Lecture 12. Seaborn 개요

기초 데이터 분석

Seaborn

- **Seaborn은 Matplotlib을 사용하는 high-level 라이브러리**
- 여러 통계 정보와 데이터 분석에 사용됨
- matplotlib과 비교하여 간결한 사용성
- 그래프 스타일 변경의 편리성
- matplotlib과 함께 사용!
- Matplotlib에서 지원하는 대부분의 함수를 동일하게 사용가능
- load_dataset()을 통한 샘플 데이터 사용

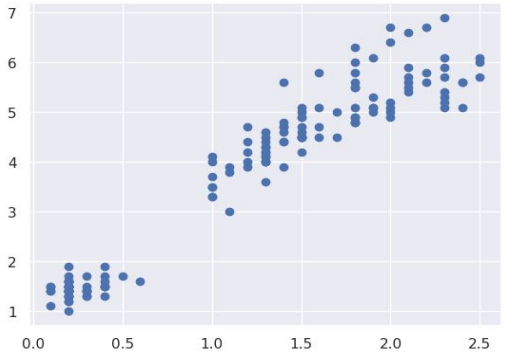
설치

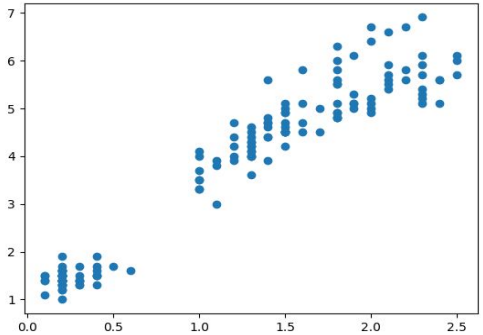
```
!pip install seaborn
```

```
import seaborn as sns
```

테마 선정

- 차트의 배경 테마를 선택 가능
- Default : ‘darkgrid’
- whitegrid, white, ticks 등 선택 가능
- seaborn의 set() 함수 사용 + style = “테마명”
- ex) sns.set(style=whitegrid)



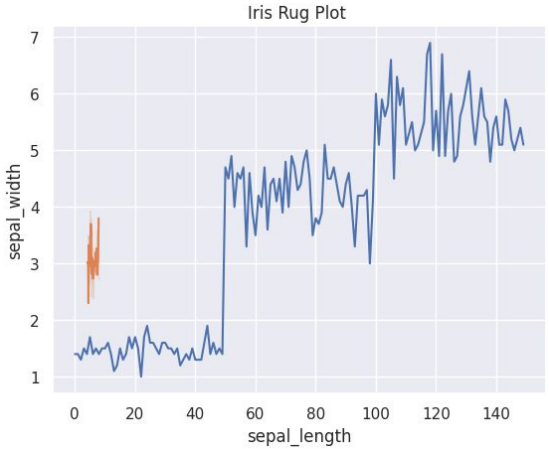


Line Plot

- 기존 matplotlib의 plt.plot()과 동일한 작동
- sns.lineplot() 함수 사용
- matplotlib과 동일하게 여러개의 선 사용 가능
- 다변량 그래프의 경우 데이터를 직접 함수에 넣어서 사용 가능

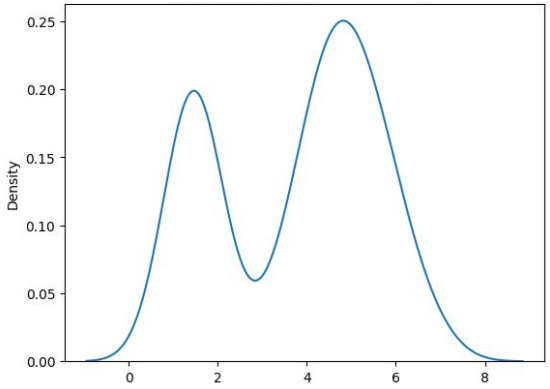
In [3]:
```
x = iris.petal_length.values
sns.lineplot(x)

sns.lineplot(data=iris,x="sepal_length",y='sepal_width')
plt.title("Iris Rug Plot ")
plt.show()
```



Single Variate Plot

- 하나의 차원의 데이터의 분포를 나타내는 차트
- 실수 분포 플롯은 자료의 분포를 묘사
- **Matplotlib**의 단순한 히스토그램과 달리 커널 밀도(**kernel density**) 및 러그(**rug**) 표시 기능 및 다차원 복합 분포 기능 등을 제공
- **rugplot(), kdeplot(), distplot()**

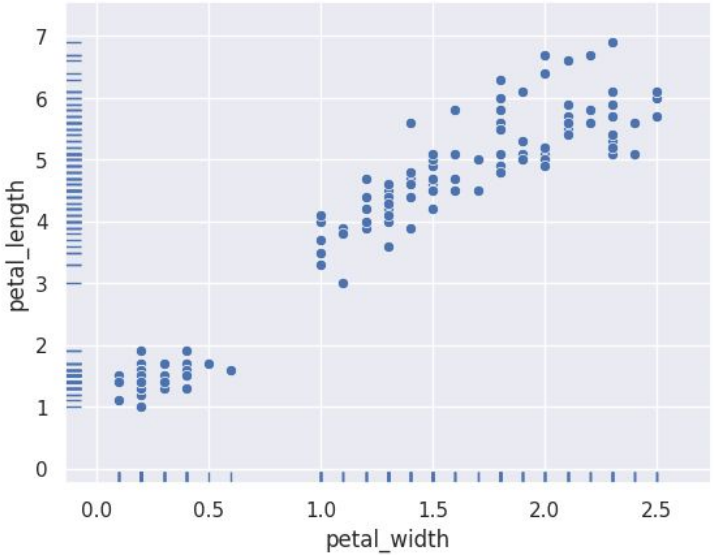


Rugplot()

- 데이터 위치를 x축 위에 작은 선분(rug)로 표현
- Scatter Plot 등 다른 Plot 함수와 함께 사용 가능

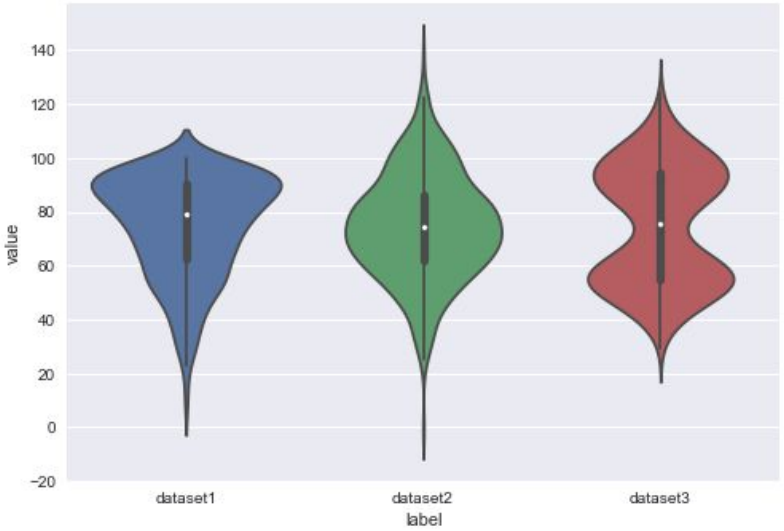
In [3]:
```
# Scatter Plot
sns.scatterplot(data=iris,
x='petal_width',y='petal_length')

# Rug Plot
sns.rugplot(data=iris,
x='petal_width',y='petal_length')
```



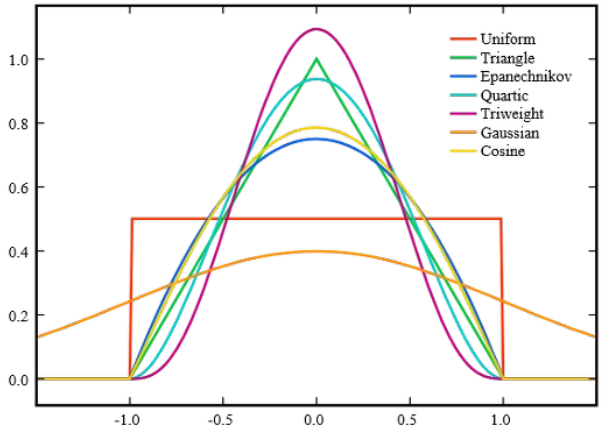
Violin Plot

- 데이터셋의 카테고리별 분포를 시각화
- matplotlib의 바이올린 플롯과 동일
- 중심에 대칭인 kdeplot
- 통계 수치
  - 흰색점 : median 값
  - 굵은 중앙선 : 사분위선
  - 얇은 중앙선 : 신뢰구간



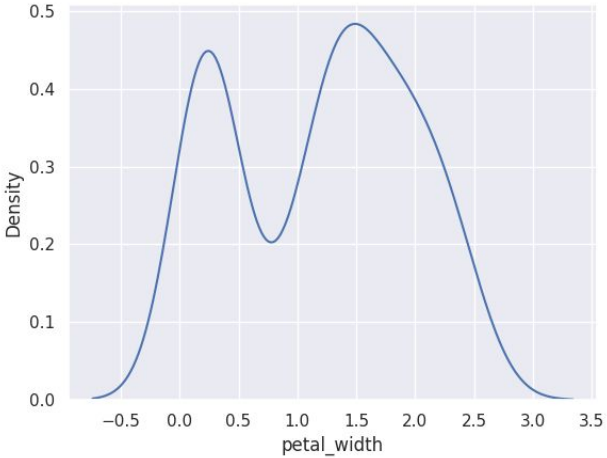
kdeplot()

- 커널 밀도 함수를 사용하여 데이터셋의 분포 시각화
- 히스토그램과 비교해 연속적인 분포 곡선
- 1,2 차원의 데이터 모두 표현 가능



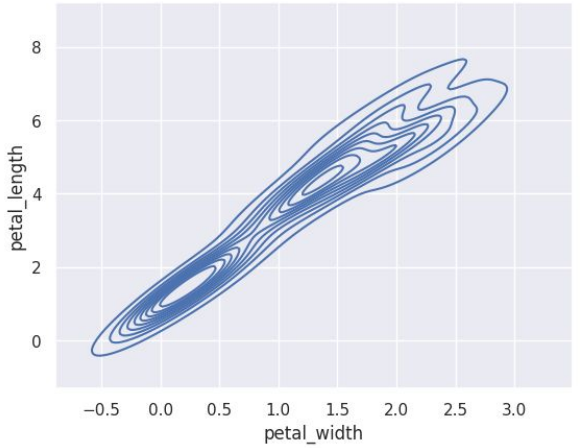
kdeplot()

- 커널 밀도 함수를 사용하여 데이터셋의 분포 시각화
- 히스토그램과 비교해 연속적인 분포 곡선
- 1,2 차원의 데이터 모두 표현 가능



In [3]:
```
sns.kdeplot(data=iris,x='petal_width')
```

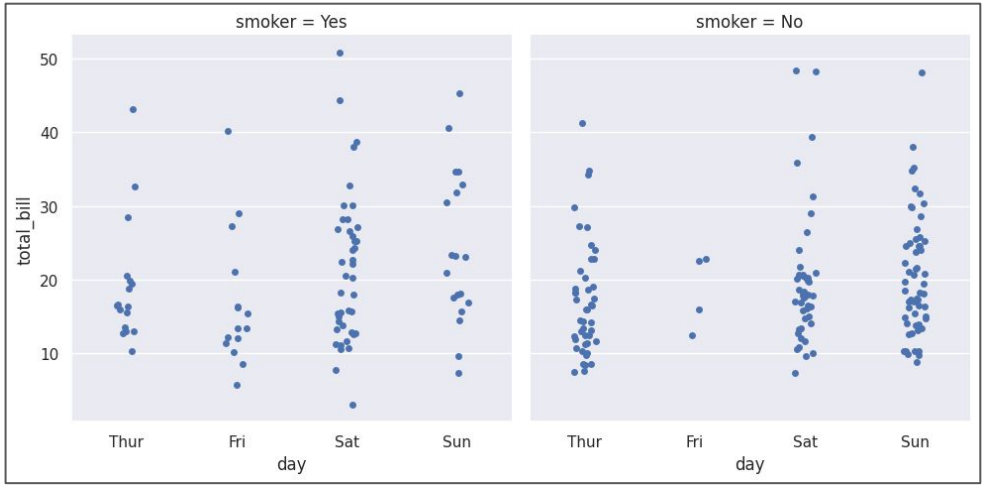
```
sns.kdeplot(data=iris, x='petal_width',y='petal_length')
```



Catplot

- 카테고리 데이터의 분포를 표현
- x : 사용할 카테고리 컬럼
- y : 수치형 데이터
- smoker : 카테고리 세분화
- kind : 분포도 선택 (box)

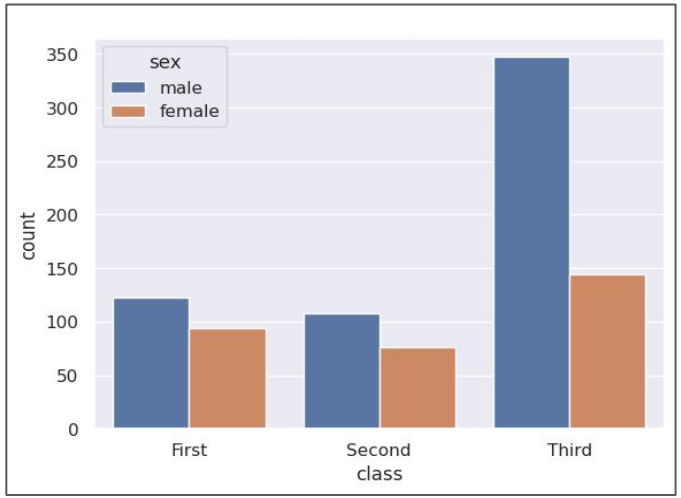
In [3]:
```
tips = sns.load_dataset('tips')
sns.catplot(data=tips, x='day',
y='total_bill',
col='smoker',
)
```



Countplot()

- 카테고리 별 갯수를 카운팅하여 시각화
- countplot() 함수는 데이터프레임에만 사용 가능
- countplot(x="column_name", data=dataframe)
- hue : 값에 다른 컬럼을 추가하여 변수 추가

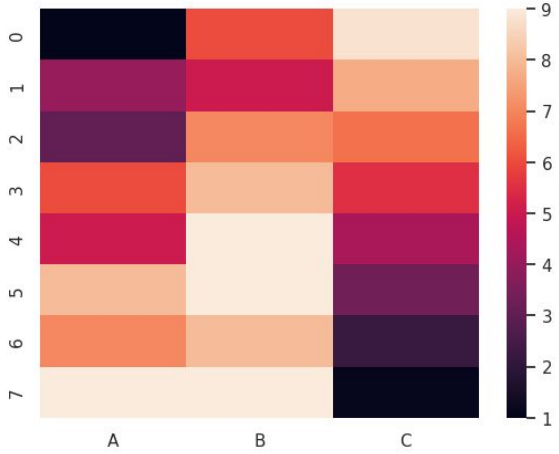
In [3]:
```
titanic = sns.load_dataset('titanic')
sns.countplot(x="class", data=titanic, hue='sex')
plt.title("타이타닉호의 각 클래스별, 승객 수")
plt.show()
```



Heatmap

- 하나의 변량이 아닌 여러개의 인자의 분포에 대한 정보 전달
- 2개 이상의 변량에 대해, 각 경우에 따른 점수를 색상으로 표현
- 피봇테이블의 경우 인자로 데이터프레임 사용
- sns.heatmap() 사용
- annot : True 설정시 수치 작성 가능

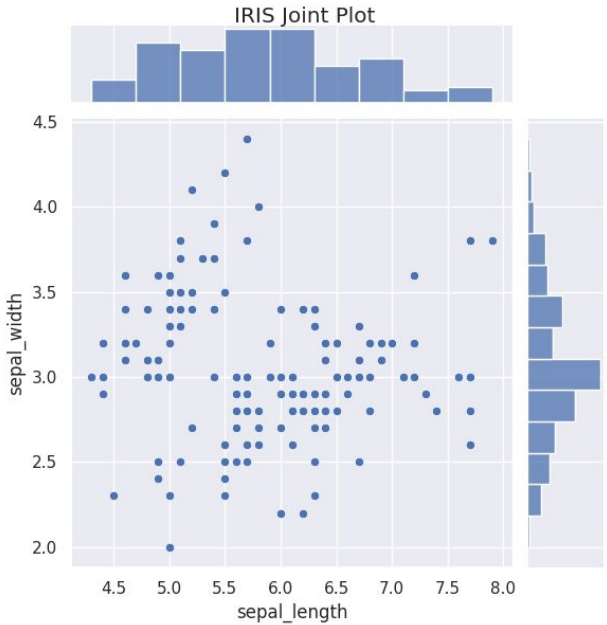
In [3]:
```
data = pd.DataFrame(data={
'A': [1,4,3,6,5,8,7,9],
'B': [6,5,7,8,9,9,8,9],
'C': [8.8,7.7,6.6,5.5,4.4,3.3,2.2,1.1]
})
sns.heatmap(data)
```



jointplot()

- ScatterPlot으로 전달하고자 하는 정보가 부족할 경우
- Jointplot은 jointplot 명령은 스캐터 플롯뿐 아니라 차트의 가장자리(margin)에 각 변수의 히스토그램도 그린다.
- kind 값을 변경하여 그래프 형식 변경 (ex. scatter)

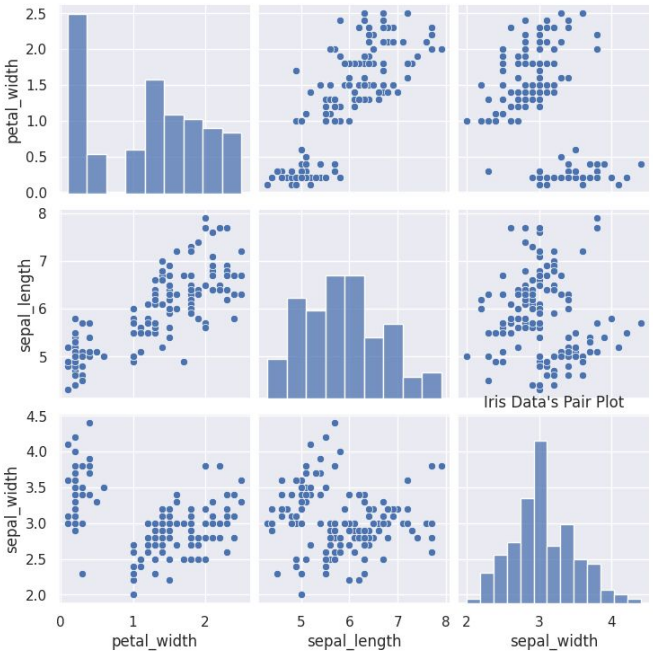
In [3]:
```
sns.jointplot(x="sepal_length",
      y="sepal_width", data=iris)
plt.suptitle("IRIS Joint Plot", y=1)
plt.show()
```



Pariplot()

- 3차원 이상의 데이터라면 **pairplot** 명령을 사용
- pairplot은 그리드(grid) 형태로 각 데이터 열의 조합에 대해 플롯
- 대각선 영역에는 해당 데이터의 히스토그램을 그린다

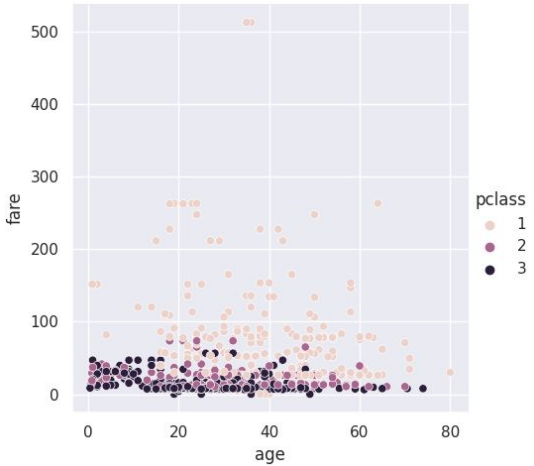
In [3]:
```
sns.pairplot(
      iris[['petal_width','sepal_length','sepal_width']])
plt.title("Iris Data's Pair Plot")
plt.show()
```



Relational Plots : relplot()

- Scatterplot() 과 lineplot() 모두 사용가능
- kind 값을 조정하여 원하는 스타일 설정 (line, scatter)
- relplot()의 리턴값은 AxesSubplot이 아닌 FacetGrid
- 그 외 사용법은 scatter, line 과 동일

In [3]:
```
sns.relplot(data=titanic,
            x='age',
            y='fare',
            hue='pclass')
```



Relational Plots : relplot()

- Scatterplot() 과 lineplot() 모두 사용가능
- kind 값을 조정하여 원하는 스타일 설정 (line, scatter)
- relplot()의 리턴값은 AxesSubplot이 아닌 FacetGrid
- 그 외 사용법은 scatter, line 과 동일

In [3]:
```
sns.relplot(data=tips,x='total_bill',
            y='tip',
            hue='smoker',col='time')
```