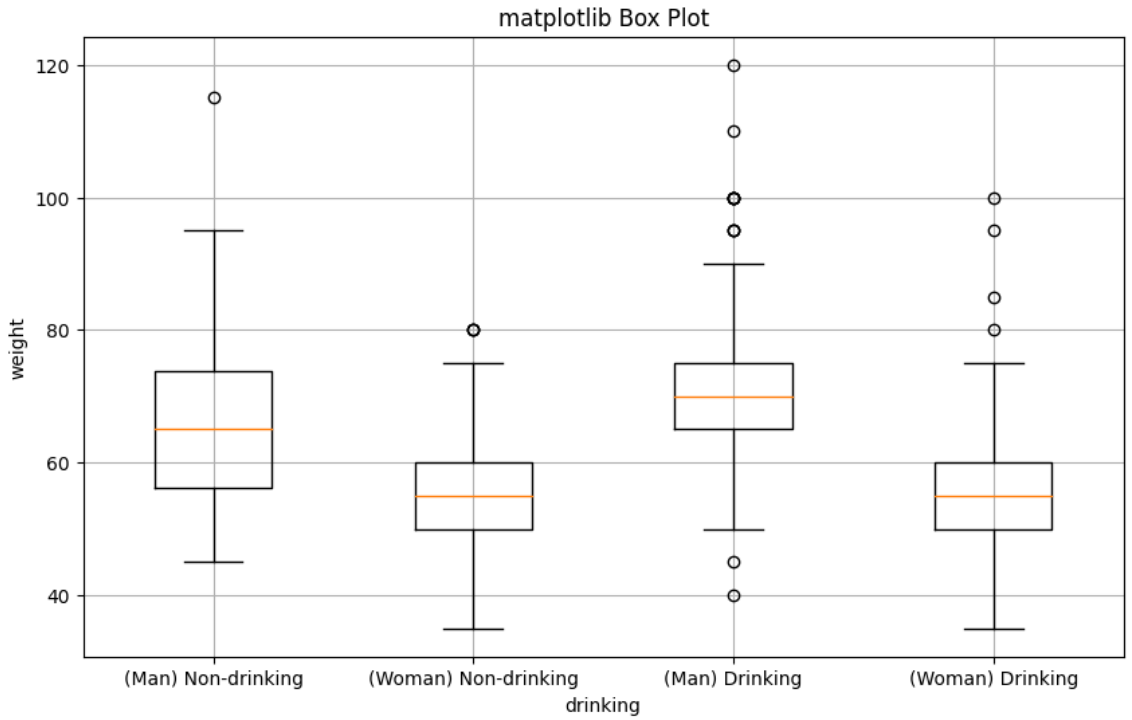
❖ 상자 그림: ② matplotlib 상자 그림 그리기 (1/2)

- matplotlib 라이브러리를 활용해 남성과 여성의 몸무게(weight)를 상자 그림으로 시각화 해보자

```
plt.figure(figsize=(10, 6))
plt.title("matplotlib Box Plot")
plt.boxplot([m_non_drink, w_non_drink, m_drink, w_drink],
            tick_labels=["(Man) Non-drinking", "(Woman) Non-drinking",
                    "(Man) Drinking", "(Woman) Drinking"])
plt.xlabel("drinking")
plt.ylabel("weight")
plt.grid()
plt.show()
```

❖ 상자 그림: ② matplotlib 상자 그림 그리기 (2/2)

실행결과



02. seaborn 라이브러리

❖ 상자 그림: ③ matplotlib 가로 상자 그림 그리기 (1/2)

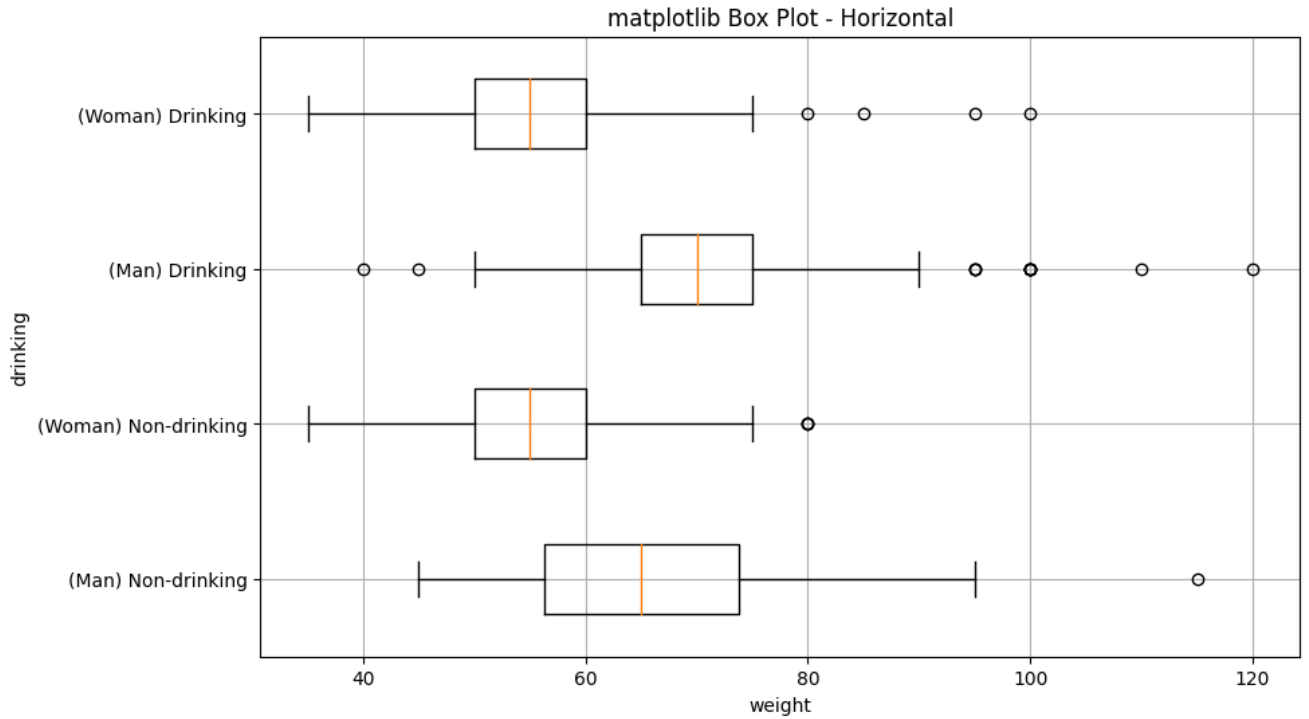
- vert 매개변수에 False를 지정해, 가로 상자 그림을 그려보자

```
1  plt.figure(figsize=(10, 6))
2  plt.title("matplotlib Box Plot - Horizontal")
3  plt.boxplot([m_non_drink, w_non_drink, m_drink, w_drink],
4              tick_labels=["(Man) Non-drinking", "(Woman) Non-drinking",
5                           "(Man) Drinking", "(Woman) Drinking"],
6              vert=False)
7  plt.xlabel("weight")
8  plt.ylabel("drinking")
9  plt.grid()
10 plt.show()
```

- ✓ 기본값은 True임
- ✓ True로 설정하면, 가로축에 수직인 박스(Vertical Boxes)로 그림
- ✓ False로 설정하면, 가로축과 수평인 박스(Horizontal Boxes)로 그림

❖ 상자 그림: ③ matplotlib 가로 상자 그림 그리기 (2/2)

실행결과



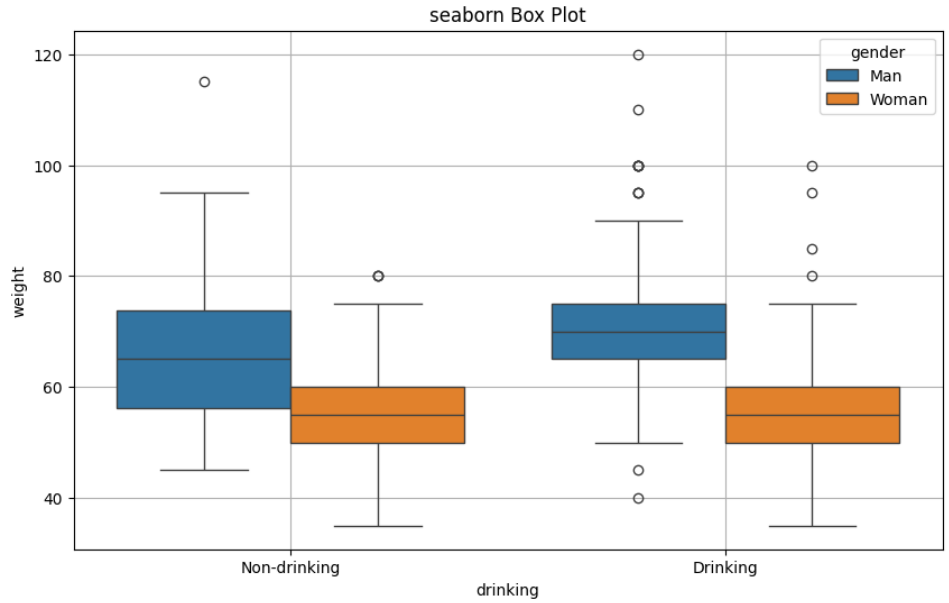
02. seaborn 라이브러리

❖ 상자 그림: ④ seaborn 상자 그림 그리기

- 이번에는 seaborn 라이브러리를 활용해 남성과 여성의 몸무게(weight)를 상자 그림으로 시각화 해보자

```
1 plt.figure(figsize=(10, 6))
2 plt.title("seaborn Box Plot")
3 sns.boxplot(data=df, x="drinking", y="weight", hue="gender")
4 plt.grid()
5 plt.show()
```

실행결과



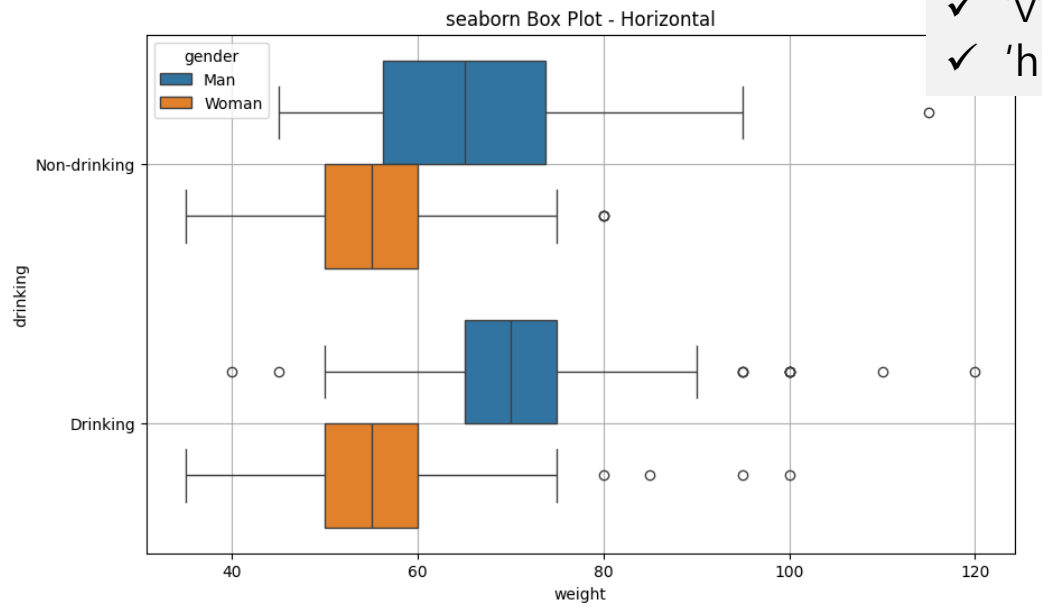
02. seaborn 라이브러리

❖ 상자 그림: ⑤ seaborn 가로 상자 그림 그리기

- orient 매개변수에 'h'를 지정해, 가로 상자 그림을 그려보자

```
1 plt.figure(figsize=(10, 6))
2 plt.title("seaborn Box Plot")
3 sns.boxplot(data=df, x="weight", y="drinking", hue="gender", orient='h')
4 plt.grid()
5 plt.show()
```

실행결과



- ✓ 기본값은 'v'임
- ✓ 'v'는 수직(Vertical)을 의미하며, 가로축과 수직인 박스로 그림
- ✓ 'h'는 수평(Horizontal)을 의미하며, 가로축과 수평인 박스로 그림

02. seaborn 라이브러리

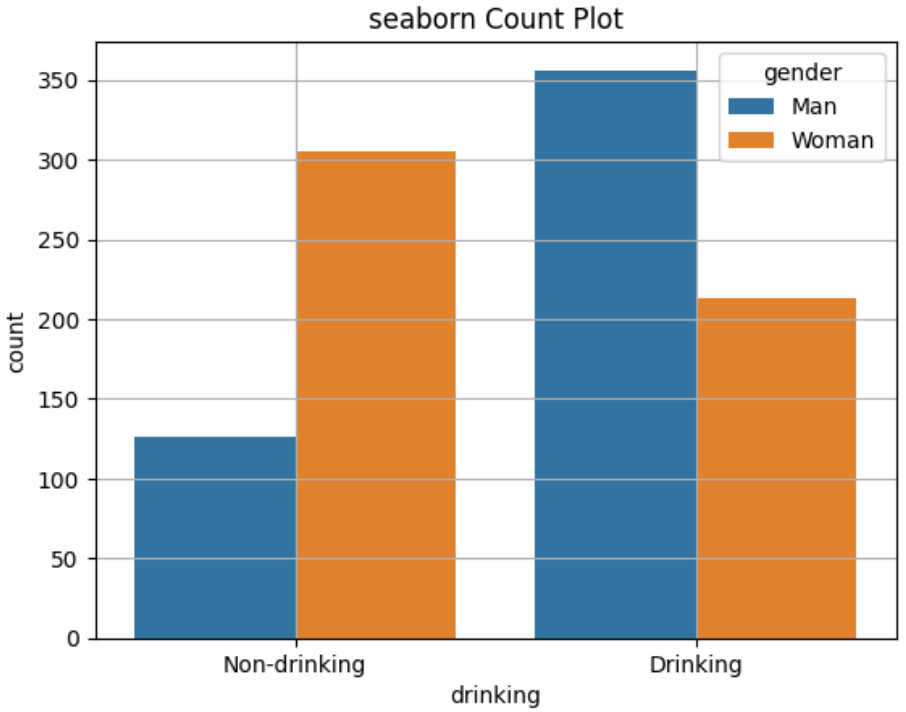
❖ 카운터 플롯: ① seaborn 카운터 플롯 그리기 (1/2)

- 카운트 플롯(Count Plot)은 범주형 데이터에 대하여 항목별 개수를 세어 막대 그래프로 표현함
- 카운트 플롯을 이용하여 각 범주별로 데이터가 얼마나 있는지 표현할 수 있으며,
  해당 열을 구성하고 있는 값을 범주별로 구분하여 보여 줌

```
1 plt.figure()
2 plt.title("seaborn Count Plot")
3 sns.countplot(data=df, x="drinking", hue="gender")
4 plt.grid()
5 plt.show()
```

❖ 카운터 플롯: ① seaborn 카운터 플롯 그리기 (2/2)

실행결과



02. seaborn 라이브러리

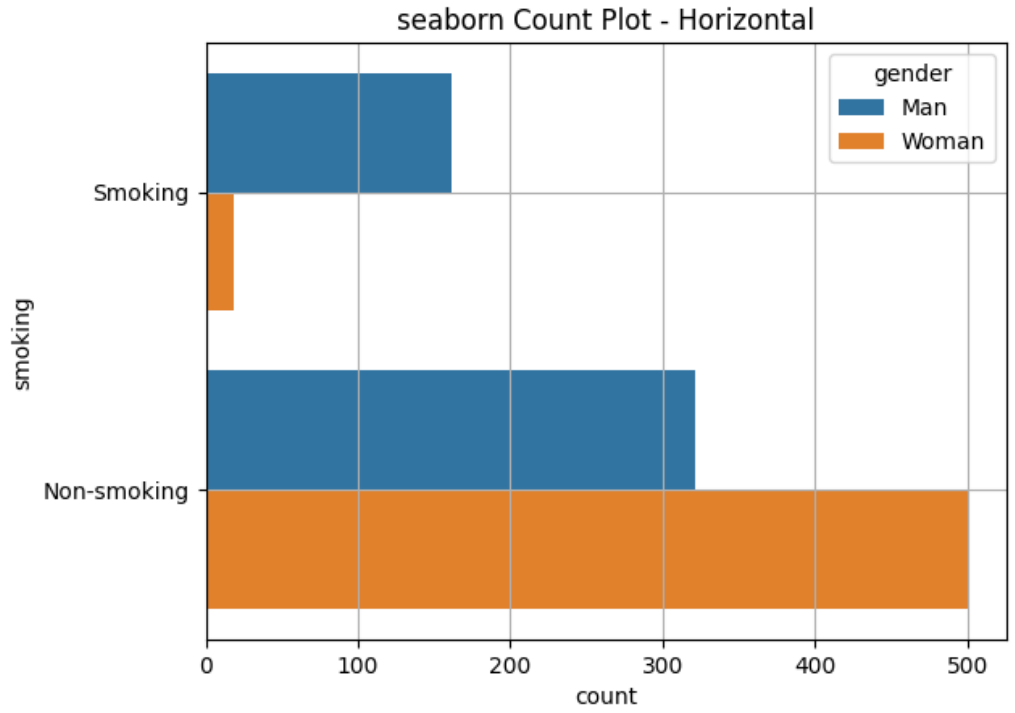
❖ 카운터 플롯: ② seaborn 가로 카운터 플롯 그리기 (1/2)

- x축 대신 y축으로 가로 막대를 지정하고, order 매개변수를 통해 막대 출력 순서를 지정해 보자

```
1 plt.figure()
2 plt.title("seaborn Count Plot - Horizontal")
3 sns.countplot(data=df, y="smoking", hue="gender", order=["Smoking", "Non-smoking"])
4 plt.grid()
5 plt.show()
```

❖ 카운터 플롯: ② seaborn 가로 카운터 플롯 그리기 (2/2)

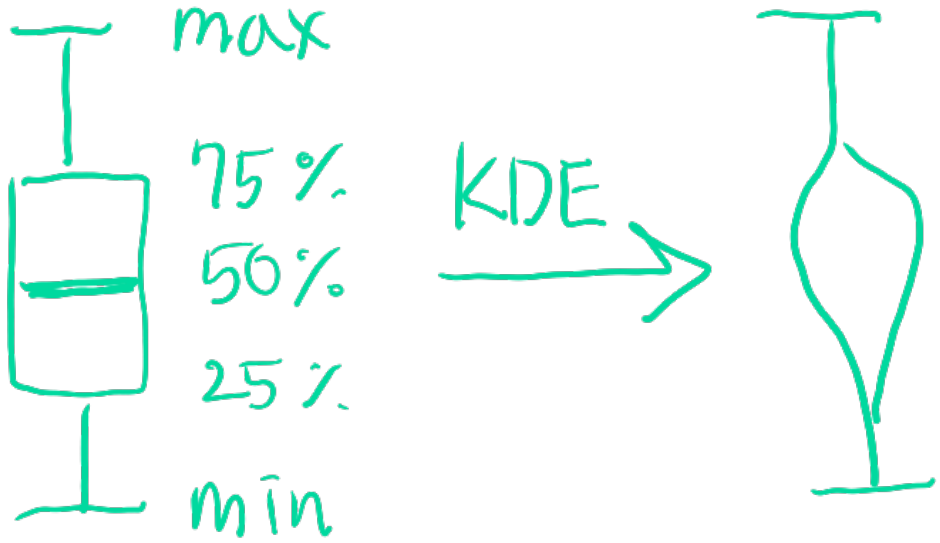
실행결과



02. seaborn 라이브러리

❖ 바이올린 플롯 <-- 상자 그림(Box Plot)

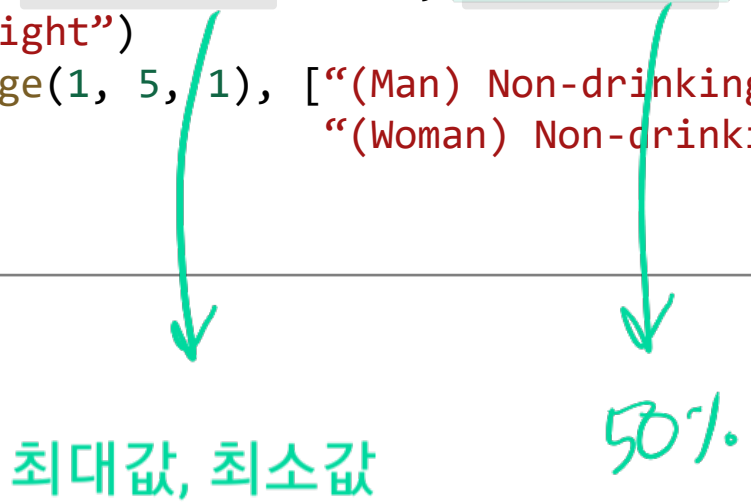
- 바이올린 플롯(Violin Plot)은 바이올린처럼 생겼다고 붙여진 이름임
- 상자 그림(Box Plot)과 달리, 커널 밀도 추정치를 이용하여 그래프를 그림
- 분포에 대한 결과를 보일 때 효과적이지만, 작은 샘플 크기로 그래프를 보여 주는 경우, 분석에 오해의 소지가 있을 수 있음



❖ 바이올린 플롯: ① matplotlib 바이올린 플롯 그리기 (1/2)

- matplotlib 라이브러리를 활용해 음주 여부(drinking), 성별(gender)을 기준으로 몸무게(weight)를 바이올린 플롯으로 시각화 해보자

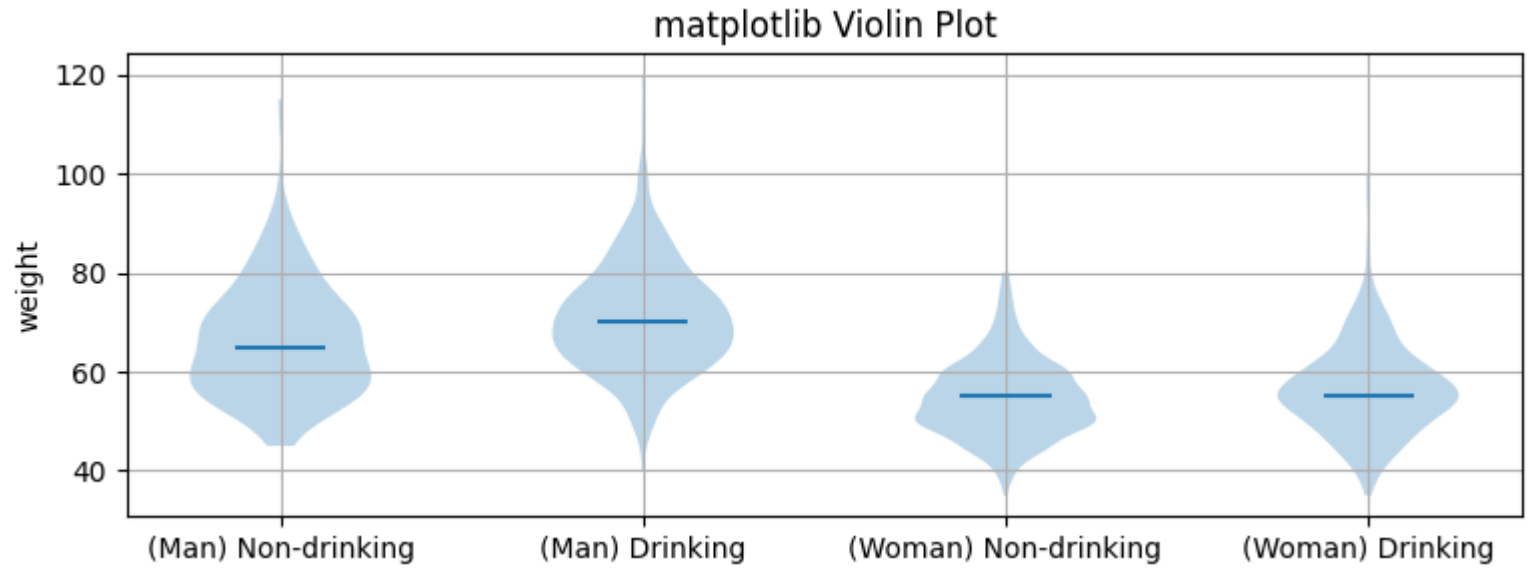
```
1  plt.figure(figsize=(9, 3))
2  plt.title("matplotlib Violin Plot")
3  plt.violinplot([m_non_drink, m_drink, w_non_drink, w_drink],
4                 showextrema=False, showmedians=True)
5  plt.ylabel("weight")
6  plt.xticks(range(1, 5, 1), ["(Man) Non-drinking", "(Man) Drinking",
7                              "(Woman) Non-drinking", "(Woman) Drinking"])
8  plt.grid()
9  plt.show()
```



- extremum
  (명사) 극값
- extrema
  (명사) extremum의 복수형
- median
  (명사) 중앙값, 가운뎃값

❖ 바이올린 플롯: ① matplotlib 바이올린 플롯 그리기 (2/2)

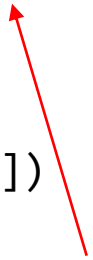
실행결과



❖ 바이올린 플롯: ② matplotlib 가로 바이올린 플롯 그리기 (1/2)

- vert 매개변수에 False를 지정해, 가로 상자 그림을 그려보자

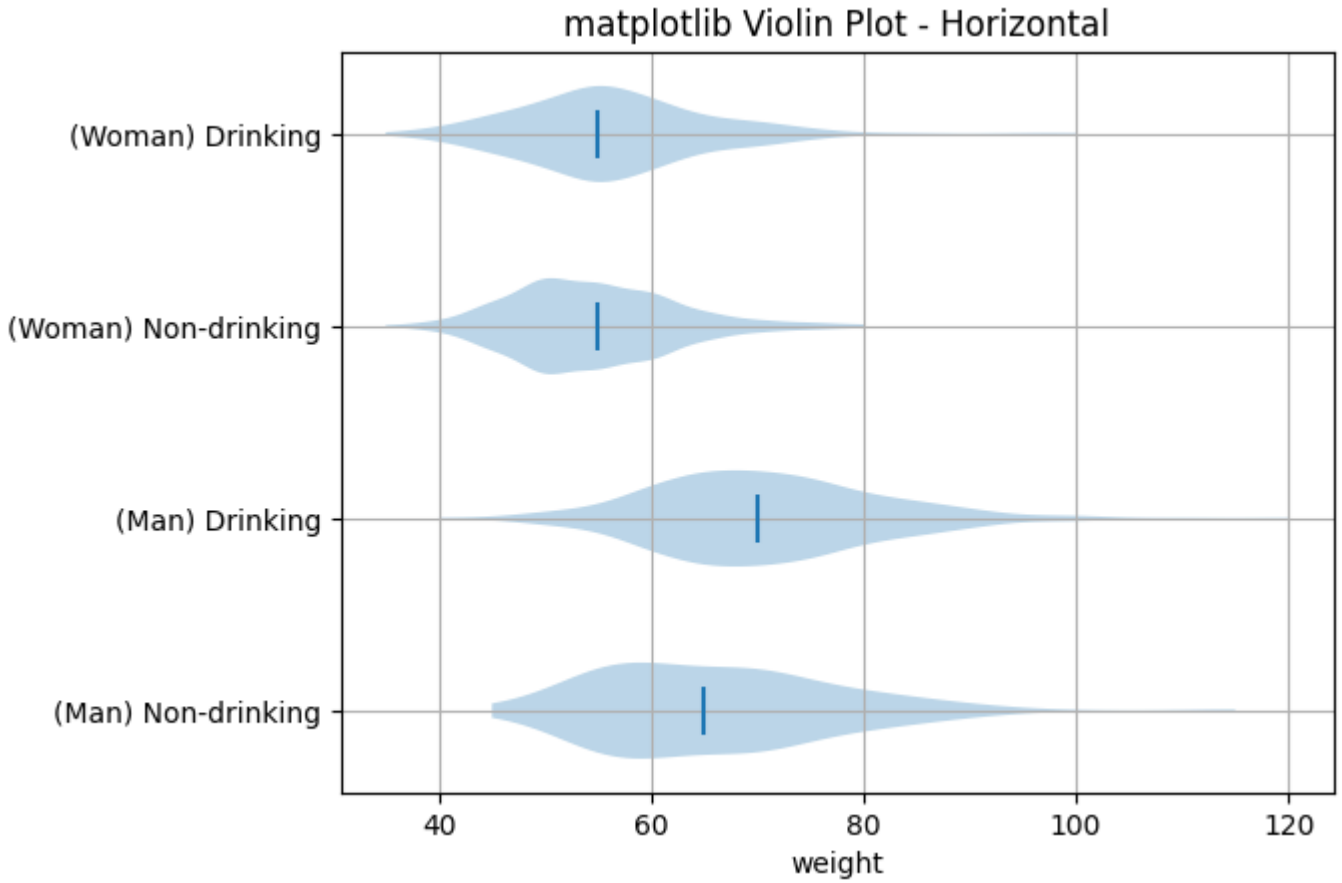
```
1 plt.figure()
2 plt.title("matplotlib Violin Plot - Horizontal")
3 plt.violinplot([m_non_drink, m_drink, w_non_drink, w_drink], vert=False,
4                showextrema=False, showmedians=True)
5 plt.xlabel("weight")
6 plt.yticks(range(1, 5, 1), ["(Man) Non-drinking", "(Man) Drinking",
7                             "(Woman) Non-drinking", "(Woman) Drinking"])
8 plt.grid()
9 plt.show()
```



✓ 기본값은 True임
✓ True로 설정하면, 가로축에 수직인 박스(Vertical Boxes)로 그림
✓ False로 설정하면, 가로축과 수평인 박스(Horizontal Boxes)로 그림

❖ 바이올린 플롯: ② matplotlib 가로 바이올린 플롯 그리기 (2/2)

실행결과

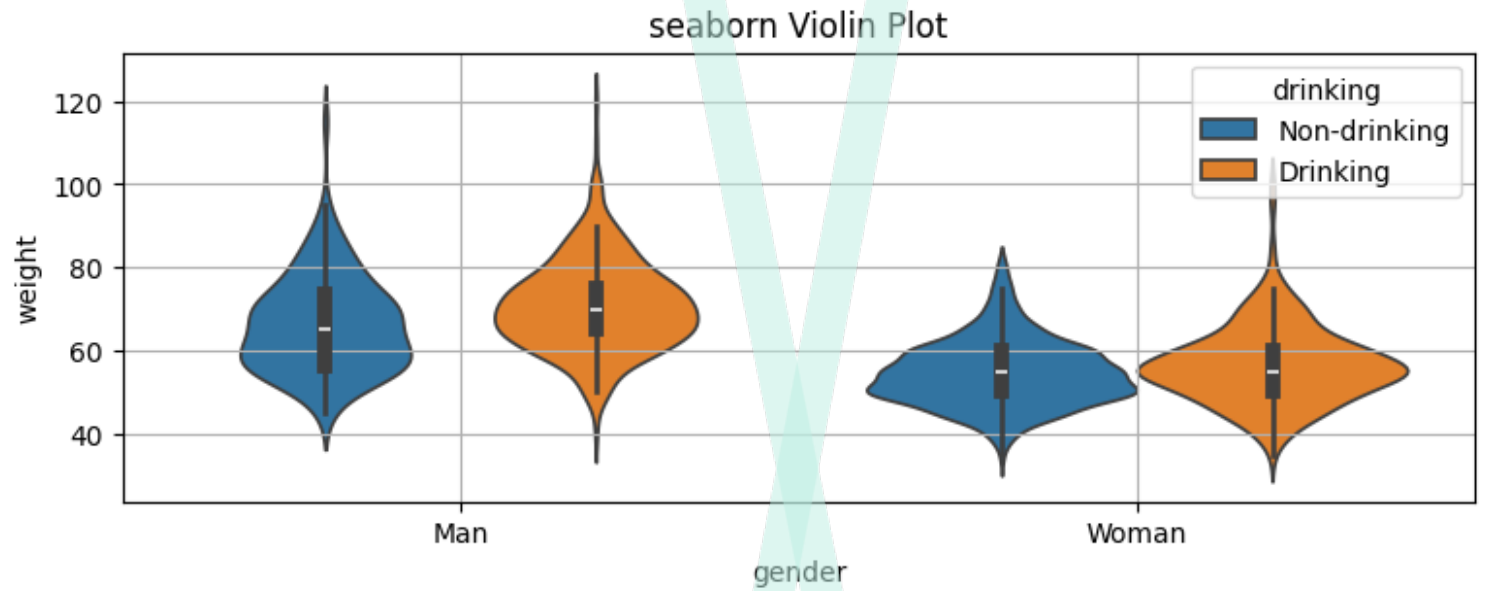


❖ 바이올린 플롯: ③ seaborn 바이올린 플롯 그리기

- 이번에는 seaborn 라이브러리를 활용해 바이올린 플롯을 그려 보자

```
1 plt.figure(figsize=(9, 3))
2 plt.title("seaborn Violin Plot")
3 sns.violinplot(data=df, x="gender", y="weight", hue="drinking")
4 plt.grid()
5 plt.show()
```

실행결과

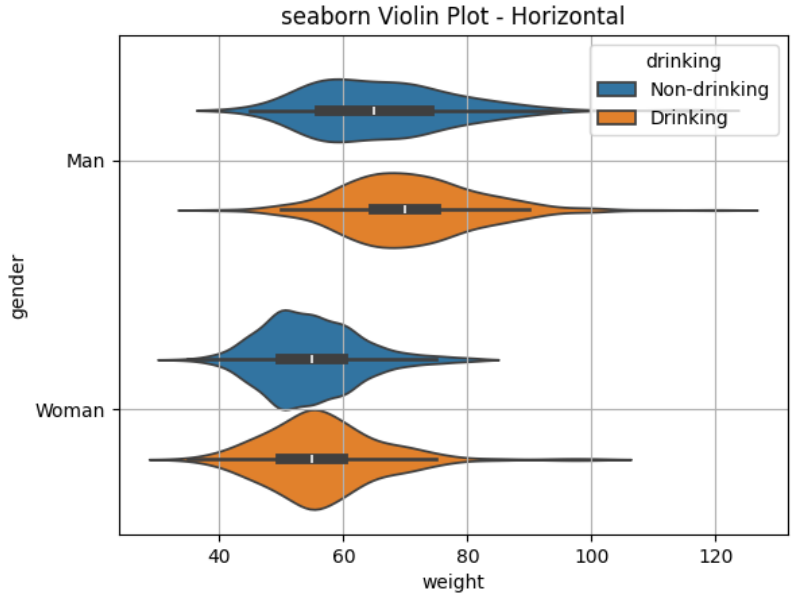


❖ 바이올린 플롯: ④ seaborn 가로 바이올린 플롯 그리기

- x축 대신 y축으로 가로 형태를 지정해, 가로 바이올린 플롯을 그려 보자

```
1 plt.figure()
2 plt.title("seaborn Violin Plot - Horizontal")
3 sns.violinplot(data=df, x="weight", y="gender", hue="drinking")
4 plt.grid()
5 plt.show()
```

실행결과



02. seaborn 라이브러리

❖ 히트맵: ① 데이터 준비하기

- 데이터프레임에서 성별(gender), 몸무게(weight), 허리둘레(waist) 데이터를 읽어와 df3에 저장하자
- 또한, 성별(gender), 몸무게(weight), 허리둘레(waist), 콜레스테롤(cholesterol), HDL, LDL, 흡연 상태(smoking), 음주 여부(drinking) 데이터를 읽어와 df8에 저장하자

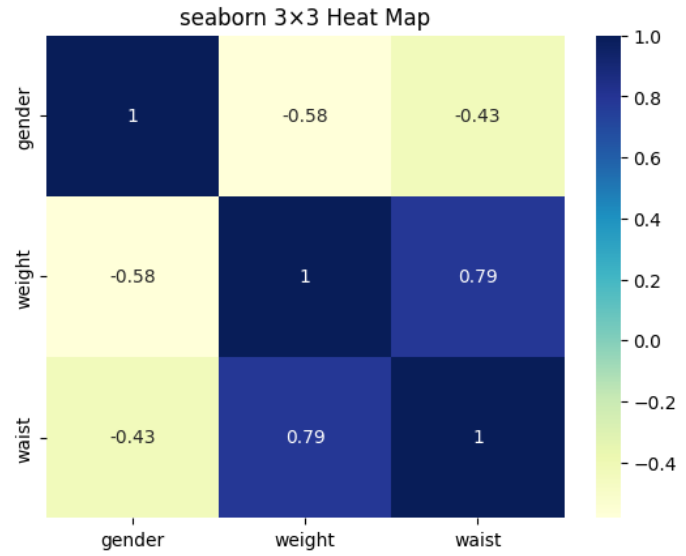
```
1 df = pd.read_excel("health_screenings.xlsx")
2
3 df3 = df.loc[:, ["gender", "weight", "waist"]]
4 df8 = df.loc[:, ["gender", "weight", "waist", "cholesterol", "HDL", "LDL",
5                  "smoking", "drinking"]]
```

❖ 히트맵: ② seaborn 3×3 히트맵 그리기 (1/2)

- df3 데이터프레임에서 corr( ) 함수를 이용해서 성별, 몸무게, 허리둘레 사이의 상관계수를 구하고, 이를 히트맵으로 시각화 해보자

```
1 plt.figure()
2 plt.title("seaborn 3×3 Heat Map")
3 sns.heatmap(df3.corr(), annot=True, cmap="YlGnBu")
4 plt.show()
```

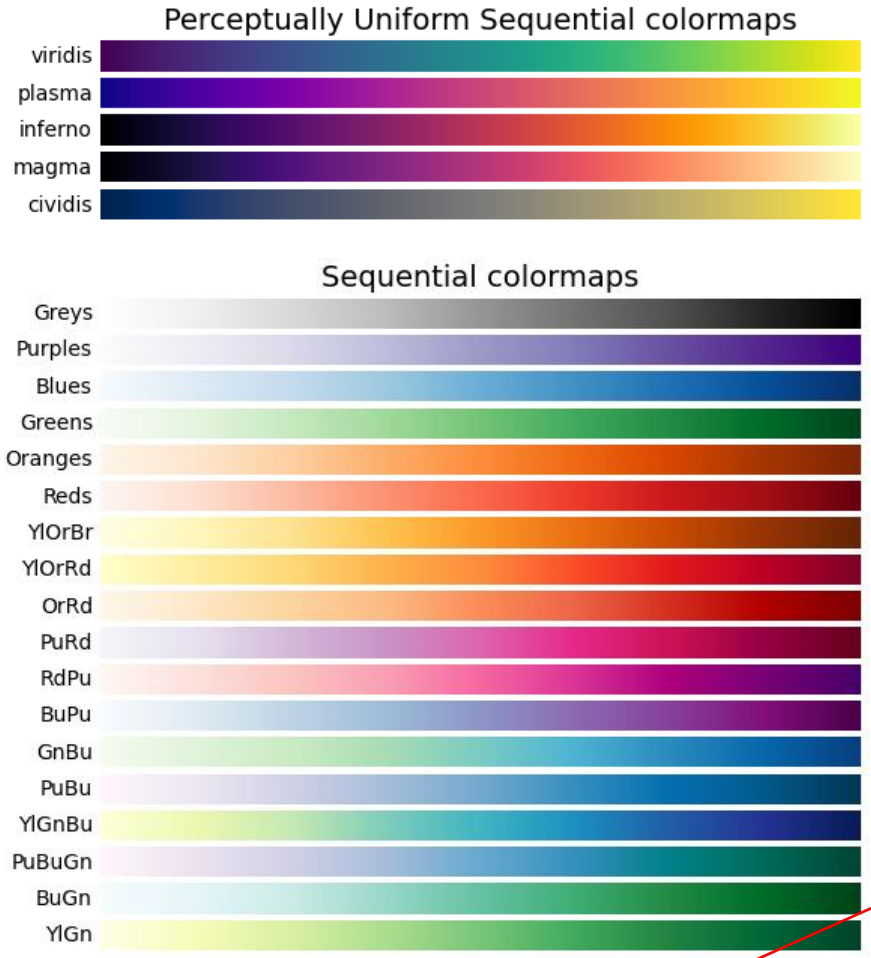
실행결과

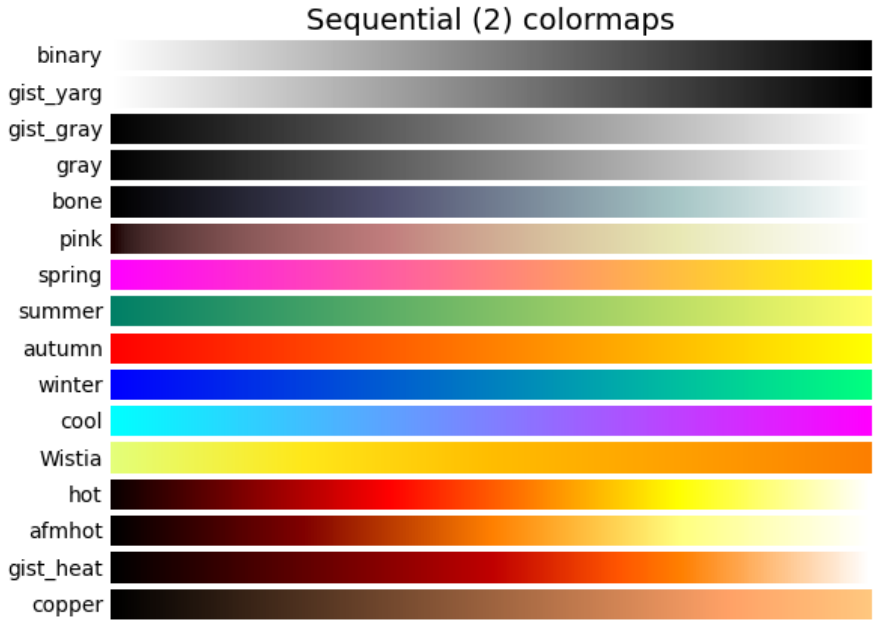


- ✓ annot 매개변수를 True로 설정하면, 각 셀에 데이터 값에 대한 주석을 달아 줌(Annotate)
- ✓ cmap 매개변수에는 원하는 컬러맵(Color Map)을 설정함

❖ 히트맵: ② seaborn 3×3 히트맵 그리기 (2/2)

- 컬러맵(Color Map) 종류





이 외에 Color Map 종류는
아래 matplotlib 사이트에서 확인하자!

[사진출처] https://matplotlib.org/stable/gallery/color/colormap_reference.html

02. seaborn 라이브러리

❖ 히트맵: ③ seaborn 8×8 히트맵 그리기 (1/3)

- 이번에는 df8 데이터프레임에서 corr( ) 함수를 이용해서 각 변수 사이의 상관계수를 구하고, 이를 히트맵으로 시각화 해보자
- NumPy 라이브러리의 triu( ) 함수를 활용해, 히트맵의 오른쪽 상단 영역을 보이지 않게 해보자
- triu( ) 함수는 입력 행렬의 상삼각행렬(Upper Triangular Matrix)를 반환해 주는 함수임
- 참고로, 하삼각행렬(Lower Triangular Matrix)를 반환해 주는 함수는 tril( )임

Upper Triangular Matrix

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

Lower Triangular Matrix

$$L = \begin{bmatrix} \ell_{1,1} & & & & 0 \\ \ell_{2,1} & \ell_{2,2} & & & \\ \ell_{3,1} & \ell_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{n,1} & \ell_{n,2} & \cdots & \ell_{n,n-1} & \ell_{n,n} \end{bmatrix}$$

[사진출처] https://en.wikipedia.org/wiki/Triangular_matrix

02. seaborn 라이브러리

❖ 히트맵: ③ seaborn 8×8 히트맵 그리기 (2/3)

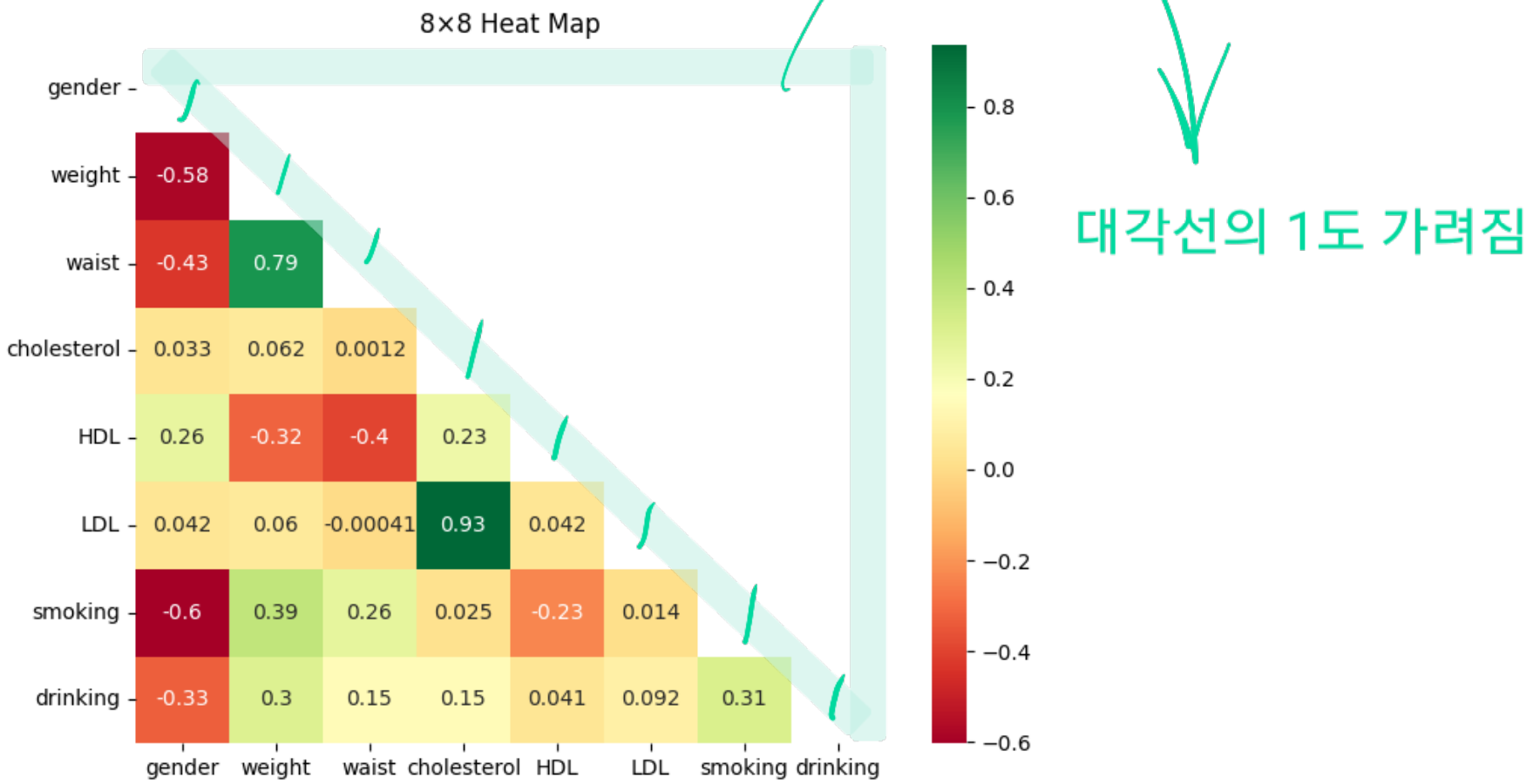
- heatmap( ) 함수의 mask 매개변수에 triu( ) 함수의 반환값을 지정하면, 히트맵의 오른쪽 상단을 안보이게 할 수 있음

```
1 plt.figure(figsize=(8, 6))
2 plt.title("8×8 Heat Map")
3 upp_mat = np.triu(df8.corr())
4 sns.heatmap(df8.corr(), annot=True, cmap="RdYlGn", mask=upp_mat)
5 plt.show()
```

02. seaborn 라이브러리

❖ 히트맵: ③ seaborn 8×8 히트맵 그리기 (3/3)

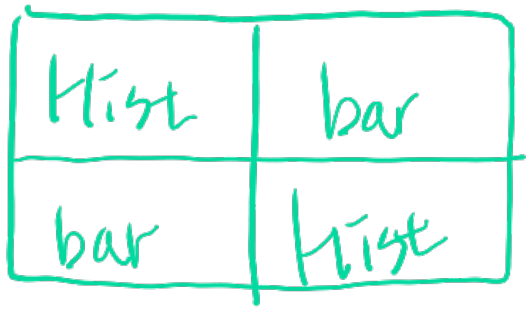
실행결과



❖ 다중 그래프: ① seaborn 패싯 그리드 (1/4)

- 패싯 그리드(Facet Grid)는 어떠한 조건에 따라 그래프를 각각 확인하고 싶을 때 사용함
- 행과 열 방향으로 서로 다른 조건을 적용하여 여러 개의 서브플롯을 만들고,
  각 서브플롯에 적용할 그래프 종류를 map( ) 함수를 이용하여 그리드 객체에 전달함

시험 X



- facet
  (명사) 측면, 양상
  (명사) (보석의 깎인) 면

**우선, 열을 기준으로 나눈 패싯 그리드를 그려 보고,
그다음 행과 열을 기준으로 나눈 패싯 그리드도 그려 보자**

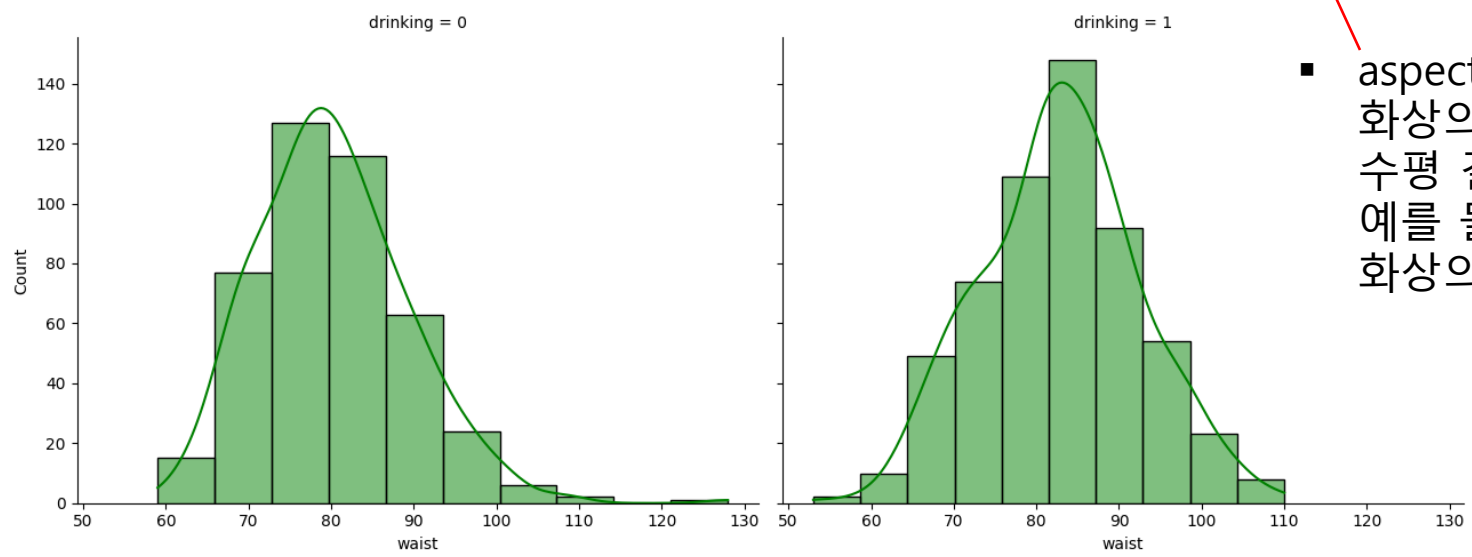
02. seaborn 라이브러리

❖ 다중 그래프: ① seaborn 패싯 그리드 (2/4)

- seaborn 라이브러리 FacetGrid( ) 함수의 col 매개변수에, 열 기준으로 나눌 음주 여부 속성을 지정하자
- 그다음 height 매개변수에 그래프 높이를 지정하고, aspect 매개변수에 세로 대비 가로 비율을 지정하자
- map( ) 함수에 서브플롯에 그릴 히스토그램 histplot( ) 함수를 전달하여, 패싯 그리드를 그려 보자

```
1 fg = sns.FacetGrid(df, col="drinking", height=5, aspect=1.3)
2 fg.map(sns.histplot, "waist", bins=10, color='g', kde=True)
3 plt.show()
```

실행결과



- aspect:
  화상의 크기를 나타내는
  수평 길이 대 수직 높이의 비율.
  예를 들어, 비율이 4 대 3이라는 것은
  화상의 수평 차원이 수직 차원의 4/3배라는 뜻

02. seaborn 라이브러리

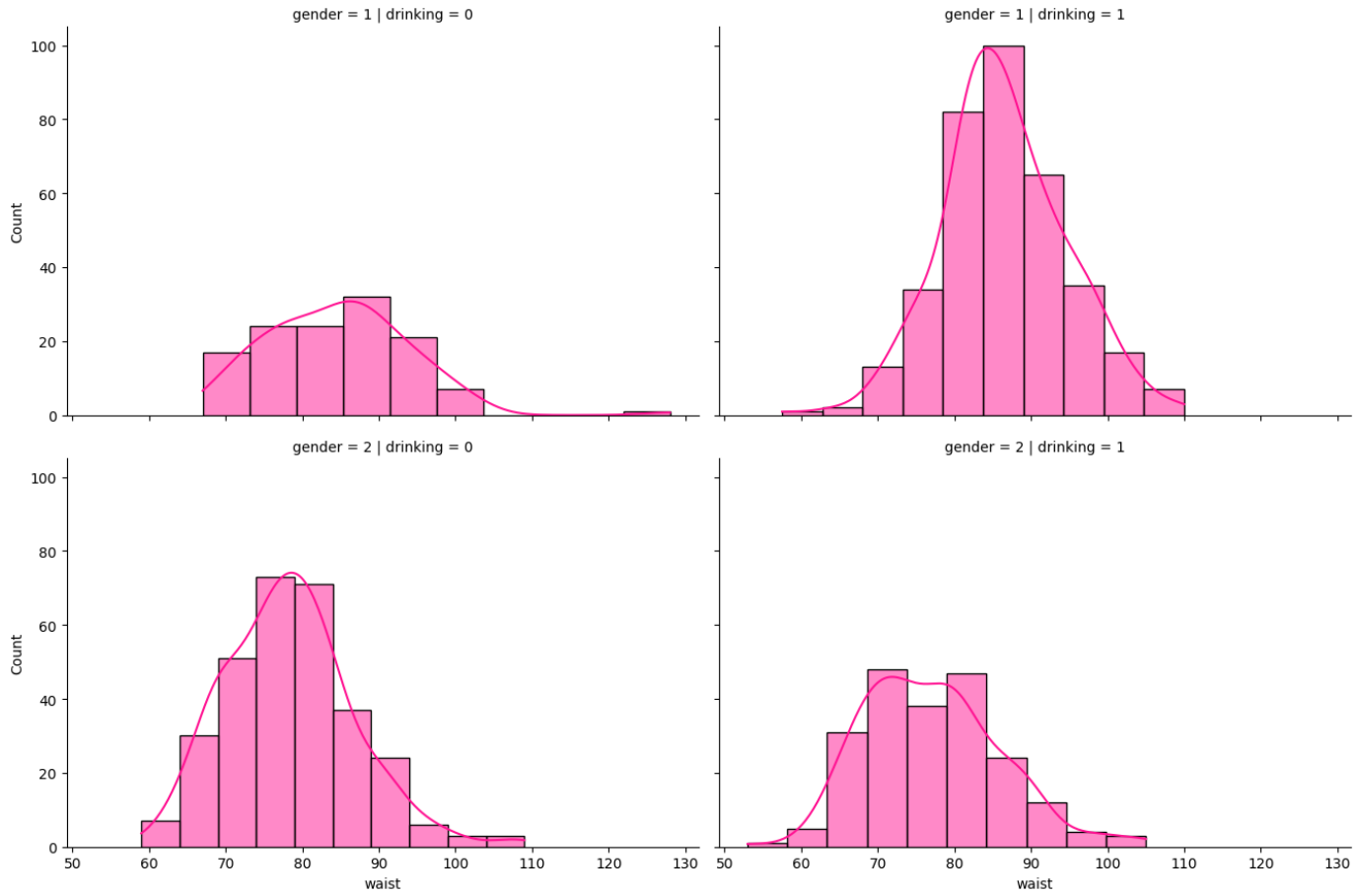
❖ 다중 그래프: ① seaborn 패싯 그리드 (3/4)

- 이번에는 행과 열을 기준으로 나눈 패싯 그리드를 그려 보자
- row 매개변수에, 행 기준으로 나눌 성별(gender) 속성을 지정하자
- col 매개변수에는, 열 기준으로 나눌 음주 여부(drinking) 속성을 지정하자

```
1 fg = sns.FacetGrid(df, row="gender", col="drinking", height=4.5, aspect=1.5)
2 fg.map(sns.histplot, "waist", bins=10, color="deeppink", kde=True)
3 plt.show()
```

❖ 다중 그래프: ① seaborn 패싯 그리드 (4/4)

실행결과



02. seaborn 라이브러리

❖ 다중 그래프: ② seaborn 페어 플롯 (1/6)

- 페어 플롯(Pair Plot)은 데이터에서 각각 2개의 열(Column) 간의 관계를 표현하는데 효과적임
- 그리드(Grid) 형태로에서 각 데이터 열 조합은 산점도로 표현하며,
  동일한 데이터가 만나는 대각선 영역에는 히스토그램으로 표현함
- 페어 플롯의 열에는 숫자형 데이터만 가능함
- 다양한 기능이나 세부적인 설정을 원할 때는, PairGrid( ) 함수를 이용하면 됨

**우선, pairplot( ) 함수로 페어 플롯을 그려 보고,
그다음 PairGrid( ) 함수로 페어 플롯을 그려 보자**

02. seaborn 라이브러리

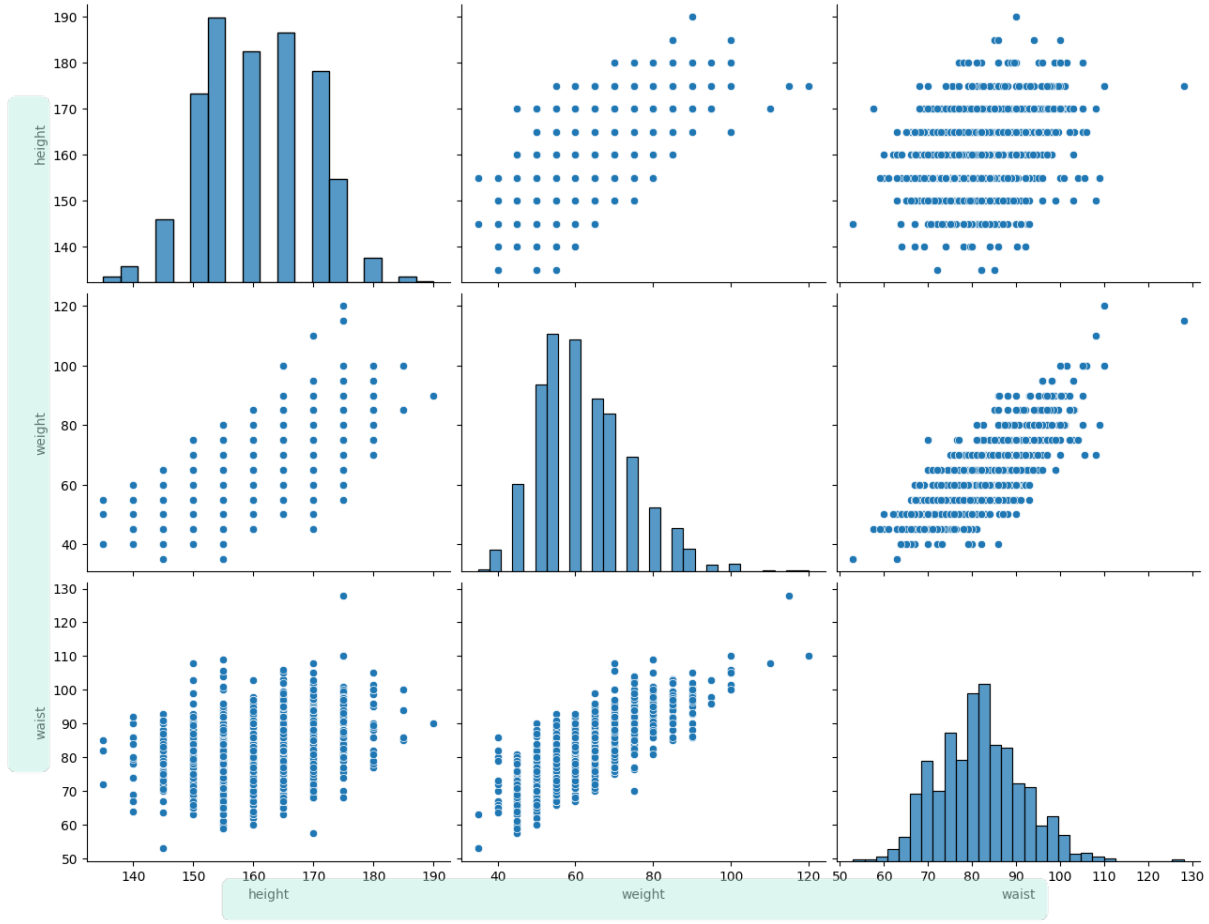
❖ 다중 그래프: ② seaborn 페어 플롯 (2/6)

- 데이터프레임에서 키(height), 몸무게(weight), 허리둘레(waist)를 추출해, pairplot( ) 함수에 전달하자
- height 매개변수에는 그래프 높이를 지정하고, aspect 매개변수에는 세로 대비 가로 비율을 지정하자

```
1 sns.pairplot(df[[“height”, “weight”, “waist”]], height=3.3, aspect=1.3)
2 plt.show()
```

❖ 다중 그래프: ② seaborn 페어 플롯 (3/6)

실행결과



02. seaborn 라이브러리

❖ 다중 그래프: ② seaborn 페어 플롯 (4/6)

- 이번에는 PairGrid( ) 함수를 이용해 페어 플롯을 그려 보자
- 성별(gender)에 따른 키(height), 몸무게(weight), 허리둘레(waist) 데이터를 비교하기 위해서, 데이터프레임에서 성별 데이터도 함께 추출해 PairGrid( ) 함수에 전달하자
- hue 매개변수에 성별 데이터를 지정하자
- palette 매개변수에 색상을 지정하는데, 헥스 색상 코드(Hex Color Codes)를 이용해 지정할 수 있음
- map_diag( ) 함수로 대각선(Diagonal Line)에 그릴 히스토그램 및 사용할 막대기이 개수를 지정함
- map_offdiag( ) 함수로 대각선 외에 그릴, 산점도를 지정하자
- 범례는 add_plot( ) 함수를 사용하면 지정할 수 있음

```
1 color = [“#00994C”, “#FF007F”]
2 pp = sns.PairGrid(df[[“gender”, “height”, “weight”, “waist”]], hue=“gender”,
3                    palette=color, height=3.3, aspect=1.3)
4 pp.map_diag(sns.histplot, bins=10)
5 pp.map_offdiag(sns.scatterplot)
6 pp.add_legend()
7 plt.show()
```

02. seaborn 라이브러리

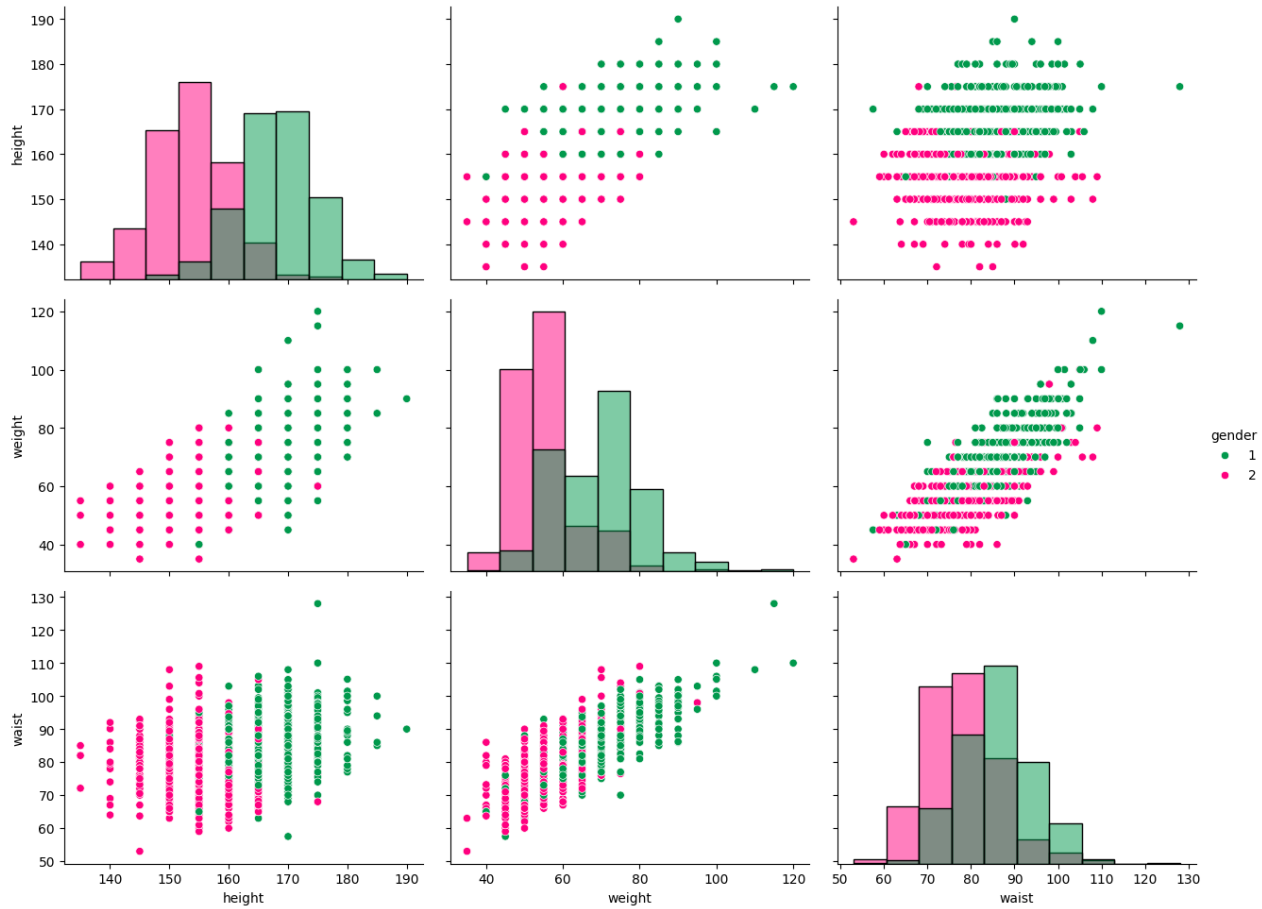
❖ 다중 그래프: ② seaborn 페어 플롯 (5/6)

헥스 색상 코드(Hex Color Codes)

- ✓ RGB 방식 색상 코드 표기법임
- ✓ #과 뒤에 붙는 여섯 자리 또는 세 자리의 숫자로 색상을 표기하는 방식임
- ✓ 여섯 자리인 경우는 두 자리씩 끊어서 각각 Red, Green, Blue의 강도를 256단계에 걸쳐 나타냄
- ✓ 각 두 자리수는 16진수이므로 0x00(=0)일 때 가장 어둡고, 0xFF(=255)일 때 가장 밝음
- ✓ 예를 들어, #00994C를 살펴보자
- ✓ Red: 0x00 → 0
- ✓ Green: 0x99 → 153
- ✓ Blue: 0x4C → 76

❖ 다중 그래프: ② seaborn 페어 플롯 (6/6)

실행결과



끝맺음

❖ 01. 데이터 시각화 이해

❖ 02. seaborn 라이브러리

THANK YOU!
Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr