제조데이터 분석 및 시각화 ('22.02.22-'22.02.25) 4일차

Prepared by DaeKyeong Kim
Ph.D.

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
1일차 (○○/○○)		Syllabus Python Basics		
	5교시		Syllabus와 작업 환경 이해	강의, 실습 및 코칭
	6교시		파이썬 기초	강의, 실습 및 코칭
	7교시		파이썬 기초	강의, 실습 및 코칭
	8교시		파일 읽고 쓰기	강의, 실습 및 코칭
2일차 (○○/○○)	1교시	Data Structures Data Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		수치배열 데이터 패키지 소개	강의, 실습 및 코칭
	6교시		과학기술계산패키지 소개	강의, 실습 및 코칭
	7교시		sympy	강의, 실습 및 코칭
	8교시		데이터분석 패키지 소개	강의, 실습 및 코칭

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
3일차 (○○/○○)	1교시	데이터 셋 적재하기와 Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		데이터 셋 적재하기	강의, 실습 및 코칭
	6교시		Data Structures	강의, 실습 및 코칭
	7교시		Data Cleanup: Investigation, Matching, and Formatting	강의, 실습 및 코칭
	8교시		Data Cleanup: Investigation, Matching, and Formatting	강의, 실습 및 코칭
4일차 (○○/○○)	1교시	Data Structures Data Cleanup		
	2교시			
	3교시			
	4교시			
	5교시		시각화 패키지 소개	
	6교시		시각화 패키지 소개	
	7교시		시각화 패키지 소개	
	8교시		Iris를 통한 시각화 연습	

학습일정 및 내용

일차	시 간	교육 내용	세부 학습 내용	비고
5일차 (○○/○○)	1교시	제조데이터 분석 및 시각화 미니 프로젝트		
	2교시			
	3교시			
	4교시			
	5교시		미니 프로젝트	강의, 실습 및 코칭
	6교시		미니 프로젝트	강의, 실습 및 코칭
	7교시		미니 프로젝트	강의, 실습 및 코칭
	8교시		미니 프로젝트	강의, 실습 및 코칭

Contents

1

시각화 패키지 소개

1. matplotlib

학습목표

- Matplotlib를 임포트할 수 있다.
- Seaborn을 임포트할 수있다.
- Seaborn 고수준의 인터페이스를 통해 직관적이고 좋은 그래프를 그릴 수 있다.

Unit 2

matplotlib

Unit 1

개요

- Matplotlib는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지이다. Matplotlib는 다음과 같은 정형화된 차트나 플롯 이외에도 저수준 api를 사용한 다양한 시각화 기능을 제공한다.
  - 라인 플롯(line plot)
  - 스캐터 플롯(scatter plot)
  - 컨투어 플롯(contour plot)
  - 서피스 플롯(surface plot)
  - 바 차트(bar chart)
  - 히스토그램(histogram)
  - 박스 플롯(box plot)

pyplot 서브패키지

개요

❖ Matplotlib 패키지에는 pylab 라는 서브패키지가 존재한다. 이 pylab 서브패키지는 matlab 이라는 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용할 수 있도록 Matplotlib 의 하위 API를 포장(wrapping)한 명령어 집합을 제공한다. 간단한 시각화 프로그램을 만드는 경우에는 pylab 서브패키지의 명령만으로도 충분하다. 다음에 설명할 명령어들도 별도의 설명이 없으면 pylab 패키지의 명령라고 생각하면 된다.

❖ Matplotlib 패키지를 사용할 때는 보통 다음과 같이 주 패키지는 mpl 이라는 별칭(alias)으로 임포트하고 pylab 서브패키지는 plt 라는 다른 별칭으로 임포트하여 사용하는 것이 관례이므로 여기에서도 이러한 방법을 사용한다.

```
[ ]  import matplotlib as mpl
     import matplotlib.pyplot as plt
```

주피터 노트북을 사용하는 경우에는 다음처럼 %matplotlib 매직(magic) 명령으로 노트북 내부에 그림을 표시하도록 지정해야 한다.

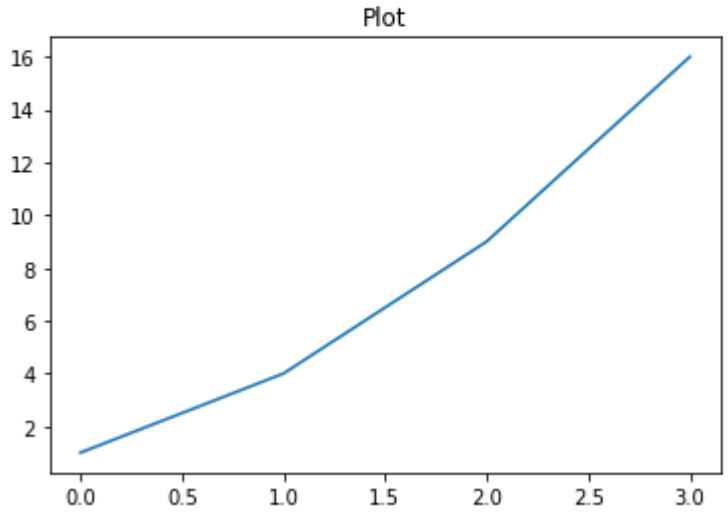
```
[ ]  %matplotlib inline
```

플롯 그려보기-라인 플롯

❖ 가장 간단한 플롯은 선을 그리는 라인 플롯(line plot)이다. 라인 플롯은 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.

- ✓ 명령은 pylab 서브패키지의 plot 명령을 사용한다.
- ✓ 만약 데이터가 1, 4, 9, 16 으로 변화하였다면 다음과 같이 plot 명령에 데이터 리스트 혹은 ndarray 객체를 넘긴다.

```
plt.title("Plot")
plt.plot([1, 4, 9, 16])
plt.show()
```



한글폰트 사용

- 맷플롯리브에서 한글을 사용하려면 다음과 같이 한글 폰트를 적용해야 한다. 당연히 해당 폰트는 컴퓨터에 깔려 있어야 한다. 여기에서는 나눔고딕 폰트를 사용하였다. 나눔고딕 폰트 설치법은 다음과 같다.

- 윈도우/맥

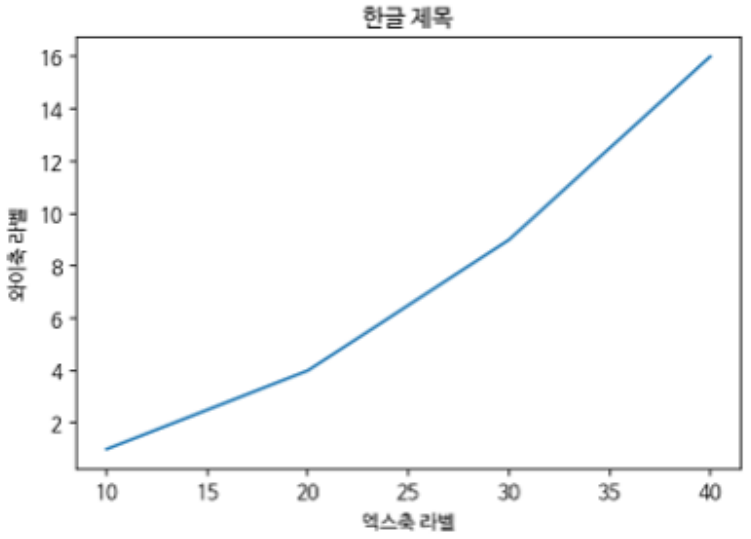
- http://hangeul.naver.com/2017/nanum 에서 폰트 인스톨러를 내려받아 실행한다.
- 리눅스

- 콘솔에서 다음과 같이 실행한다.

한글폰트 사용

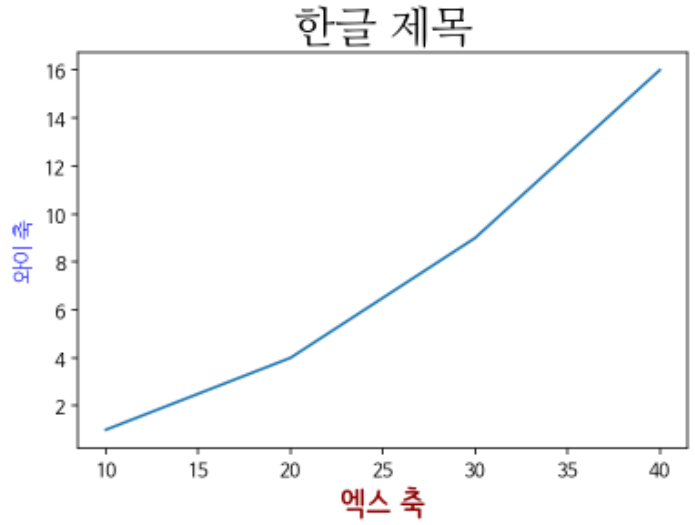
```
import matplotlib as mpl
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'retina'

!apt -qq -y install fonts-nanum

import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()
```

```
import matplotlib as mpl
import matplotlib.pyplot as plt

plt.title('한글 제목')
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.xlabel("엑스축 라벨")
plt.ylabel("와이축 라벨")
plt.show()
```



한글폰트 사용

```
font1 = {'family': 'NanumMyeongjo', 'size': 24,
         'color':  'black'}
font2 = {'family': 'NanumBarunpen', 'size': 18, 'weight': 'bold',
         'color':  'darkred'}
font3 = {'family': 'NanumBarunGothic', 'size': 12, 'weight': 'light',
         'color':  'blue'}

plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.title('한글 제목', fontdict=font1)
plt.xlabel('엑스 축', fontdict=font2)
plt.ylabel('와이 축', fontdict=font3)
plt.show()
```



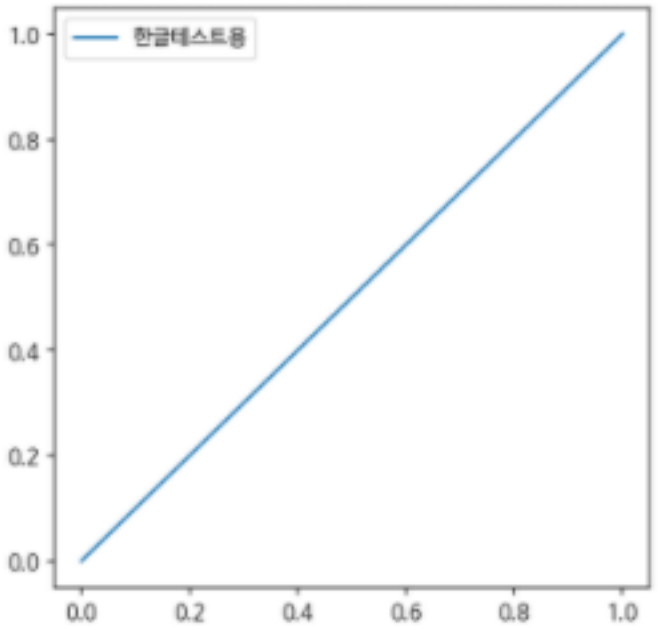
한글폰트 사용-또 다른 방법

```
import matplotlib as mpl
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'retina'

!apt -qq -y install fonts-nanum

import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()
```

```
fonts-nanum is already the newest version (20170925-1).
The following packages were automatically installed and are no longer required:
  cuda-command-line-tools-10-0 cuda-command-line-tools-10-1
  cuda-command-line-tools-11-0 cuda-compiler-10-0 cuda-compiler-10-1
  cuda-compiler-11-0 cuda-cuobjdump-10-0 cuda-cuobjdump-10-1
  cuda-cuobjdump-11-0 cuda-cupti-10-0 cuda-cupti-10-1 cuda-cupti-11-0
```

한글폰트 사용-또 다른 방법

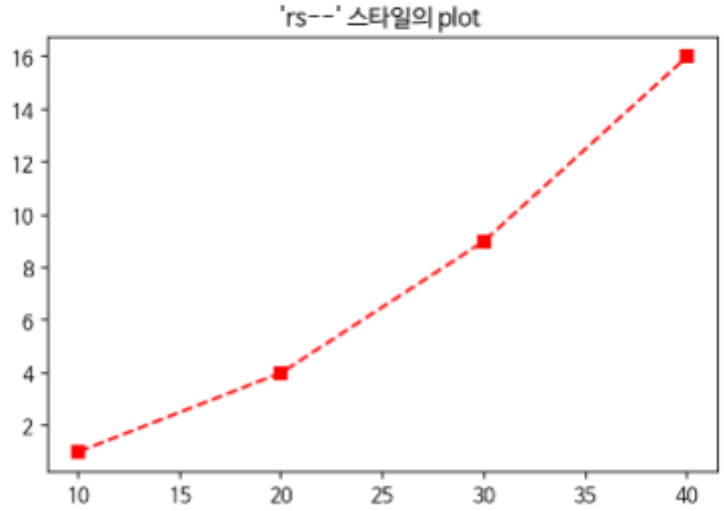
```
plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```



스타일 지정사용

❖ 플롯 명령어는 보는 사람이 그림을 더 알아보기 쉽게 하기 위해 다양한 스타일(style)을 지원한다. plot 명령어에서는 다음과 같이 추가 문자열 인수를 사용하여 스타일을 지원한다.

```
plt.title("'rs--' 스타일의 plot ")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16], 'rs--')
plt.show()
```



스타일 지정사용

❖ 색깔

- ✓ 색깔을 지정하는 방법은 색 이름 혹은 약자를 사용하거나 # 문자로 시작되는 RGB코드를 사용한다.
- ✓ 자주 사용되는 색깔은 한글자 약자를 사용할 수 있으며 약자는 아래 표에 정리하였다. 전체 색깔 목록은 다음 웹사이트를 참조한다.
- ✓ http://matplotlib.org/examples/color/named_colors.html
- ✓ 문자열, 약자

문자열	약자
blue	b
green	g
red	r
cyan	c
magenta	m
yellow	y
black	k
white	w

스타일 지정사용

❖ 마커

✓ 데이터 위치를 나타내는 기호를 마커(marker)라고 한다. 마커의 종류는 다음과 같다.

마커 문자열	의미
.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker
4	tri_right marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker

스타일 지정사용

❖ 선 스타일

✓ 선 스타일에는 실선(solid), 대시선(dashed), 점선(dotted), 대시-점선(dash-dit) 이 있다. 지정 문자열은 다음과 같다.

선 스타일 문자열	의미
-	solid line style
--	dashed line style
-.	dash-dot line style
:	dotted line style

스타일 지정사용

❖ 기타 스타일

- ✓ 라인 플롯에서는 앞서 설명한 세 가지 스타일 이외에도 여러가지 스타일을 지정할 수 있지만 이 경우에는 인수 이름을 정확하게 지정해야 한다. 사용할 수 있는 스타일 인수의 목록은 matplotlib.lines.Line2D 클래스에 대한 다음 웹사이트를 참조한다.

- ✓ http://matplotlib.org/api/lines_api.html#matplotlib.lines.Line2D
- ✓ 라인 플롯에서 자주 사용되는 기타 스타일은 다음과 같다.

스타일 문자열	약자	의미
color	c	선 색깔
linewidth	lw	선 굵기
linestyle	ls	선 스타일
marker		마커 종류
markersize	ms	마커 크기
markeredgecolor	mec	마커 선 색깔
markeredgewidth	mew	마커 선 굵기
markerfacecolor	mfc	마커 내부 색깔

스타일 적용 예

```
plt.plot([10, 20, 30, 40], [1, 4, 9, 16], c="b",
         lw=5, ls="--", marker="o", ms=15, mec="g", mew=5, mfc="r")
plt.title("스타일 적용 예")
plt.show()
```

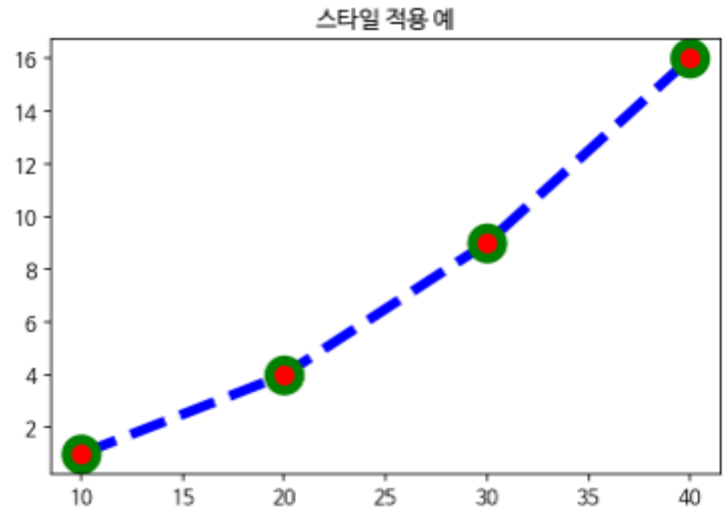
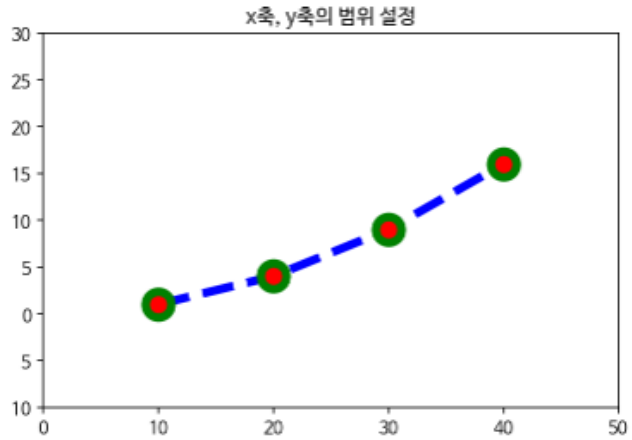


그림 범위 지정

❖ 플롯 그림을 보면 몇몇 점들은 그림의 범위 경계선에 있어서 잘 보이지 않는 경우가 있을 수 있다. 그림의 범위를 수동으로 지정하려면 xlim 명령과 ylim 명령을 사용한다. 이 명령들은 그림의 범위가 되는 x축, y축의 최소값과 최대값을 지정한다.

```
plt.title("x축, y축의 범위 설정")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16],
         c="b", lw=5, ls="--", marker="o", ms=15, mec="g", mew=5, mfc="r")
plt.xlim(0, 50)
plt.ylim(-10, 30)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183:
  font.set_text(s, 0, flags=flags)
```

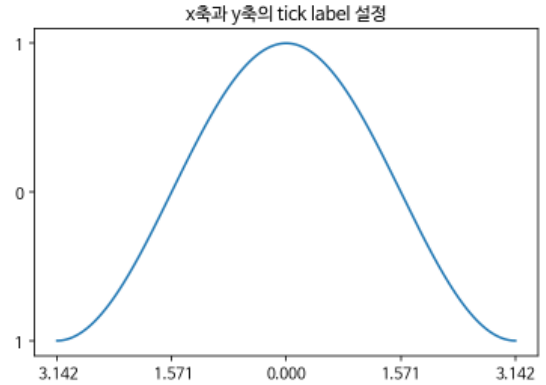


틱 설정

❖ 플롯이나 차트에서 축상의 위치 표시 지점을 틱(tick)이라고 하고 이 틱에 써진 숫자 혹은 글자를 틱 라벨(tick label)이라고 한다. 틱의 위치나 틱 라벨은 맷플롯리브가 자동으로 정해주지만 만약 수동으로 설정하고 싶다면 xticks 명령이나 yticks 명령을 사용한다.

```
import numpy as np

X = np.linspace(-np.pi, np.pi, 256)
C = np.cos(X)
plt.title("x축과 y축의 tick label 설정")
plt.plot(X, C)
plt.xticks([-np.pi, -np.pi / 2, 0, np.pi / 2, np.pi])
plt.yticks([-1, 0, +1])
plt.show()
```

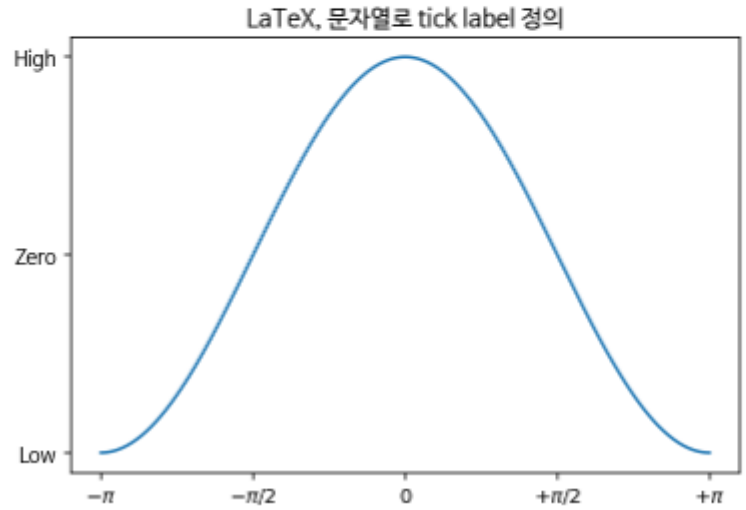
```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/ba
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/ba
  font.set_text(s, 0, flags=flags)
```



틱 설정

❖ 틱 라벨 문자열에는 $$ 사이에 LaTeX 수학 문자식을 넣을 수도 있다.

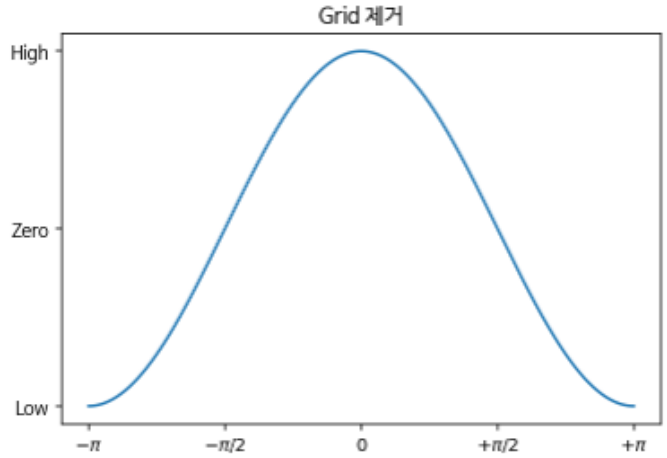
```
X = np.linspace(-np.pi, np.pi, 256)
C = np.cos(X)
plt.title("LaTeX, 문자열로 tick label 정의")
plt.plot(X, C)
plt.xticks([-np.pi, -np.pi / 2, 0, np.pi / 2, np.pi],
           [r'$-₩pi$', r'$-₩pi/2$', r'$0$', r'$+₩pi/2$', r'$+₩pi$'])
plt.yticks([-1, 0, 1], ["Low", "Zero", "High"])
plt.show()
```



그리드 설정

❖ 틱 위치를 잘 보여주기 위해 그림 중간에 그리드 선(grid line)이 자동으로 그려진 것을 알 수 있다. 그리드를 사용하지 않으려면 grid(False) 명령을 사용한다. 다시 그리드를 사용하려면 grid(True)를 사용한다.

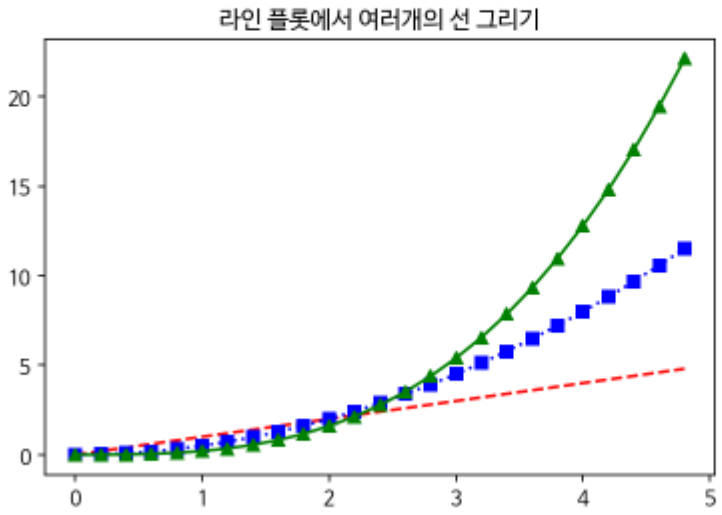
```
X = np.linspace(-np.pi, np.pi, 256)
C = np.cos(X)
plt.title("Grid 제거")
plt.plot(X, C)
plt.xticks([-np.pi, -np.pi / 2, 0, np.pi / 2, np.pi],
           [r'$-₩pi$', r'$-₩pi/2$', r'$0$', r'$+₩pi/2$', r'$+₩pi$'])
plt.yticks([-1, 0, 1], ["Low", "Zero", "High"])
plt.grid(False)
plt.show()
```



여러개의 선을 그리기

❖ 라인 플롯에서 선을 하나가 아니라 여러개를 그리고 싶은 경우에는 x 데이터, y 데이터, 스타일 문자열을 반복하여 인수로 넘긴다. 이 경우에는 하나의 선을 그릴 때 처럼 x 데이터나 스타일 문자열을 생략할 수 없다.

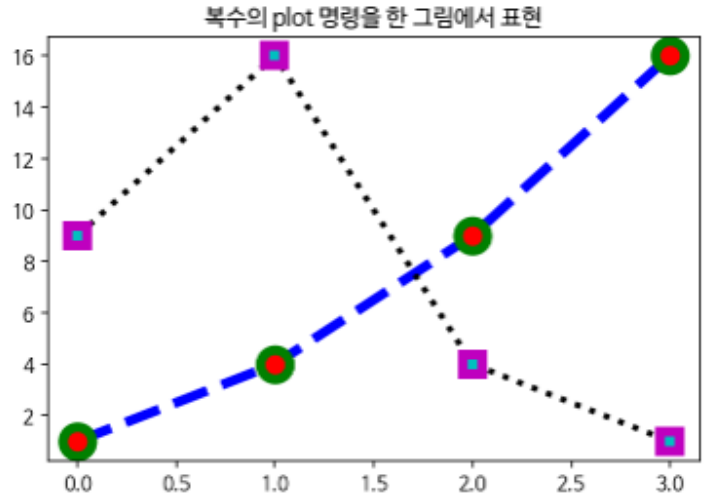
```
t = np.arange(0., 5., 0.2)
plt.title("라인 플롯에서 여러개의 선 그리기")
plt.plot(t, t, 'r--', t, 0.5 * t**2, 'bs:', t, 0.2 * t**3, 'g^-')
plt.show()
```



겹쳐그리기

❖ 하나의 plot 명령이 아니라 복수의 plot 명령을 하나의 그림에 겹쳐서 그릴 수도 있다.

```
plt.title("복수의 plot 명령을 한 그림에서 표현")
plt.plot([1, 4, 9, 16],
         c="b", lw=5, ls="--", marker="o", ms=15, mec="g", mew=5, mfc="r")
# plt.hold(True)   # <- 1,5 버전에서는 이 코드가 필요하다.
plt.plot([9, 16, 4, 1],
         c="k", lw=3, ls=":", marker="s", ms=10, mec="m", mew=5, mfc="c")
# plt.hold(False)  # <- 1,5 버전에서는 이 코드가 필요하다.
plt.show()
```



범례

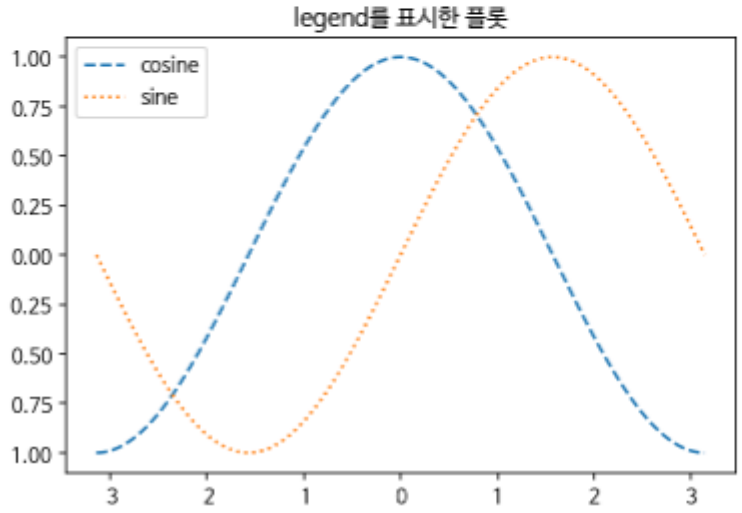
❖ 여러개의 라인 플롯을 동시에 그리는 경우에는 각 선이 무슨 자료를 표시하는지를 보여주기 위해 legend 명령으로 범례(legend)를 추가할 수 있다. 범례의 위치는 자동으로 정해지지만 수동으로 설정하고 싶으면 loc 인수를 사용한다. 인수에는 문자열 혹은 숫자가 들어가며 가능한 코드는 다음과 같다.

loc 문자열	숫자
best	0
upper right	1
upper left	2
lower left	3
lower right	4
right	5
center left	6
center right	7
lower center	8
upper center	9
center	10

범례

```
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)
plt.title("legend를 표시한 플롯")
plt.plot(X, C, ls="--", label="cosine")
plt.plot(X, S, ls=":", label="sine")
plt.legend(loc=2)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:21
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:18
  font.set_text(s, 0, flags=flags)
```

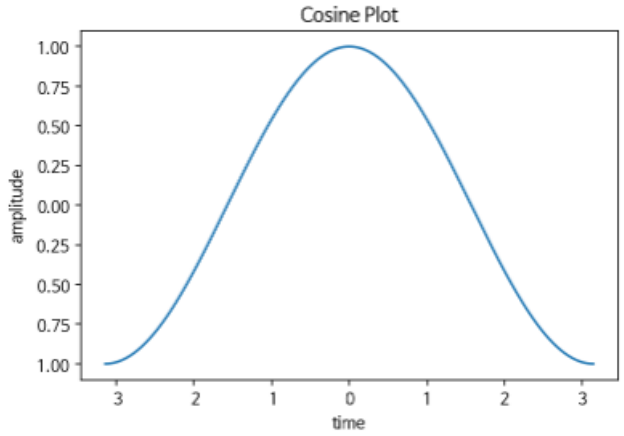


x축, y축 라벨, 타이틀

❖ 플롯의 x축 위치와 y축 위치에는 각각 그 데이터가 의미하는 바를 표시하기 위해 라벨(label)를 추가할 수 있다. 라벨을 붙이려면 xlabel. ylabel 명령을 사용한다. 또 플롯의 위에는 title 명령으로 제목(title)을 붙일 수 있다.

```
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)
plt.plot(X, C, label="cosine")
plt.xlabel("time")
plt.ylabel("amplitude")
plt.title("Cosine Plot")
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:
  font.set_text(s, 0, flags=flags)
```



그림의 구조

❖ 맷플롯리브가 그리는 그림은 Figure 객체, Axes 객체, Axis 객체 등으로 구성된다. Figure 객체는 한 개 이상의 Axes 객체를 포함하고 Axes 객체는 다시 두 개 이상의 Axis 객체를 포함한다.

❖ Figure는 그림이 그려지는 캔버스나 종이를 뜻하고 Axes는 하나의 플롯, 그리고 Axis는 가로축이나 세로축 등의 축을 뜻한다. Axes와 Axis의 철자에 주의한다.

❖ Figure 객체

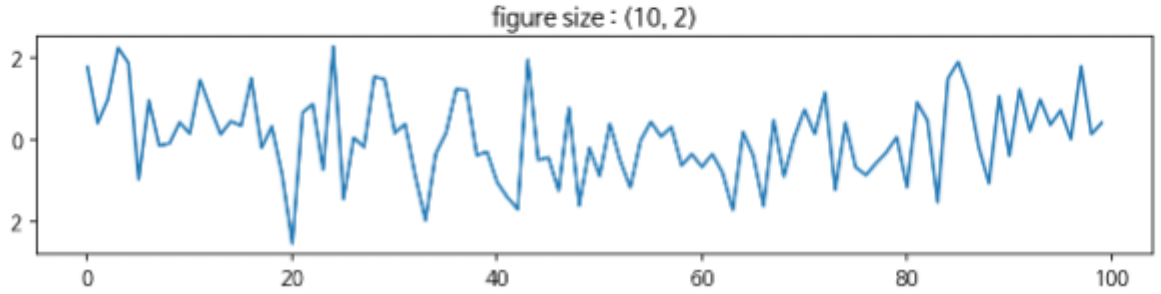
- ✓ 모든 그림은 Figure 객체. 정식으로는 matplotlib.figure.Figure 클래스 객체에 포함되어 있다. 내부 플롯(inline plot)이 아닌 경우에는 하나의 Figure는 하나의 아이디 숫자와 윈도우(Window)를 가진다. 주피터 노트북에서는 윈도우 객체가 생성되지 않지만 파이썬을 독립 실행하는 경우에는 하나의 Figure 당 하나의 윈도우를 별도로 가진다.

그림의 구조

✓ Figure를 생성하려면 figure 명령을 사용하여 그 반환값으로 Figure 객체를 얻어야 한다. 그러나 일반적인 plot 명령 등을 실행하면 자동으로 Figure를 생성해주기 때문에 일반적으로는 figure 명령을 잘 사용하지 않는다. figure 명령을 명시적으로 사용하는 경우는 여러개의 윈도우를 동시에 띄워야 하거나(line plot이 아닌 경우), Jupyter 노트북 등에서(line plot의 경우) 그림의 크기를 설정하고 싶을 때이다. 그림의 크기는 figsize 인수로 설정한다.

```
np.random.seed(0)
f1 = plt.figure(figsize=(10, 2))
plt.title("figure size : (10, 2)")
plt.plot(np.random.randn(100))
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph 8722 missing f
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph 8722 missing f
  font.set_text(s, 0, flags=flags)
```

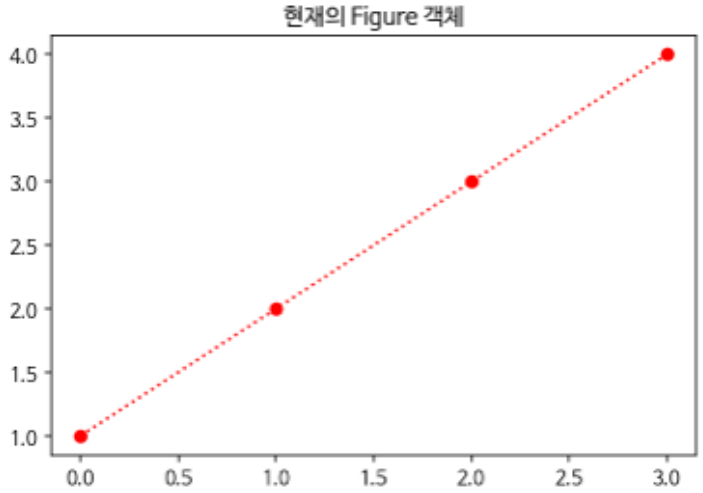


그림의 구조

❖ 현재 사용하고 있는 Figure 객체를 얻으려면(다른 변수에 할당할 수도 있다.) gcf 명령을 사용한다.

```
f1 = plt.figure(1)
plt.title("현재의 Figure 객체")
plt.plot([1, 2, 3, 4], 'ro:')

f2 = plt.gcf()
print(f1, id(f1))
print(f2, id(f2))
plt.show()
```

```
Figure(432x288) 139761462637584
Figure(432x288) 139761462637584
```



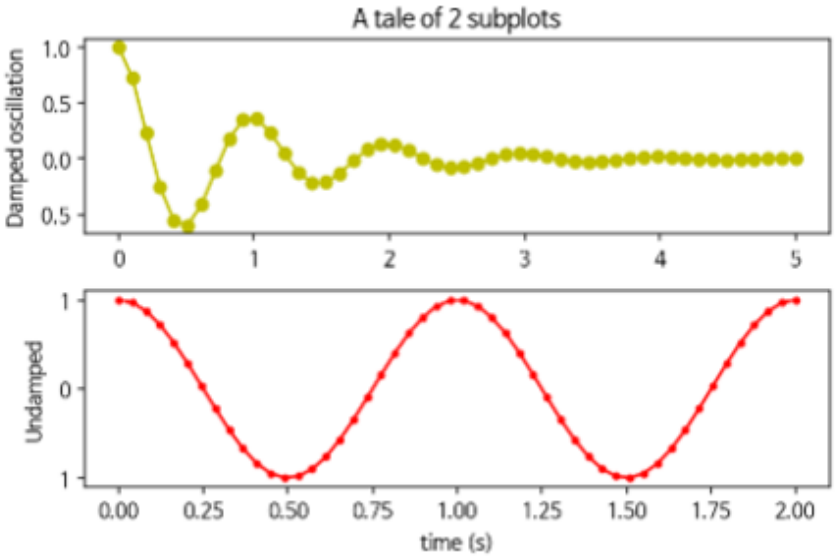
그림의 구조

❖ Axes 객체와 subplot 명령

- ✓ 때로는 다음과 같이 하나의 윈도우(Figure)안에 여러개의 플롯을 배열 형태로 보여야하는 경우도 있다. Figure 안에 있는 각각의 플롯은 Axes 라고 불리는 객체에 속한다. Figure 안에 Axes를 생성하려면 원래 subplot 명령을 사용하여 명시적으로 Axes 객체를 얻어야 한다. 그러나 plot 명령을 바로 사용해도 자동으로 Axes를 생성해 준다.
- ✓ subplot 명령은 그리드(grid) 형태의 Axes 객체들을 생성하는데 Figure가 행렬(matrix)이고 Axes가 행렬의 원소라고 생각하면 된다. 숫자 인덱싱은 파이썬이 아닌 Matlab 관행을 따르기 때문에 첫번째 플롯을 가리키는 숫자가 0이 아니라 1임에 주의
- ✓ tight_layout 명령을 실행하면 플롯간의 간격을 자동으로 맞춰준다.

그림의 구조

```
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

ax1 = plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'yo-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')

ax2 = plt.subplot(2, 1, 2)
plt.plot(x2, y2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

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

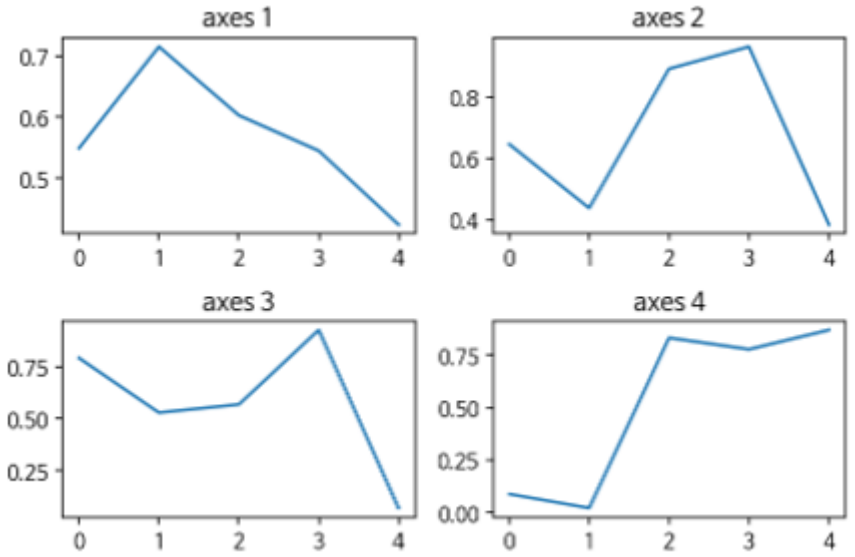


그림의 구조

❖ Axes 객체와 subplot 명령

✓ 만약 2x2 형태의 네 개의 플롯이라면 다음과 같이 그린다. 이 때 subplot 의 인수는 (2,2,1)를 줄여서 221 라는 하나의 숫자로 표시할 수도 있다. Axes의 위치는 위에서 부터 아래로, 왼쪽에서 오른쪽으로 카운트한다.

```
np.random.seed(0)

plt.subplot(221)
plt.plot(np.random.rand(5))
plt.title("axes 1")

plt.subplot(222)
plt.plot(np.random.rand(5))
plt.title("axes 2")

plt.subplot(223)
plt.plot(np.random.rand(5))
plt.title("axes 3")

plt.subplot(224)
plt.plot(np.random.rand(5))
plt.title("axes 4")

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

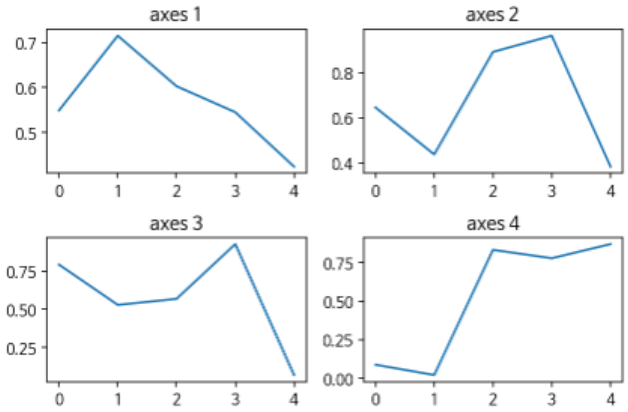


그림의 구조

❖ Axes 객체와 subplot 명령

✓ subplots 명령으로 복수의 Axes 객체를 동시에 생성할 수도 있다. 이때는 2차원 ndarray 형태로 Axes 객체가 반환된다.

```
fig, axes = plt.subplots(2, 2)

np.random.seed(0)
axes[0, 0].plot(np.random.rand(5))
axes[0, 0].set_title("axes 1")
axes[0, 1].plot(np.random.rand(5))
axes[0, 1].set_title("axes 2")
axes[1, 0].plot(np.random.rand(5))
axes[1, 0].set_title("axes 3")
axes[1, 1].plot(np.random.rand(5))
axes[1, 1].set_title("axes 4")

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

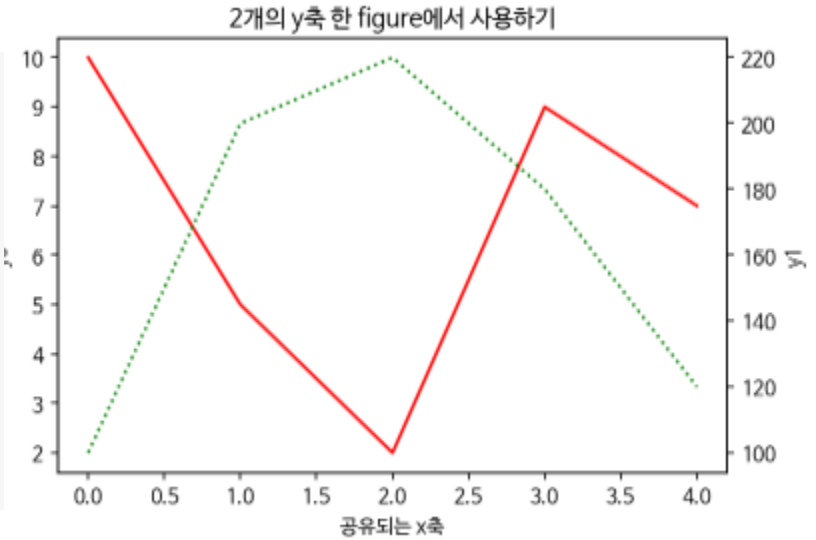


그림의 구조

❖ Axis 객체와 축

✓ 하나의 Axes 객체는 두 개 이상의 Axis 객체를 가진다. Axis 객체는 플롯의 가로축이나 세로축을 나타내는 객체이다. 러가지 플롯을 하나의 Axes 객체에 표시할 때 y값의 크기가 달라서 표시하기 힘든 경우가 있다. 이 때는 다음처럼 twinx 명령으로 대해 복수의 y 축을 가진 플롯을 만들수도 있다. twinx 명령은 x 축을 공유하는 새로운 Axes 객체를 만든다.

```
fig, ax0 = plt.subplots()
ax1 = ax0.twinx()
ax0.set_title("2개의 y축 한 figure에서 사용하기")
ax0.plot([10, 5, 2, 9, 7], 'r-', label="y0")
ax0.set_ylabel("y0")
ax0.grid(False)
ax1.plot([100, 200, 220, 180, 120], 'g:', label="y1")
ax1.set_ylabel("y1")
ax1.grid(False)
ax0.set_xlabel("공유되는 x축")
plt.show()
```



Unit 2

Seaborn

Unit 1

Seaborn?

개요

❖ Seaborn은 Matplotlib에 기반하여 제작된 파이썬 데이터 시각화 모듈이다. 고수준의 인터페이스를 통해 직관적이고 좋은 그래프를 그릴 수 있다.

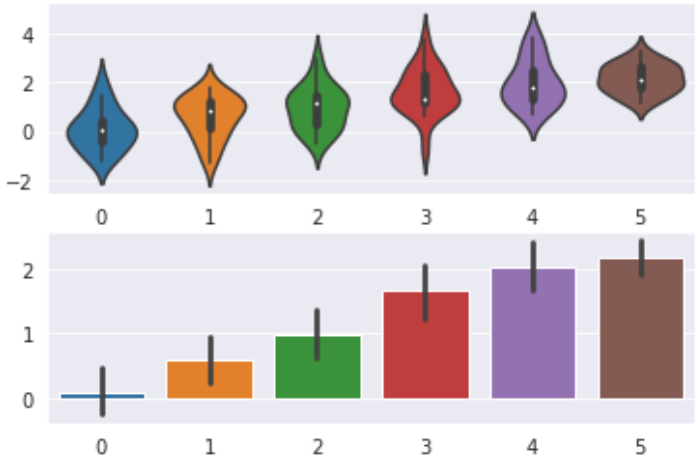
Style Management

- sns.set_style(style=None, rc=None) :: 그래프 배경을 설정함
  - ✓ style = “darkgrid”, “whitegrid”, “dark”, “white”, “ticks”
  - ✓ rc = [dict], 세부 사항을 조정함
- sns.despine(offset=None, trim=False, top=True, right=True, left=False, bottom=False) :: Plot의 위, 오른쪽 축을 제거함
  - ✓ top, right, left, bottom = True로 설정하면 그 축을 제거함
  - ✓ offset = [integer or dict], 축과 실제 그래프가 얼마나 떨어져 있을지 설정함

Style Management

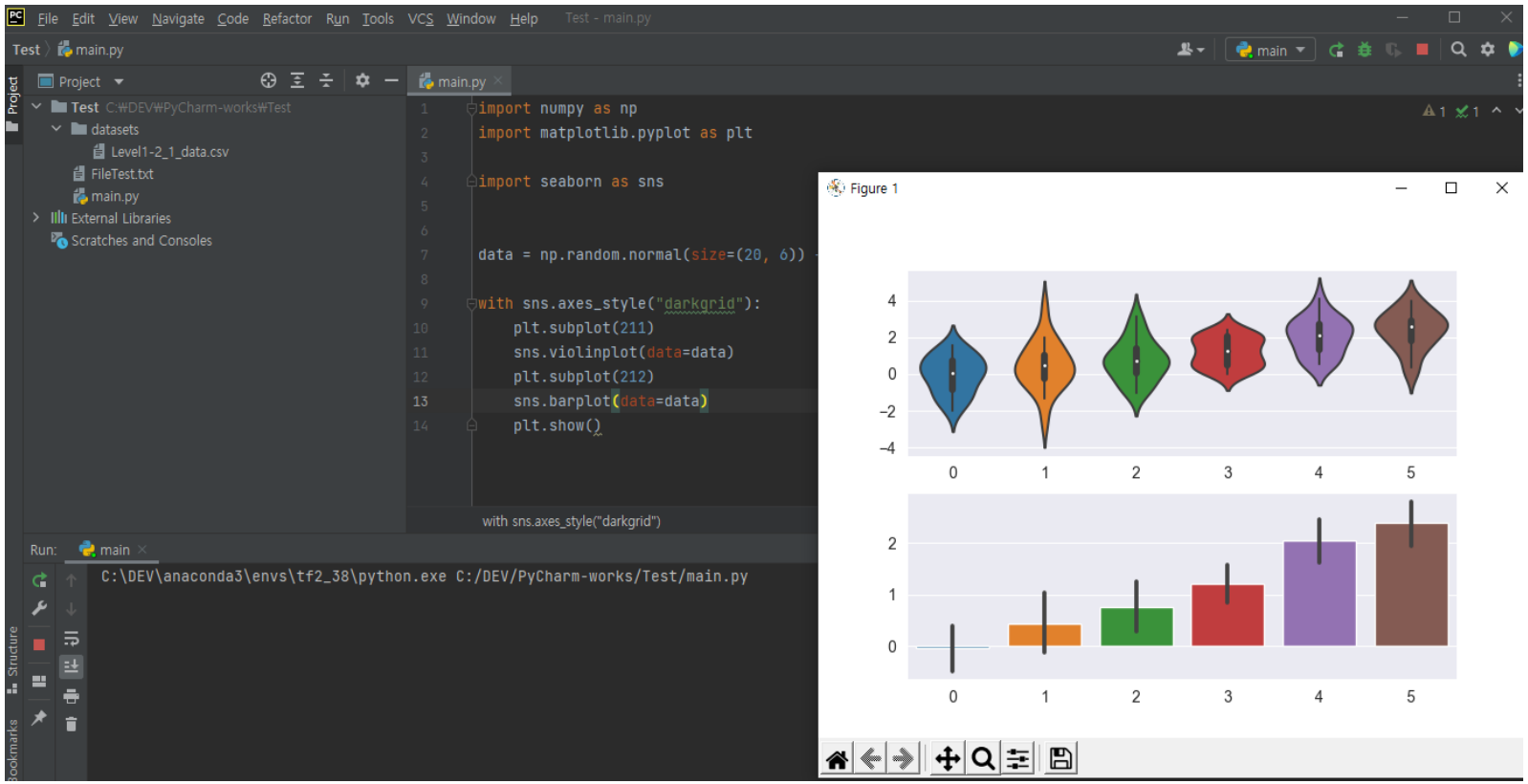
```
import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns
```

```
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2

with sns.axes_style("darkgrid"):
    plt.subplot(211)
    sns.violinplot(data=data)
    plt.subplot(212)
    sns.barplot(data=data)
    plt.show()
```



Style Management

❖ 파이참에서 실행 시



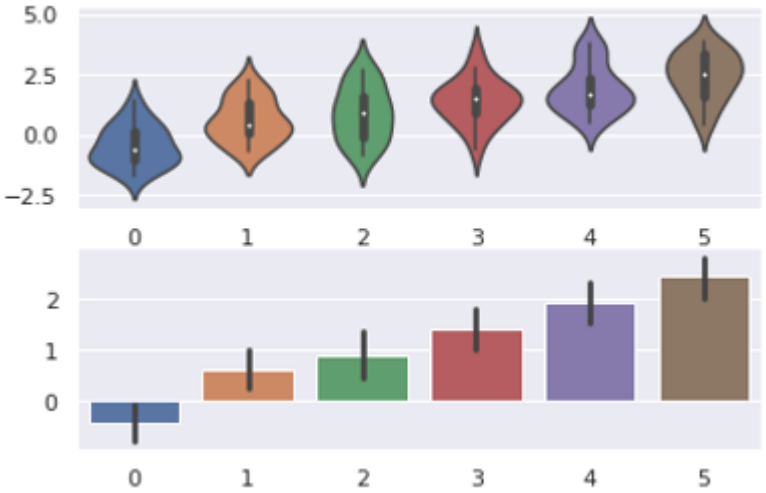
Plot Aesthetics

Style Management

❖ 전체 Style을 변경하여 지속적으로 사용하고 싶다면, 아래와 같은 절차를 거치면 된다.

```
sns.axes_style()

# 배경 스타일을 darkgrid로 적용하고 투명도를 0.9로
sns.set_style("darkgrid", {"axes.facecolor": "0.9"})

# 혹은 간단하게 darkgrid만 적용하고 싶다면,
sns.set(style="darkgrid")
```

```
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2

with sns.axes_style("darkgrid"):
    plt.subplot(211)
    sns.violinplot(data=data)
    plt.subplot(212)
    sns.barplot(data=data)
    plt.show()
```



Color Management

❖ 현재의 Color Palette를 확인하고 싶다면 다음과 같이 코드를 입력하면 된다.

```
current_palette = sns.color_palette()
sns.palplot(current_palette)
```

❖ Palette 기본적인 테마는 총 6개가 있다.

deep, muted, pastel, bright, dark, colorblind

❖ color_palette 메서드를 통해 palette를 바꿀 수 있다.

❖ sns.color_palette(palette=None, n_colors=None) :: color palette를 정의하는 색깔 list를 반환함

palette = [string], Palette 이름

n_colors = [Integer]

Color Management

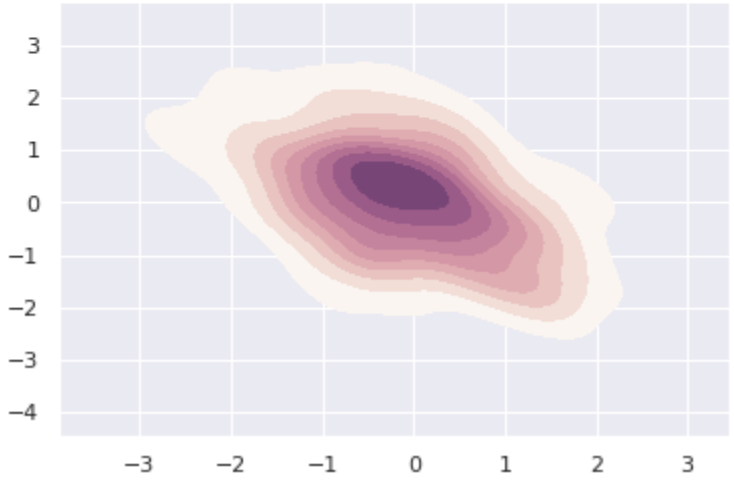
- Palette에는 위에서 본 6가지 기본 테마 외에도, hls, husl, Set1, Blues_d, RdBu 등 수많은 matplotlib palette를 사용할 수 있다.
- xkcd를 이용하여 이름으로 색깔을 불러올 수도 있다.

```
colors = ["windows blue", "amber", "greyish", "faded green", "dusty purple"]
sns.palplot(sns.xkcd_palette(colors))
```

❖ cubehelix palette

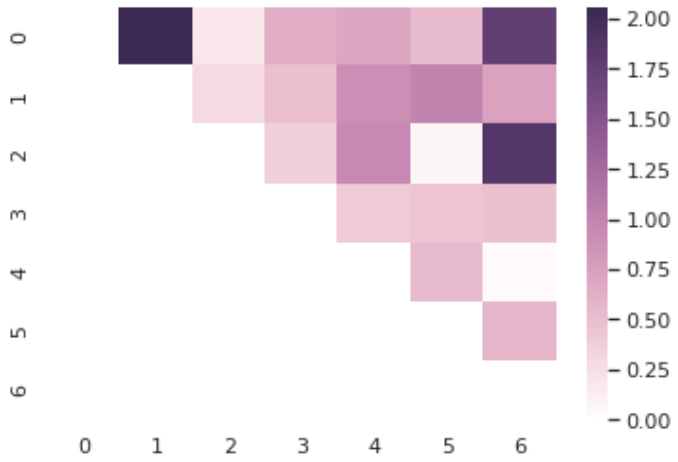
```
cubehelix_palette = sns.cubehelix_palette(8, start=2, rot=0.2, dark=0, light=.95,
                                          reverse=True, as_cmap=True)
x, y = np.random.multivariate_normal([0, 0], [[1, -0.5], [-0.5, 1]], size=300).T
cmap = sns.cubehelix_palette(dark=0.3, light=1, as_cmap=True)
sns.kdeplot(x, y, cmap=cmap, shade=True)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd9b3c2090>
```



❖ cubeleix palette를 이용하여 heatmap을 그리는 방법

```
arr = np.array(np.abs(np.random.randn(49))).reshape((7,7))
mask = np.tril_indices_from(arr)
arr[mask] = False

palette = sns.cubehelix_palette(n_colors=7, start=0, rot=0.3,
                                light=1.0, dark=0.2, reverse=False, as_cmap=True)
sns.heatmap(arr, cmap=palette)
plt.show()
```



개요

❖ **Seaborn의 Plotting 메서드들 중 가장 중요한 위치에 있는 메서드들은 아래와 같다.**

메서드	기능	종류
relplot	2개의 연속형 변수 사이의 통계적 관계를 조명	scatter, line
catplot	범주형 변수를 포함하여 변수 사이의 통계적 관계를 조명	swarm, strip, box, violin, bar, point

❖ **데이터의 분포를 그리기 위해서는 distplot, kdeplot, jointplot 등을 사용할 수 있다.**

Visualizing statistical relationships

❖ sns.relplot(x, y, kind, hue, size, style, data, row, col, col_wrap, row_order, col_order, palette, ...) :: 2개의 연속형 변수 사이의 통계적 관계를 조명함, 각종 옵션으로 추가적으로 변수를 삽입할 수도 있음

- ✓ hue, size, style = [string], 3개의 변수를 더 추가할 수 있음
- ✓ col = [string], 여러 그래프를 한 번에 그릴 수 있게 해줌. 변수 명을 입력하면 됨
- ✓ kind = [string], scatter 또는 line 입력

Visualizing statistical relationships

1. Scatter plot

```
tips = sns.load_dataset('tips')
tips.head(6)
```

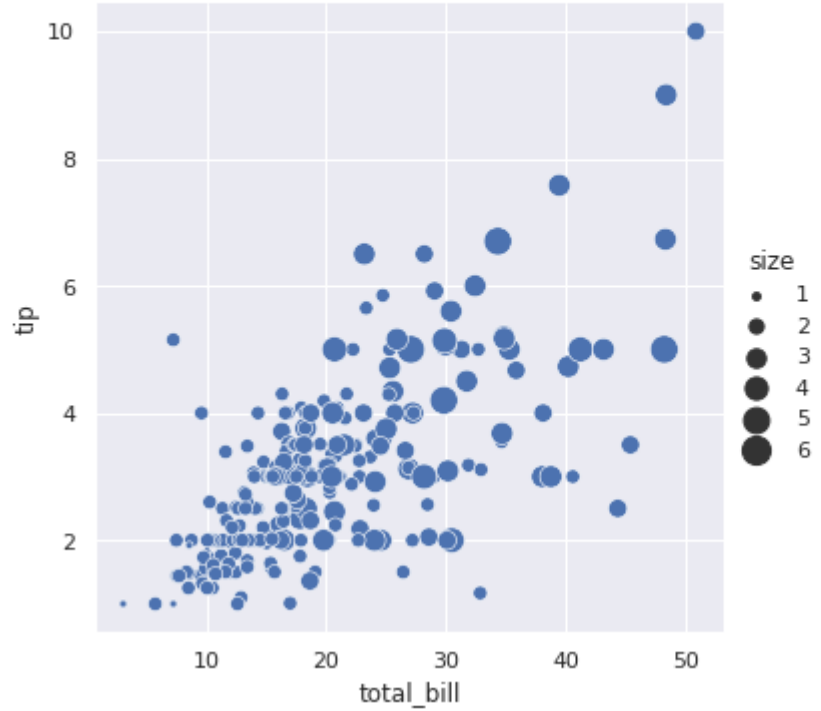
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4

Visualizing statistical relationships

1. Scatter plot

```
# 3.1.1. Scatter plot
sns.relplot(x="total_bill", y="tip", size="size", sizes=(15, 200), data=tips)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd91d27e10>
```

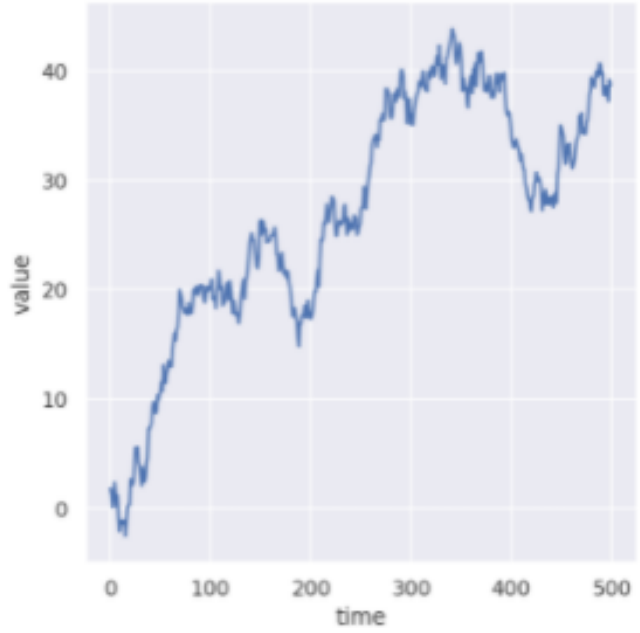


Visualizing statistical relationships

2. Line plot
일반적인 Line Plot

```
import pandas as pd
# 3,1,2, Line plot
df = pd.DataFrame(dict(time=np.arange(500),
                       value=np.random.randn(500).cumsum()))
sns.relplot(x="time", y="value", kind="line", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fb1b0a8b090>
```



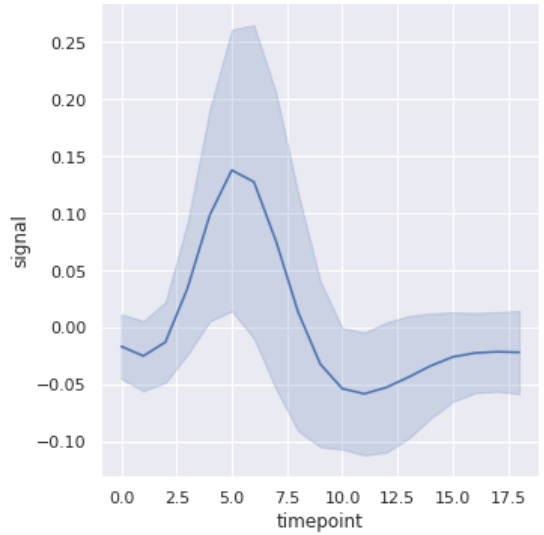
Visualizing statistical relationships

2. Line plot

같은 x 값에 여러 y가 존재할 때 (Aggregation) , Confidence Interval이 추가된다. 만약 이를 제거하고 싶으면, argument에 ci=None을 추가하면 되며, 만약 ci="sd"로 입력하면, 표준편차가 표시

```
# seaborn이 제공하는 데이터 프레임 불러오기
fmri = sns.load_dataset("fmri")

sns.relplot(x="timepoint", y="signal", kind="line", data=fmri, ci="sd")
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd917c2690>
```

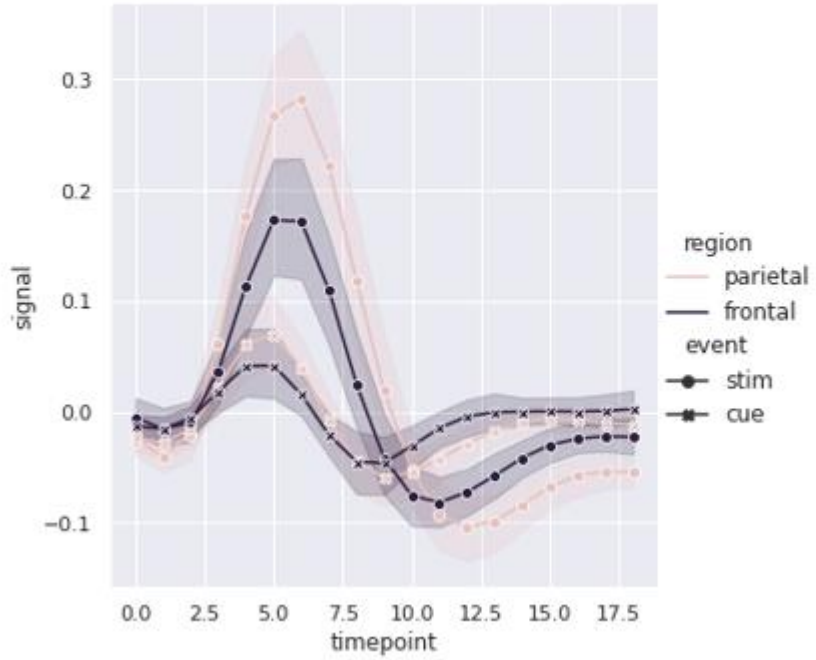


Visualizing statistical relationships

2. Line plot
여러 그래프 한 번에 그리기

```
pal = sns.cubehelix_palette(light=0.8, n_colors=2)
sns.relplot(x="timepoint", y="signal", hue="region", style="event",
            palette=pal, dashes=False, markers=True, kind="line", data=fmri)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd91767750>
```



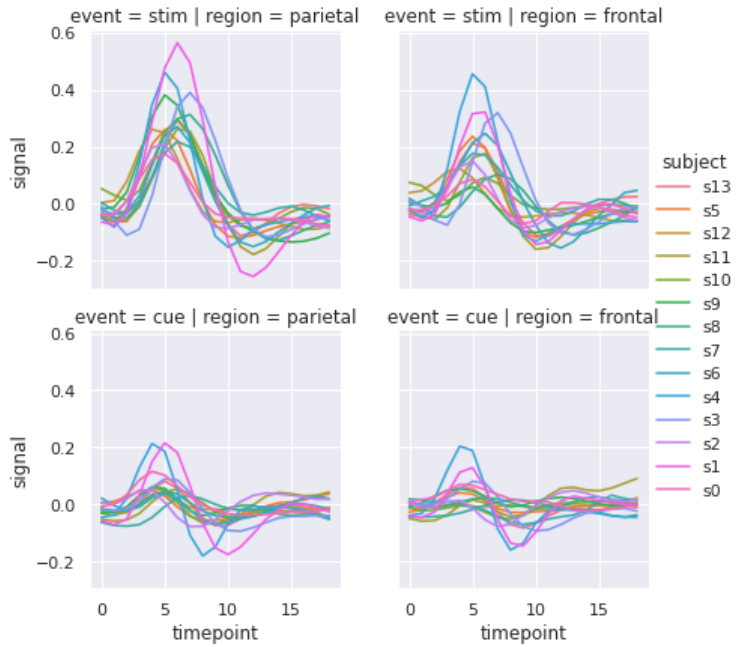
Visualizing statistical relationships

2. Line plot

여러 그래프 한 번에 그리기, col에 지정된 변수 내 값이 너무 많으면, col_wrap[integer]을 통해 한 행에 나타낼 그래프의 수를 조정할 수 있다.

```
sns.relplot(x="timepoint", y="signal", hue="subject", col="region",
            row="event", height=3, kind="line", estimator=None, data=fmri)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd9167c650>
```



Plotting with categorical data

❖ 범주형 변수를 포함한 여러 변수들의 통계적 관계를 조명하는 catplot은 kind=swarm, strip, box, violin, bar, point 설정을 통해 다양한 그래프를 그릴 수 있게 해준다.

❖ sns.catplot(x, y, kind, hue, data, row, col, col_wrap, order, row_order, col_order, hue_order, palette, ...) :: 범주형 변수를 포함한 여러 변수들의 통계적 관계를 조명함

- ✓ x, y, hue = [string], 그래프를 그릴 변수들의 이름
- ✓ row, col = [string], faceting of the grid를 결정할 범주형 변수의 이름

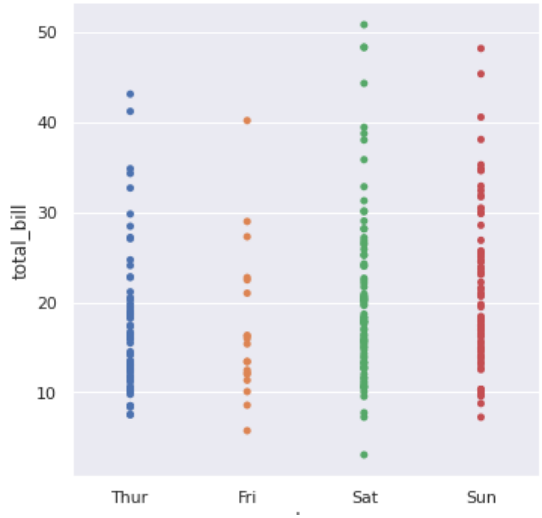
Plotting with categorical data

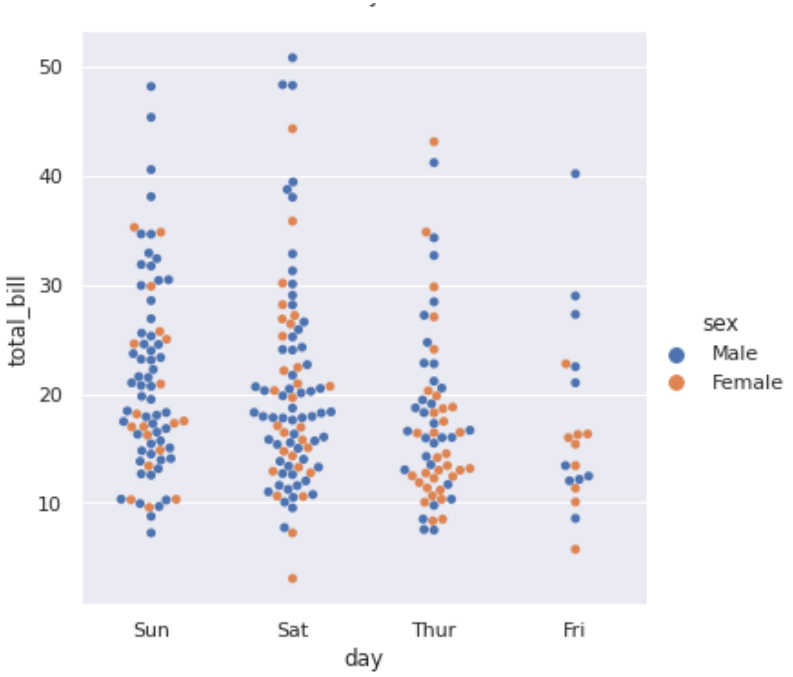
1. Categorical Scatterplots: strip, swarm

❖ swarm 그래프는 그래프 포인트끼리 겹치는 것을 막아준다.
(overlapping 방지) 가로로 그리고 싶으면, x와 y의 순서를 바꿔주면 된다.

```
# 3.2.1. Categorical Scatterplots: strip, swarm

sns.catplot(x="day", y="total_bill", jitter=False, data=tips)
sns.catplot(x="day", y="total_bill", hue="sex", kind="swarm",
            order=["Sun", "Sat", "Thur", "Fri"], data=tips)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd8fcb8d10>
```



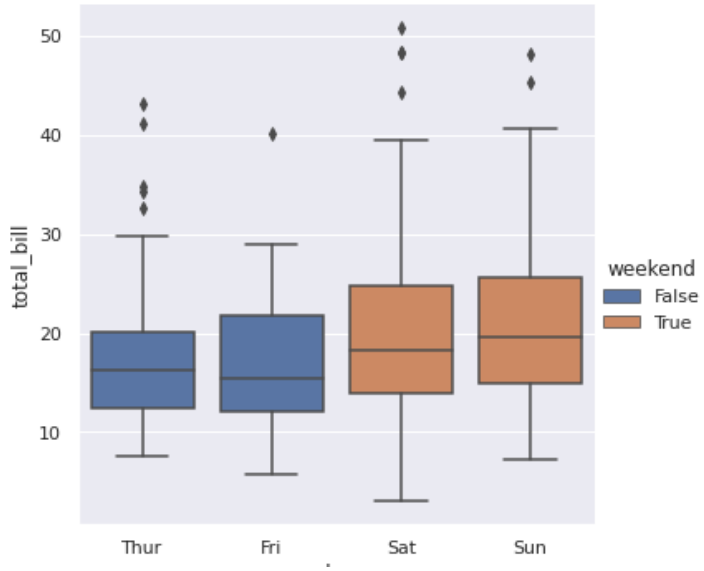


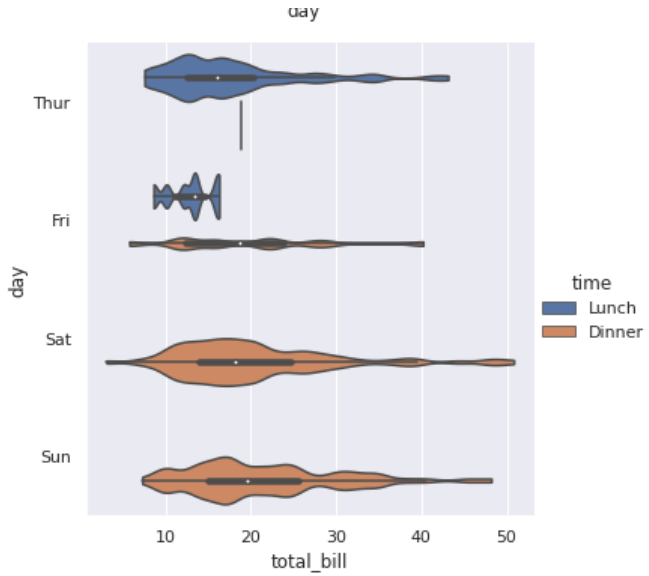
Plotting with categorical data

2. Distribution of observations within categories: box, violin

```
# 3.2.2. Distribution of observations within categories: box, violin

tips["weekend"] = tips["day"].isin(["Sat", "Sun"])

sns.catplot(x="day", y="total_bill", hue="weekend", orient='v',
            kind="box", dodge=False, data=tips, legend=True)
sns.catplot(x="total_bill", y="day", hue="time",
            kind="violin", bw=.15, cut=0, data=tips)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd9a579490>
```





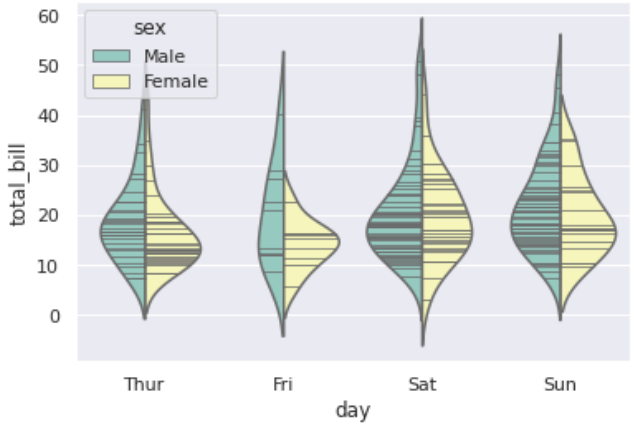
Plotting with categorical data

2. Distribution of observations within categories: box, violin

그래프 내부에 선(Inner Stick)을 추가하고 싶거나, Scatter Plot과 Distribution Plot을 동시에 그리고 싶다면 아래의 기능을 사용하면 된다.

```
# Inner Stick 사용
sns.violinplot(x="day", y="total_bill", hue="sex", data=tips,
               split=True, inner="stick", palette="Set3")

# 결합: 분포와 실제 데이터까지 한번에 보여주는 방법
g = sns.catplot(x="day", y="total_bill", kind="violin", inner=None, data=tips)
sns.swarmplot(x="day", y="total_bill", color="k", size=3, data=tips, ax=g.ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd8fba4c90>
```

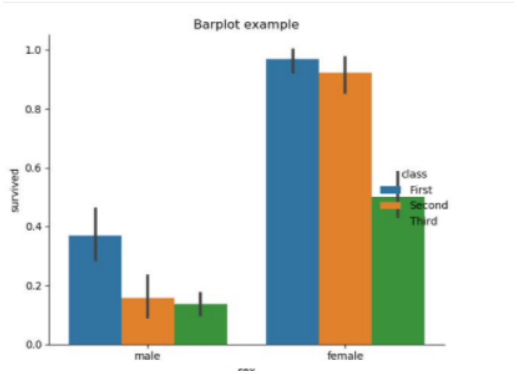


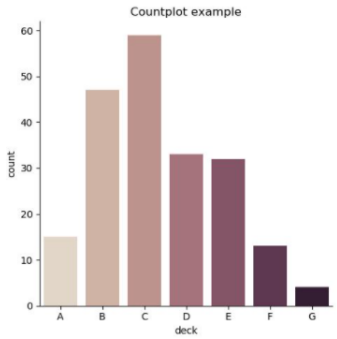
Plotting with categorical data

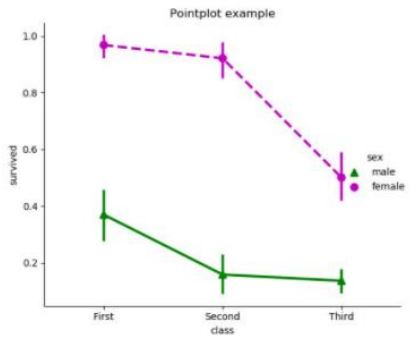
3. Statistical Estimation within categories: barplot, countplot, pointplot

기본적인 Barplot, Countplot, Pointplot을 그리는 방법

```
# 3.2.3. Statistical Estimation within categories: barplot, countplot, pointplot

titanic = sns.load_dataset("titanic")

# bar
sns.catplot(x="sex", y="survived", hue="class", kind="bar", data=titanic)

# 그냥 count를 세고 싶다면
sns.catplot(x="deck", kind="count", palette="ch:.25", data=titanic)

# point
sns.catplot(x="class", y="survived", hue="sex", data=titanic,
            palette={"male": "g", "female": "m"}, kind="point",
            markers=["^", "o"], linestyles=["-", "--"])
```





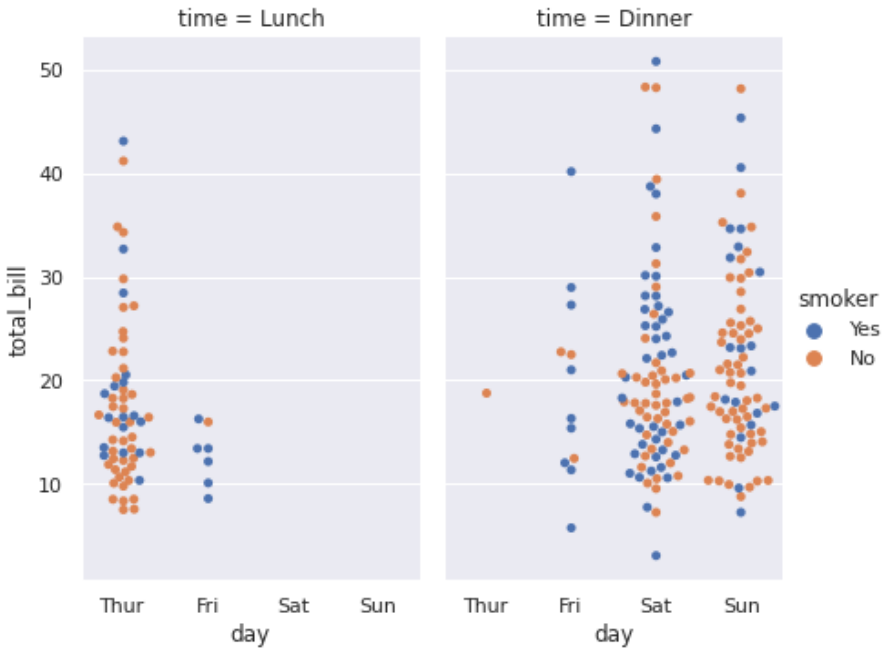


Plotting with categorical data

4. Showing multiple relationships with facets

```
# 3.2.4. Showing multiple relationships with facets
# catplot 역시 relplot 처럼 col argument를 사용해 여러 그래프를 그릴 수 있음
sns.catplot(x="day", y="total_bill", hue="smoker",
            col="time", aspect=0.6, kind="swarm", data=tips)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fdd8f615290>
```



Visualizing the distribution of a dataset

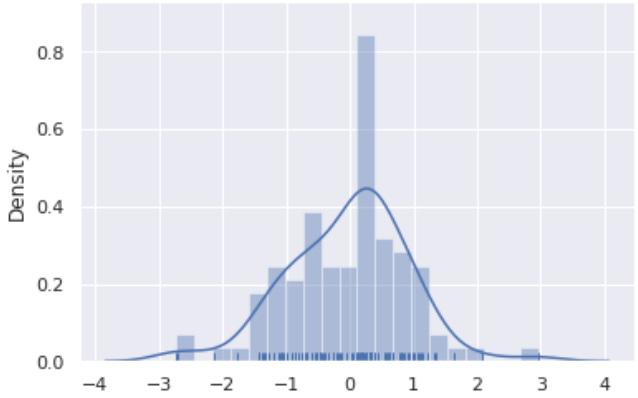
1. 일변량 분포

sns.distplot(a, bins, hist=True, rug=False, fit=None, color=None, vertical=False, norm_hist=False, axlabel, label, ax ...):: 관찰 값들의 일변량 분포를 그림

- ✓ a = [Series, 1d array, list], Observed data이며, Series에 name 속성이 있다면 이것이 label로 사용될 것임
- ✓ hist = [bool], True면 히스토그램을 그림

```
# 3.2.1. 일변량 분포
x = np.random.normal(size=100)
sns.distplot(x, hist=True, bins=20, kde=True, rug=True)
```

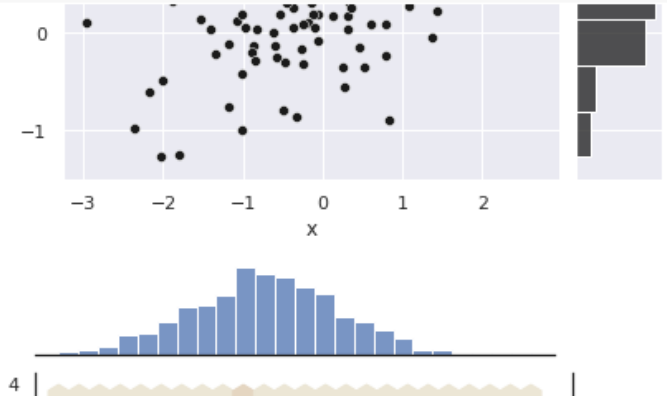
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: Fu
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2056: Fu
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd8f507bd0>
```



Visualizing the distribution of a dataset

2. 이변량 분포

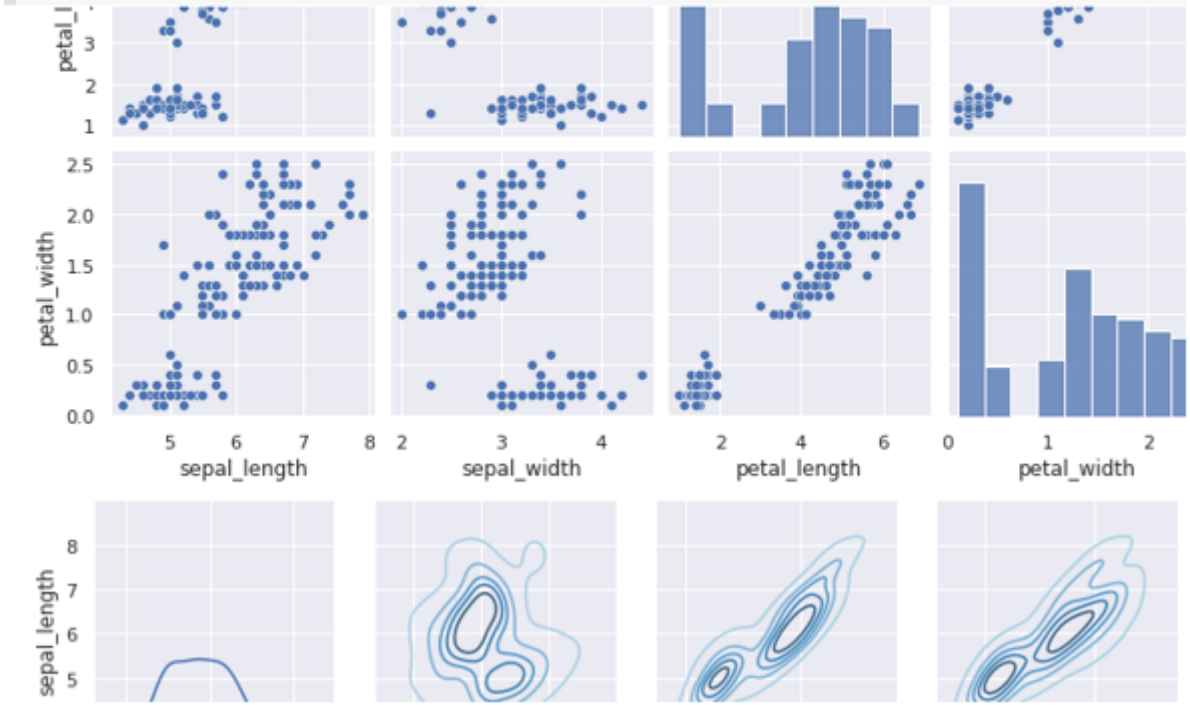
```
# 3.2.2. 이변량 분포
# Prep
mean, cov = [0, 1], [(1, .5), (.5, 1)]
data = np.random.multivariate_normal(mean, cov, 200)
df = pd.DataFrame(data, columns=["x", "y"])
x, y = np.random.multivariate_normal(mean, cov, 1000).T
cmap = sns.cubehelix_palette(as_cmap=True, start=0, rot=0.5, dark=0, light=0.9)

# Scatter plot
sns.jointplot(x="x", y="y", data=df, color='k')

# Hexbin plot
with sns.axes_style("white"):
    sns.jointplot(x=x, y=y, kind="hex", cmap=cmap)

# Kernel Density Estimation
sns.jointplot(x="x", y="y", data=df, kind="kde", cmap=cmap)
```



Visualizing the distribution of a dataset

3. Pairwise 관계 시각화

```
# 3.2.3. Pairwise 관계 시각화
iris = sns.load_dataset('iris')
sns.pairplot(iris)

g = sns.PairGrid(iris)
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.kdeplot, cmap="Blues_d", n_levels=6);
```



2

Iris를 통한 시각화 연습

1. 시각화 기초 연습

학습목표

Iris를 통한 시각화 연습

Unit 1

시각화 기초 연습

라이브러리

❖ 주피터 노트북에서 시각화를 할 때는 항상 아래 명령을 입력해서 그래프가 노트북 안에 포함되도록 한다.

```
%matplotlib inline
```

❖ seaborn은 Python 시각화에 가장 많이 사용되는 라이브러리 중에 하나이다. seaborn은 matplotlib를 기반으로 한다.

```
import seaborn as sns
from matplotlib import pyplot as plt
```

iris

❖ 실습에 사용할 데이터는 iris 데이터이다. iris 데이터는 seaborn에 내장되어 있어 아래 명령으로 불러올 수 있다.

```
df = sns.load_dataset('iris')
```

❖ 위의 명령이 안될 경우 https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv 에서 다운 받은 다음 pandas로 직접 읽어들인다.

```
import pandas
df = pandas.read_csv('iris.csv')
```

❖ iris는 150 송이 붓꽃의 데이터이다. sepal은 꽃잎, petal은 꽃받침, species는 품종을 뜻한다.

iris

❖ 데이터를 불러왔으면 데이터의 내용을 확인해보자.

```
df = sns.load_dataset('iris')
df.head()
```

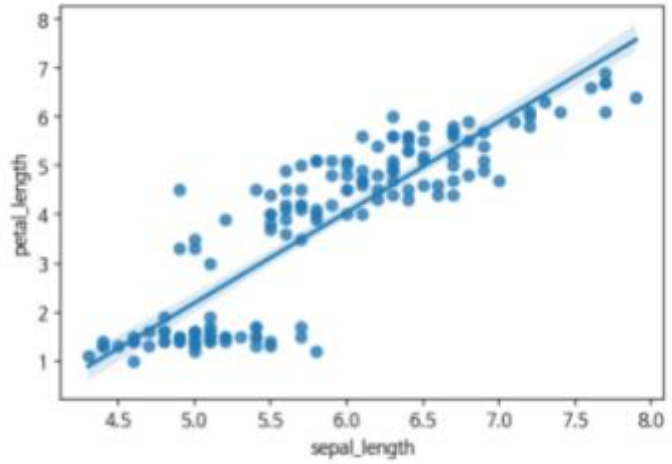
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

산점도

❖ 꽃잎의 길이와 꽃받침의 길이를 산점도로 나타내보자.

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f3375b950>
```



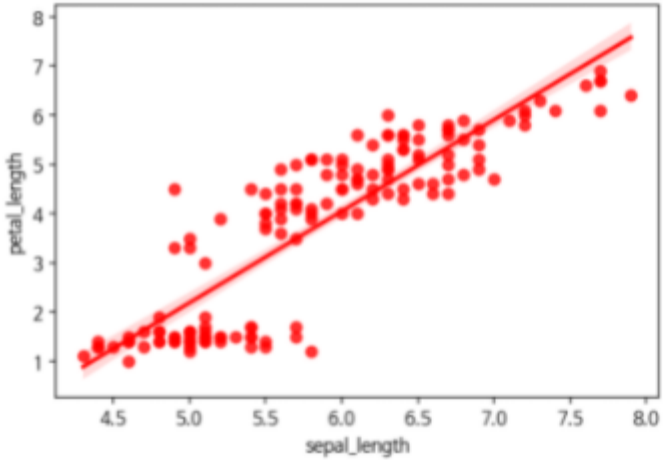
색상

❖ 색상을 빨간색으로 바꿔보자.

sns.regplot(x=df["sepal_length"], y=df["petal_length"], color='red')

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"], color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f3372d490>
```



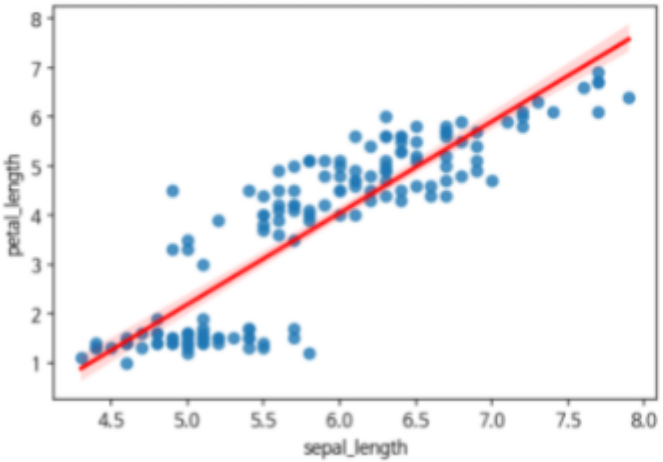
색상

❖ 선의 속성만 바꾸려면 line_kws에 설정값을 사전 형태로 넘겨준다.

sns.regplot(x=df["sepal_length"], y=df["petal_length"], line_kws={'color': 'red'})

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"], line_kws={'color': 'red'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f336798d0>
```



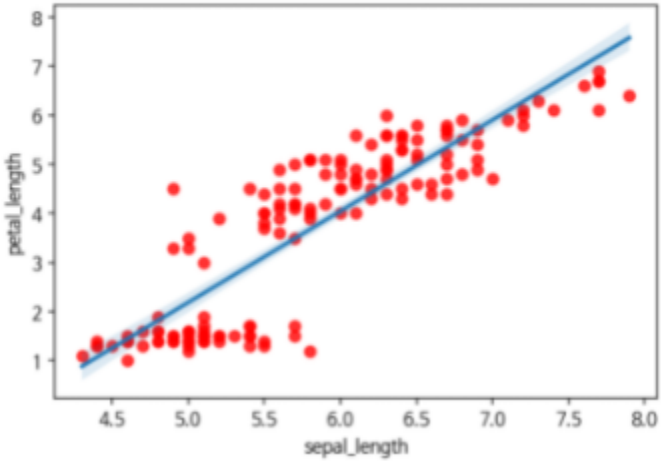
색상

❖ 점의 속성만 바꾸려면 scatter_kws에 설정값을 사전 형태로 넘겨준다.

sns.regplot(x=df["sepal_length"], y=df["petal_length"], scatter_kws={'color': 'red'})

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"], scatter_kws={'color': 'red'})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f33589c10>
```



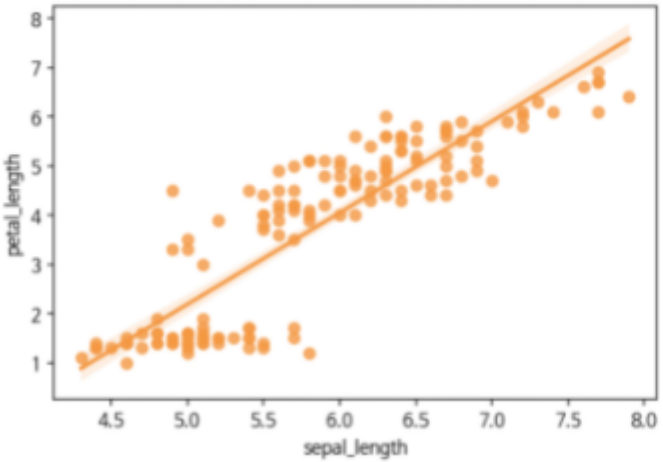
색상

❖ 색상에는 색상 코드를 사용할 수 있다. 색상 코드는 구글에서 "color picker"를 검색하자.

sns.regplot(x=df["sepal_length"], y=df["petal_length"], color='#f49842')

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"], color='#f49842')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f335bf890>
```



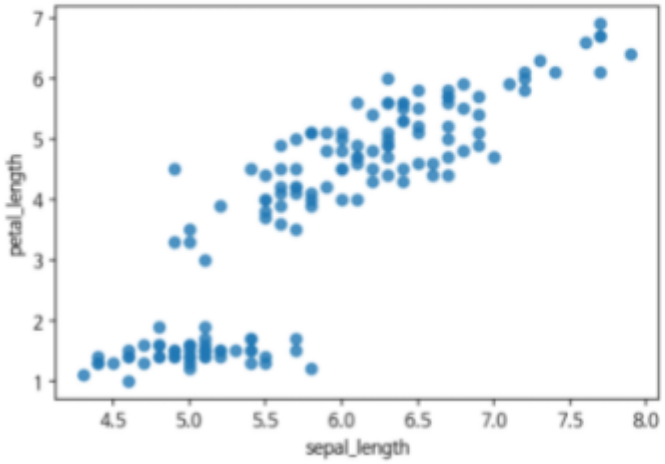
선 없애기

❖ fit_reg 옵션으로 회귀선을 제거할 수 있다.

sns.regplot(x=df["sepal_length"], y=df["petal_length"], fit_reg=False)

```
sns.regplot(x=df["sepal_length"], y=df["petal_length"], fit_reg=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9f334f6350>
```



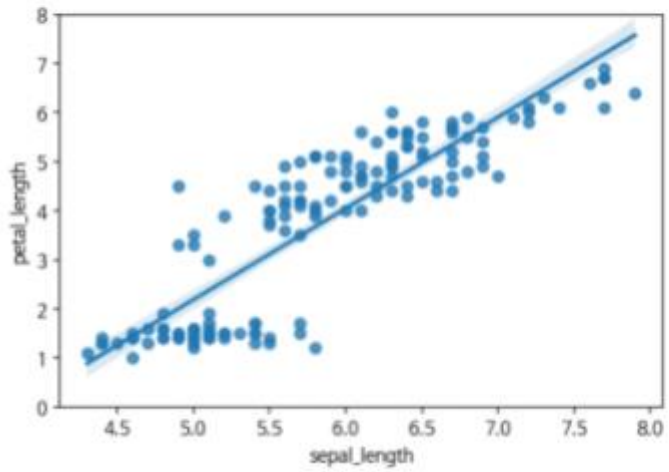
그래프의 속성 바꾸기

- matplotlib에는 피겨(figure)와 액시즈(axes)라는 개념이 있다. 피겨는 그림을 가리키고, 액시즈는 그림에 포함된 하나의 그래프를 가리킨다.
- seaborn의 그래프 함수들은 해당 그래프의 액시즈를 반환한다. 만약 그래프의 속성을 바꾸고 싶으면 해당 액시즈를 이용한다.

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_ylim([0, 8])
```

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_ylim([0, 8])
```

```
(0.0, 8.0)
```



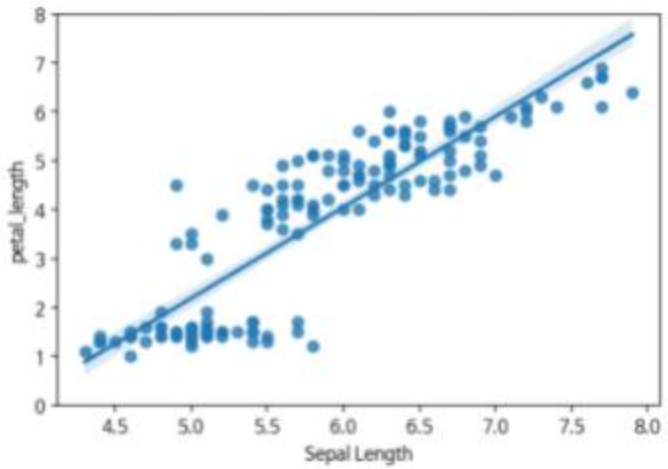
그래프의 속성 바꾸기

❖ 축 이름 붙이기

❖ 축 이름은 set_xlabel과 set_ylabel로 붙인다.

ax.set_xlabel('Sepal Length')
ax.figure # 그림을 현재 셀에서 다시 보여준다

```
ax.set_xlabel('Sepal Length')
ax.figure  # 그림을 현재 셀에서 다시 보여준다
```

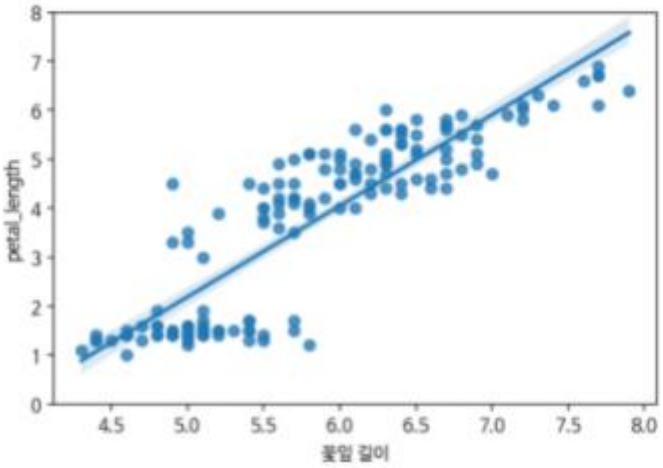


그래프의 속성 바꾸기

❖ 한글을 사용하면 글자가 깨진다.

ax.set_xlabel('꽃잎 길이')
ax.figure

```
ax.set_xlabel('꽃잎 길이')
ax.figure
```



글꼴 이름 찾기

❖ 글꼴 이름이 무엇으로 되어있는지 모르는 경우 fontManager를 사용한다.

```
from matplotlib.font_manager import fontManager
```

❖ 아래와 같이 글꼴 이름에 Nanum이 들어가는 경우를 찾는다.

```
for font in fontManager.ttflist:
    if 'Nanum' in font.name:
        print(font.name)
```

```
from matplotlib.font_manager import fontManager
```

```
for font in fontManager.ttflist:
    if 'Nanum' in font.name:
        print(font.name)
```

```
NanumSquareRound
NanumMyeongjo
NanumSquare
NanumMyeongjo
NanumSquare
NanumGothic
NanumSquareRound
NanumBarunGothic
NanumGothic
NanumBarunGothic
```

글꼴 설정

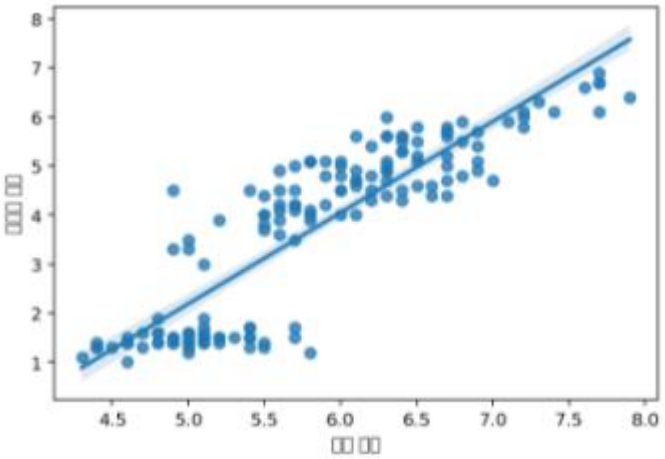
❖ 다음과 같이 글꼴을 한글 글꼴로 바꿔준다.

```
plt.rc('font', family='Malgun Gothic')
```

❖ 다시 그래프를 그리면 한글이 잘 표시된다.

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_xlabel('꽃잎 길이')
ax.set_ylabel('꽃받침 길이')
```

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_xlabel('꽃잎 길이')
ax.set_ylabel('꽃받침 길이')
```



글꼴 설정

❖ 위의 그래프를 자세히보면 왼쪽 아래 마이너스 표시가 깨지는 것을 볼 수 있다. 이것은 한글 폰트에 마이너스가 없기 때문에 생기는 현상이다. 다음과 같이 해주면 된다.

```
plt.rc('font', family='Malgun Gothic')
plt.rc('axes', unicode_minus=False)
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_xlabel('꽃잎 길이')
ax.set_ylabel('꽃받침 길이')
```

```
plt.rc('font', family='NanumGothic')
plt.rc('axes', unicode_minus=False)
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.set_xlabel('꽃잎 길이')
ax.set_ylabel('꽃받침 길이')
```

```
Text(0, 0.5, '꽃받침 길이')
```

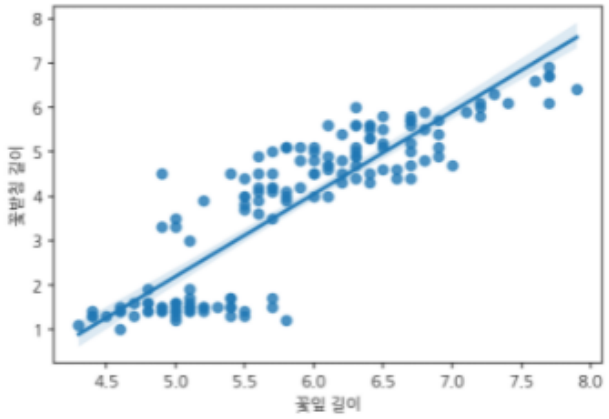
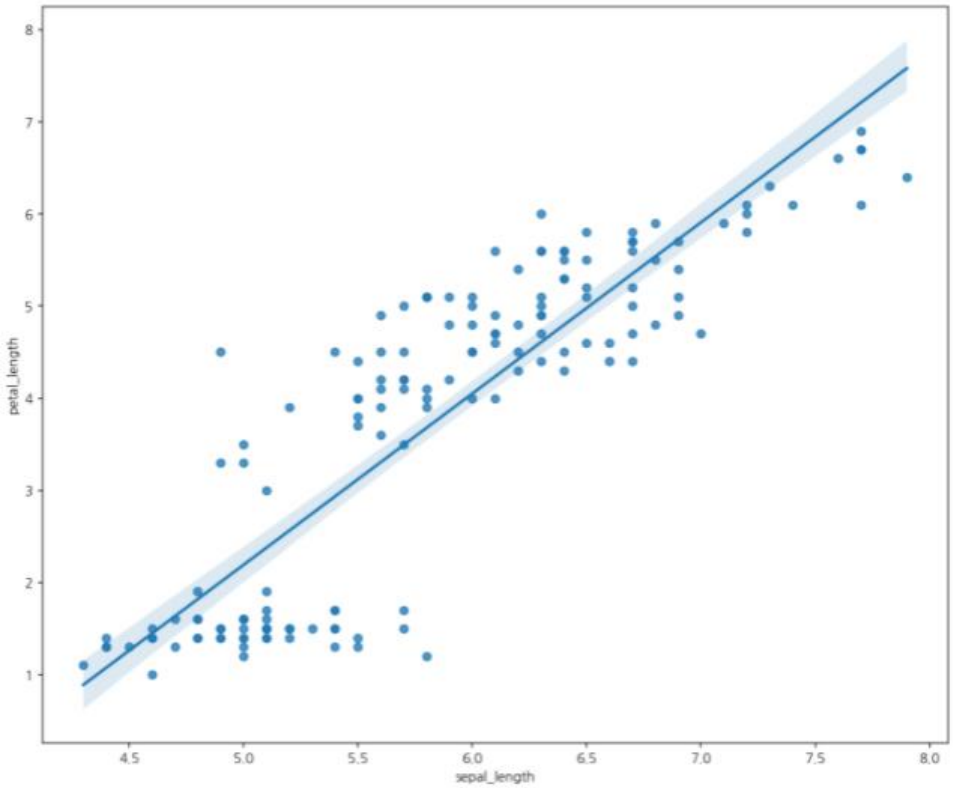


그림 속성 바꾸기

❖ ax.figure를 통해 그림 속성을 바꿀 수도 있다.

ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.figure.set_size_inches(12, 10)

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.figure.set_size_inches(12, 10)
```

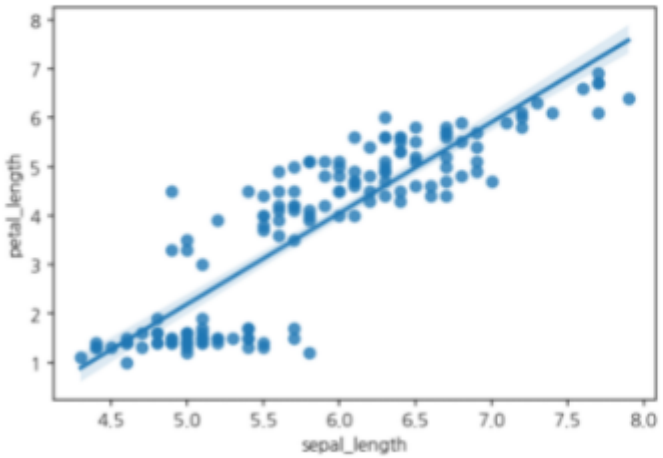


저장

❖ 그림을 저장.

ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.figure.savefig('lm.png')

```
ax = sns.regplot(x=df["sepal_length"], y=df["petal_length"])
ax.figure.savefig('lm.png')
```



Unit A

참고자료

참고자료

- http://www.ncs.go.kr
- NELLDALE/JOHN LEWIS지음, 조영석/김대경/박찬영/송창근 역, 단계별로 배우는컴퓨터과학, 홍릉과학출판사,2018
- 혼자 공부하는 머신러닝+딥러닝 박해선 지음 | 한빛미디어 | 2020년 12월
- 머신러닝 실무 프로젝트 ,아리가 미치아키, 나카야마 신타, 니시바야시 다카시 지음 | 심효섭 옮김 | 한빛미디어 | 2018년 06월
- 파이썬을 활용한 머신러닝 쿡북 크리스 알본 지음 | 박해선 옮김 | 한빛미디어 | 2019년 09월
- 처음 배우는 머신러닝 김의중 지음 | 위키북스 | 2016년 07월
- 파이썬으로 배우는 머신러닝의 교과서 : 이토 마코토 지음 | 박광수(아크몬드) 옮김 | 한빛미디어 | 2018년 11월
- 기타 서적 및 웹 사이트 자료 다수 참조

나눔고딕 인스톨

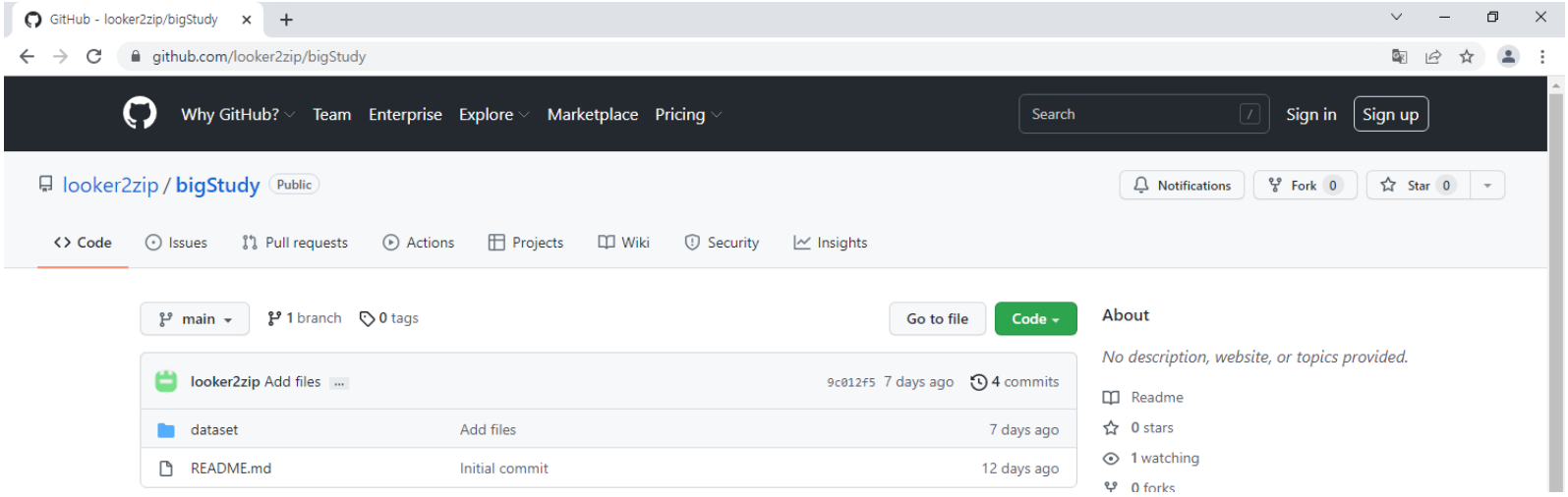
❖ 리눅스 다운로드

!rm -rf bigStudy

!git clone 'https://github.com/looker2zip/bigStudy.git'

❖ 윈도 다운로드

https://github.com/looker2zip/bigStudy



나눔고딕 인스톨

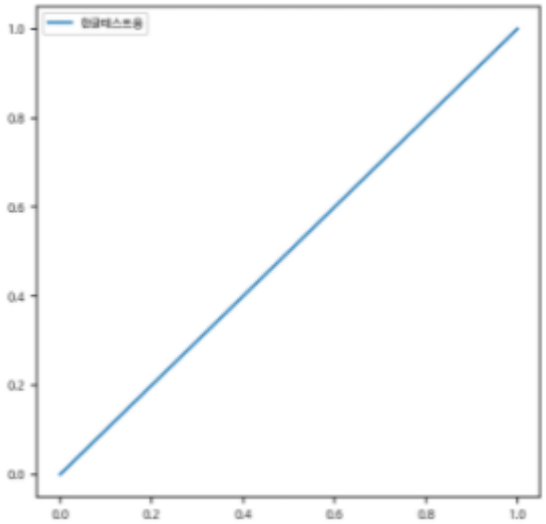
```
%config InlineBackend.figure_format = 'retina'
 !apt -qq -y install fonts-nanum
```

❖ 런타임 다시 시작

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
fontpath = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font = fm.FontProperties(fname=fontpath, size=9)
plt.rc('font', family='NanumBarunGothic')
mpl.font_manager._rebuild()


plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```

나눔고딕 인스톨

```
plt.figure(figsize=(5,5))
plt.plot([0,1], [0,1], label='한글테스트용')
plt.legend()
plt.show()
```



- Mobile: 010-9591-1401
- E-mail: onlooker2zip@naver.com