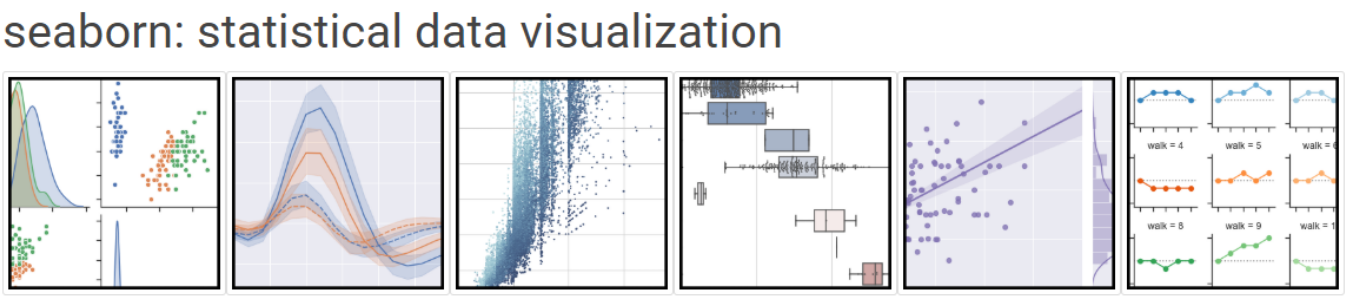
Seaborn 한번에 제대로 배우기



Seaborn 특징

- 여러 변수 간의 관계를 검사하기 위한 데이터 집합 지향 API
- 범주형 변수를 사용하여 관측치 또는 집계 통계량을 표시하기 위한 전문적인 지원
- 일변량 또는 이변량 분포를 시각화하고 데이터의 부분 집합 간 비교하기 위한 옵션
- 서로 다른 종류의 종속 변수에 대한 선형 회귀 모형의 자동 추정 및 표시
- 복잡한 데이터셋의 전체 구조에 대한 편리한 보기
- 복잡한 시각화를 쉽게 구축할 수 있는 다중 플롯 그리드 구조를 위한 높은 수준의 추상화
- 여러 테마가 내장된 matplotlib 그림 스타일링 제어
- 데이터의 패턴을 충실히 나타내는 색상 팔레트 선택 도구

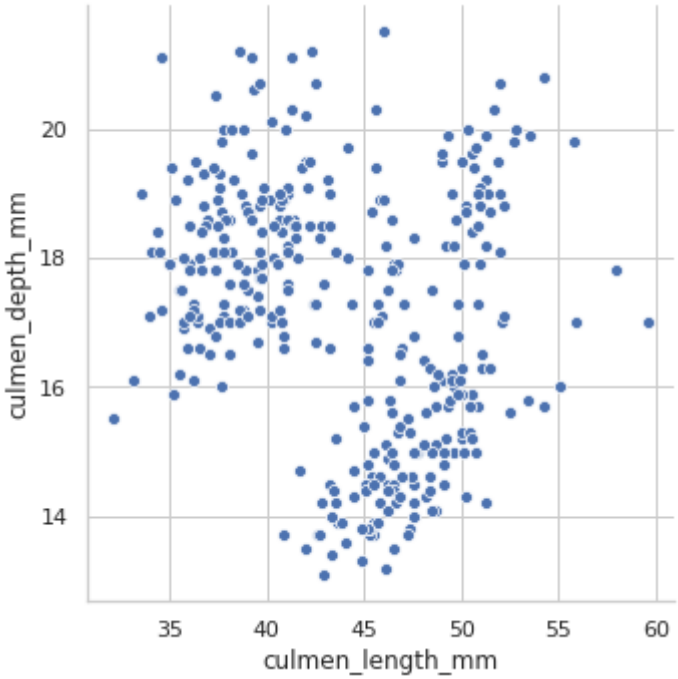
```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

산점도(Scatter Plot)

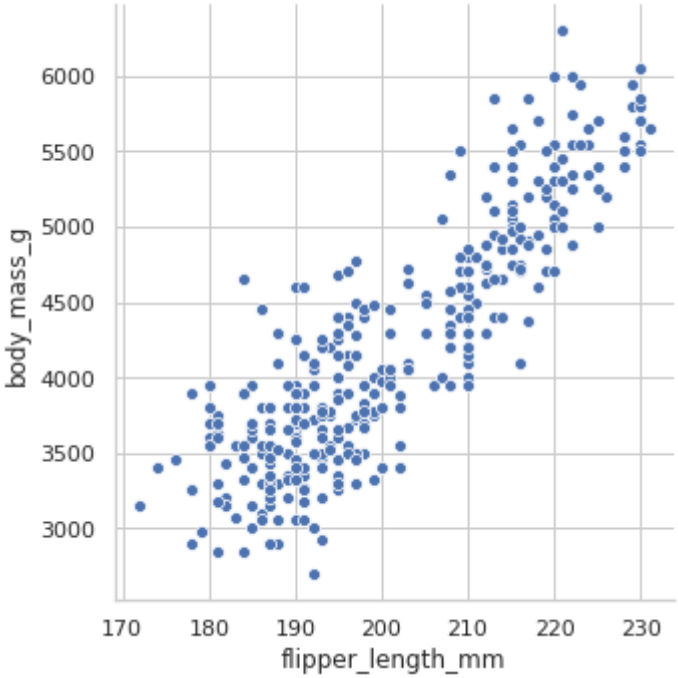
```
penguins = sns.load_dataset("penguins")
penguins
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	

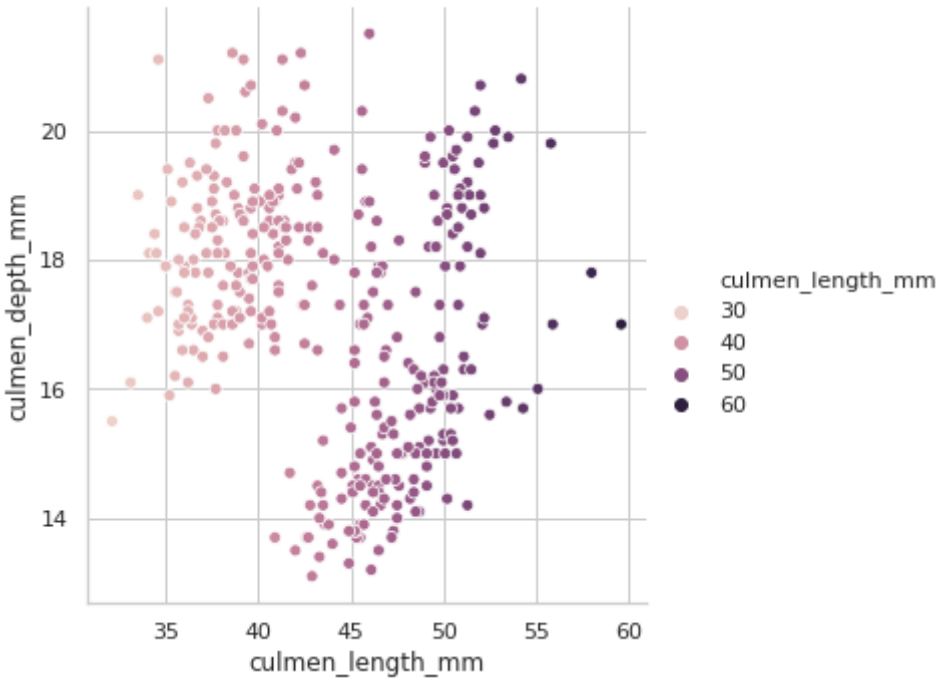
```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm", data=penguins);
```



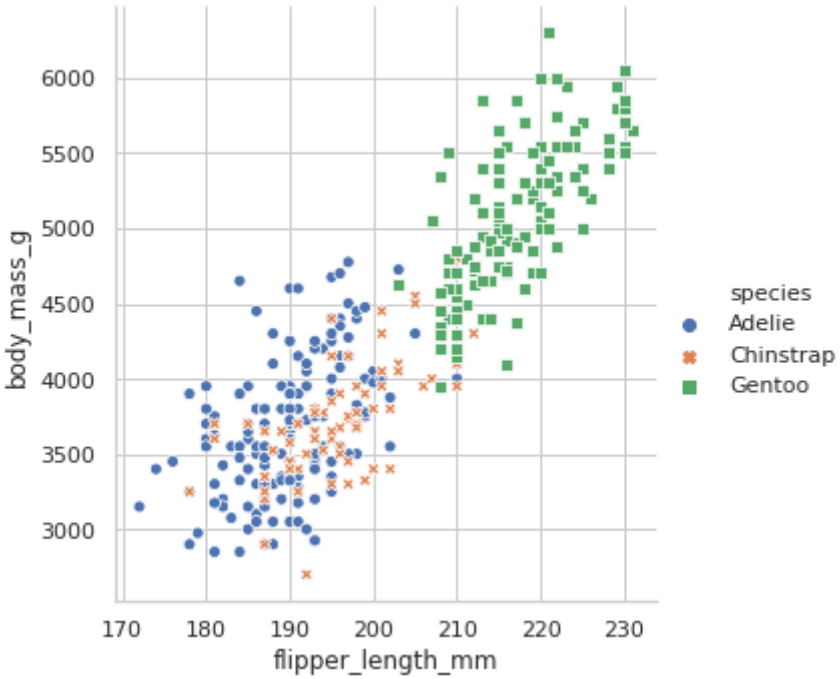

```
sns.relplot(x="flipper_length_mm", y="body_mass_g", data=penguins);
```



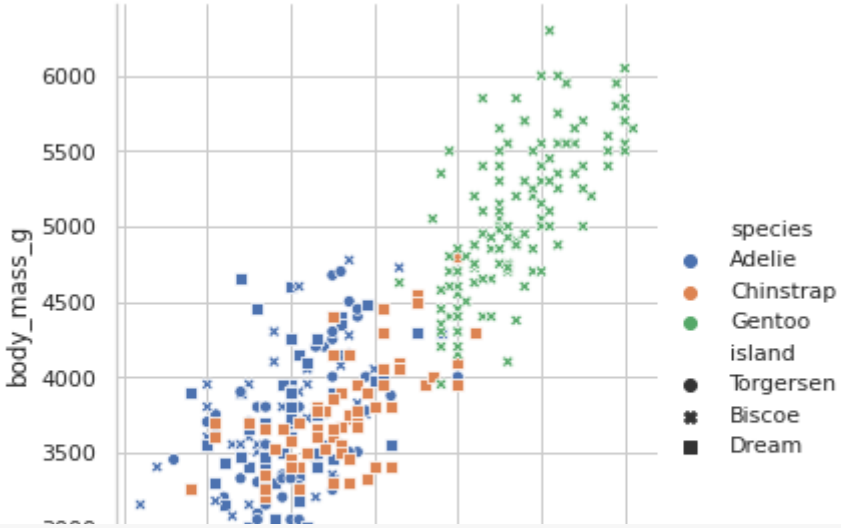
```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="culmen_length_mm", data=penguins);
```



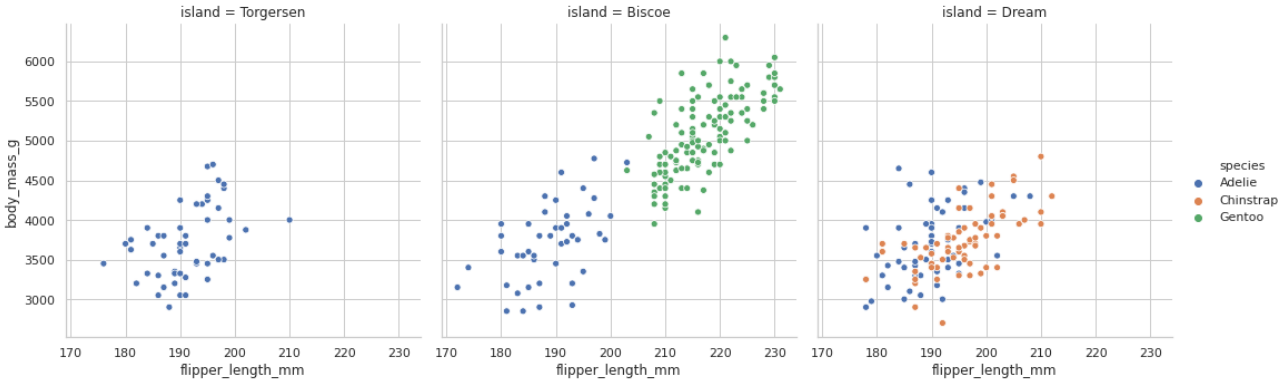

```
sns.relplot(x="flipper_length_mm", y="body_mass_g",
            hue="species", style="species", data=penguins);
```



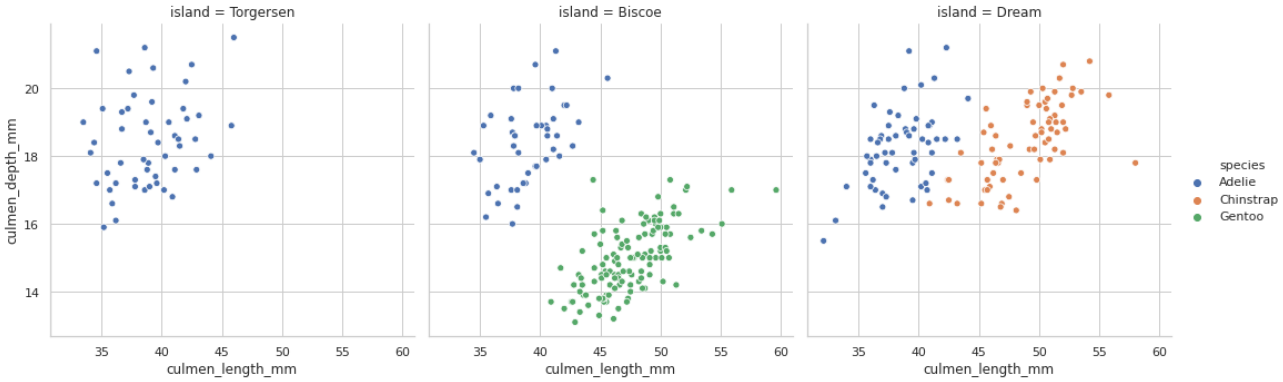

```
sns.relplot(x="flipper_length_mm", y="body_mass_g",
            hue="species", style="island", data=penguins);
```



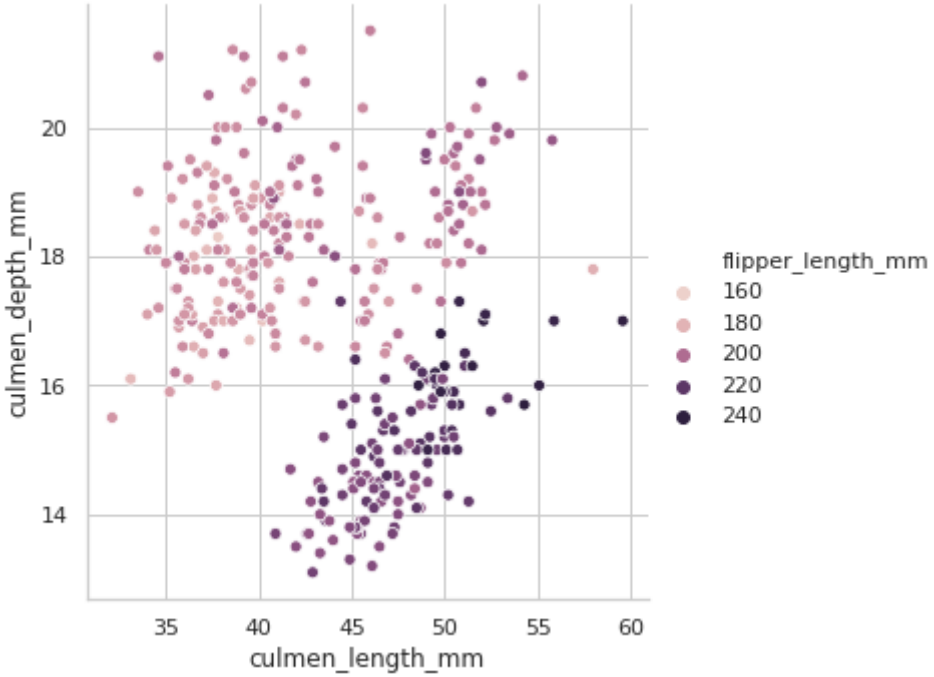

```
sns.relplot(x="flipper_length_mm", y="body_mass_g",
            hue="species", col="island", data=penguins);
```



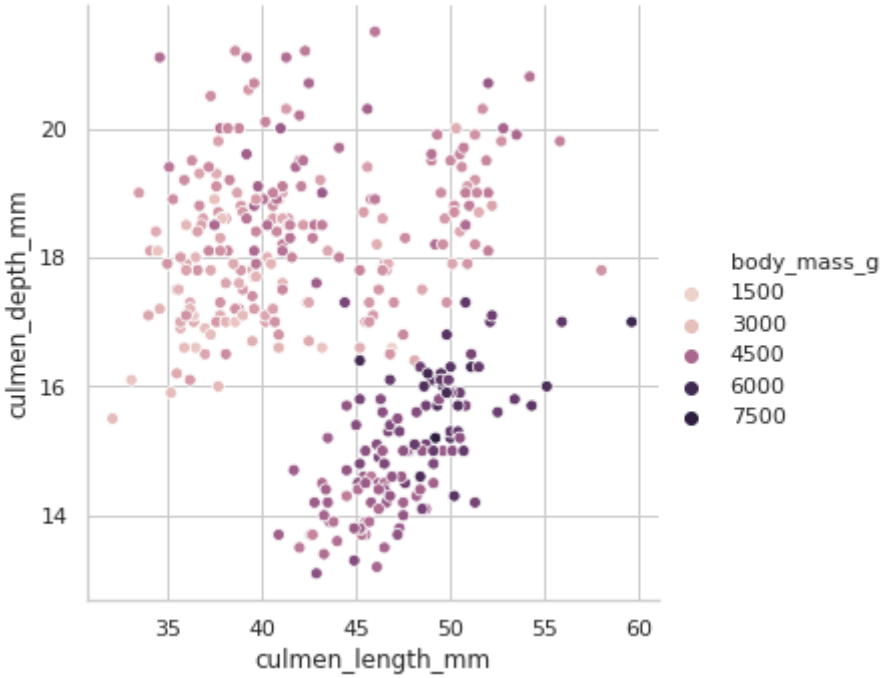

```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="species", col="island", data=penguins);
```



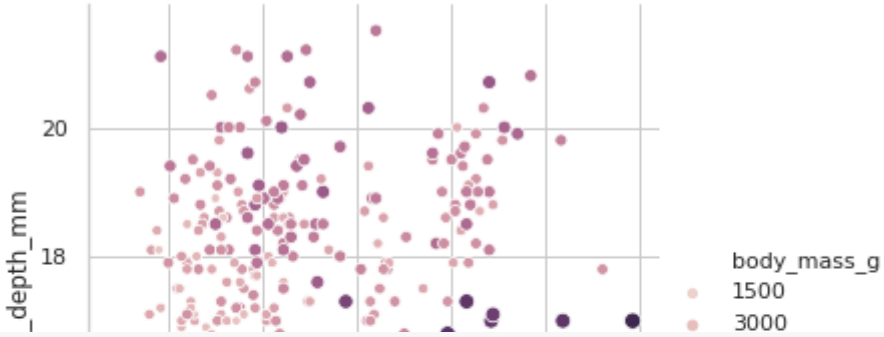

```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="flipper_length_mm", data=penguins);
```




```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="body_mass_g", data=penguins);
```




```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="body_mass_g", size="body_mass_g", data=penguins);
```



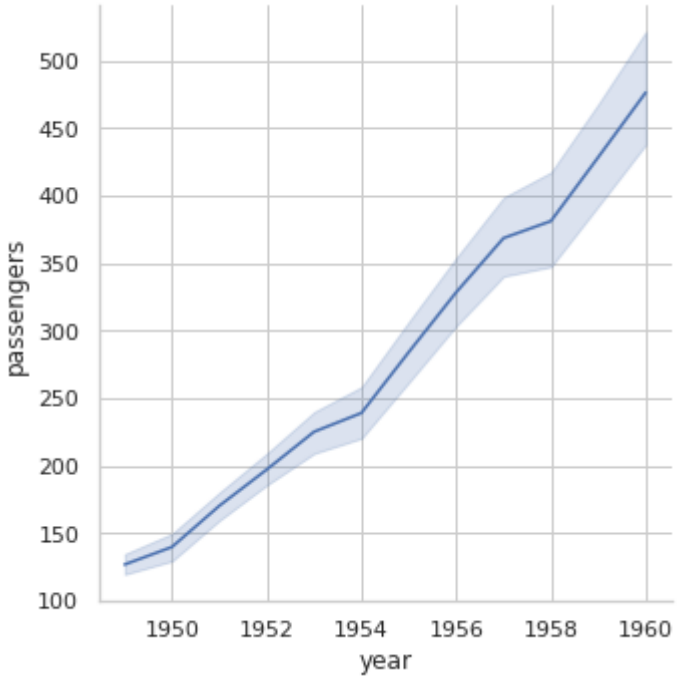

```
sns.relplot(x="culmen_length_mm", y="culmen_depth_mm",
            hue="body_mass_g", size="body_mass_g",
            sizes=(10, 300), data=penguins);
```

라인 플롯(Line Plot)

```
flights = sns.load_dataset("flights")
flights
```

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129

```
sns.relplot(x="year", y="passengers", kind="line", data=flights);
```

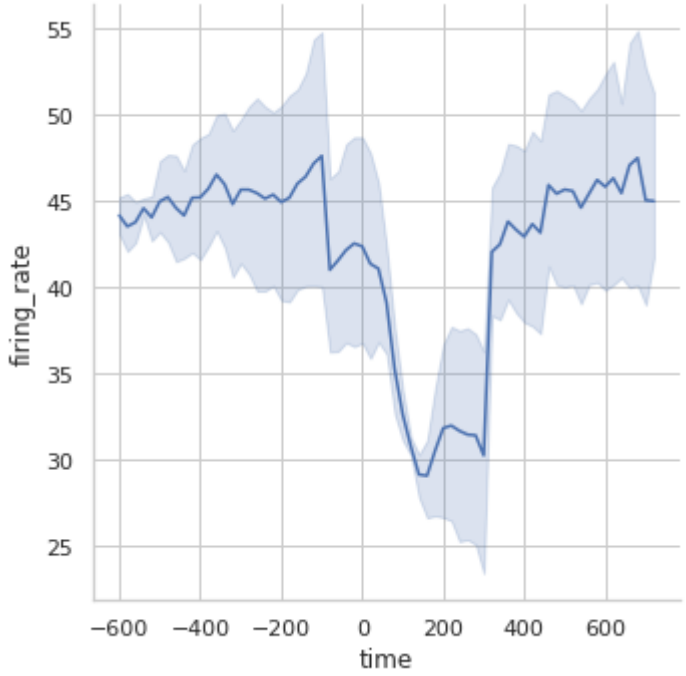



```
dots = sns.load_dataset("dots")
dots
```

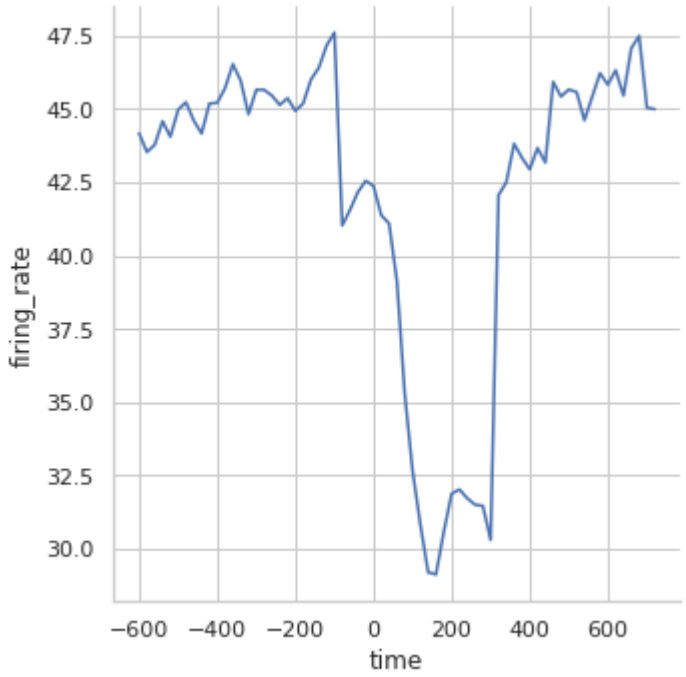
	align	choice	time	coherence	firing_rate
0	dots	T1	-80	0.0	33.189967
1	dots	T1	-80	3.2	31.691726
2	dots	T1	-80	6.4	34.279840
3	dots	T1	-80	12.8	32.631874
4	dots	T1	-80	25.6	35.060487
...	...	...	...	...	...
843	sacc	T2	300	3.2	33.281734
844	sacc	T2	300	6.4	27.583979
845	sacc	T2	300	12.8	28.511530
846	sacc	T2	300	25.6	27.009804
847	sacc	T2	300	51.2	30.959302

848 rows × 5 columns

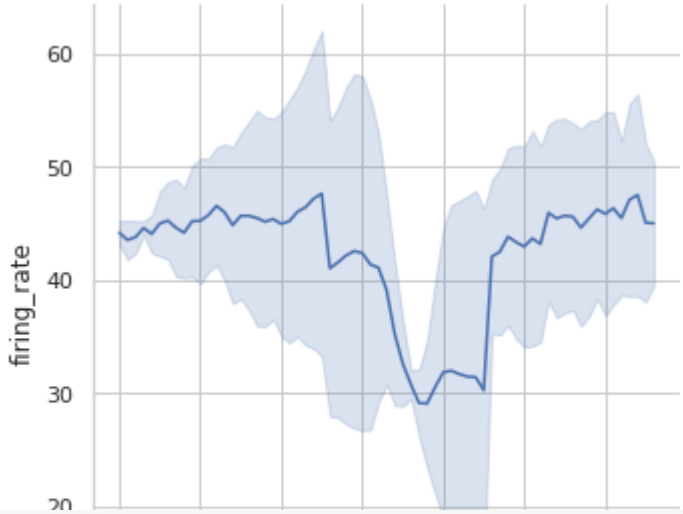
```
sns.relplot(x="time", y="firing_rate",
            kind="line", data=dots);
```



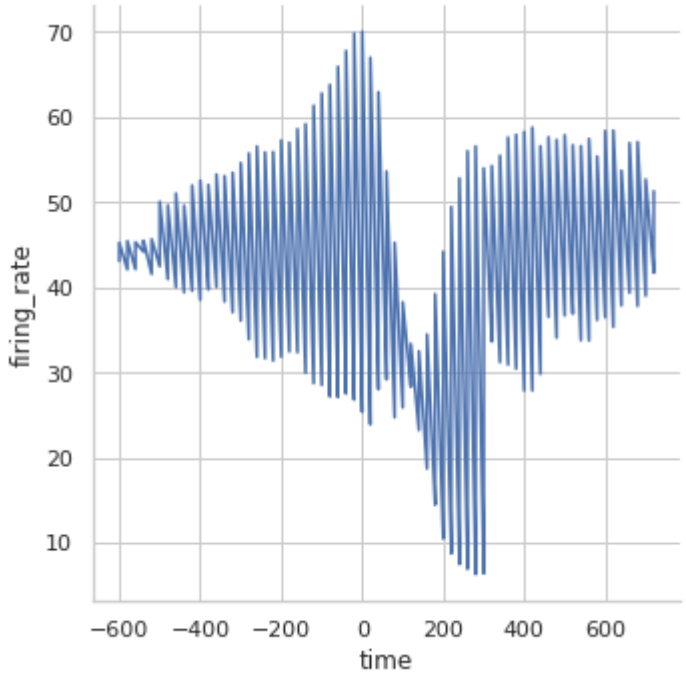

```
sns.relplot(x="time", y="firing_rate",
            ci=None, kind="line", data=dots);
```




```
sns.relplot(x="time", y="firing_rate",
            ci="sd", kind="line", data=dots);
```

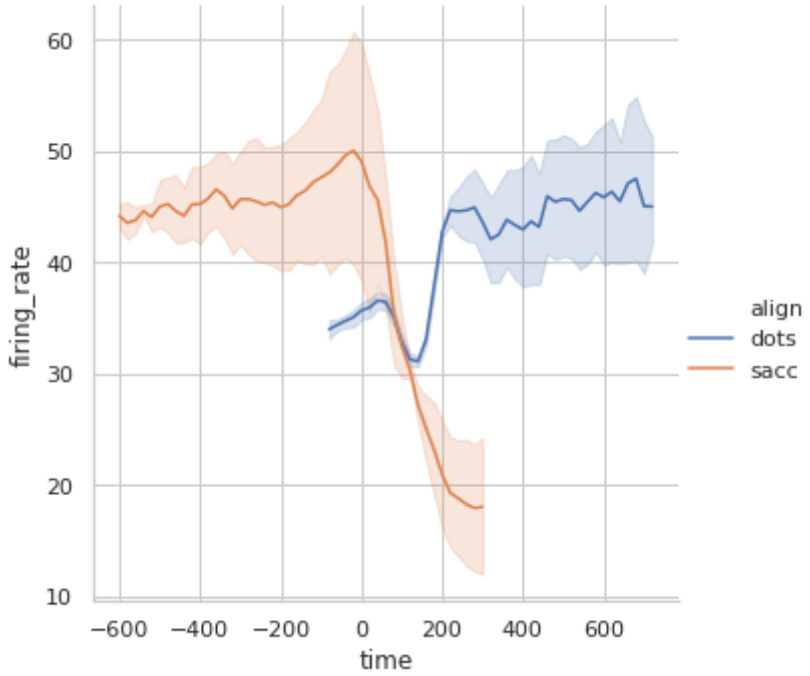



```
sns.relplot(x="time", y="firing_rate",
            estimator=None, kind="line", data=dots);
```

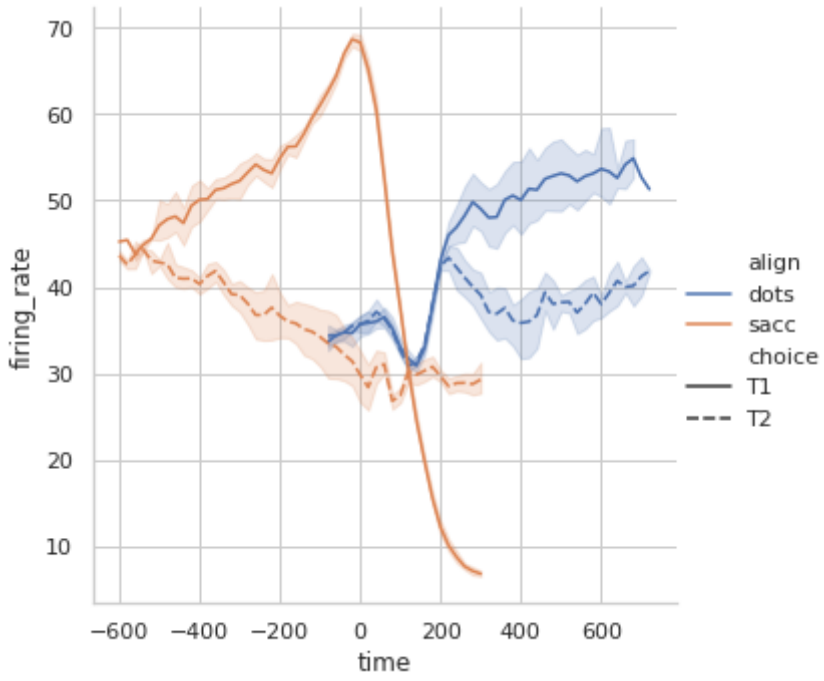



```
sns.relplot(x="time", y="firing_rate",
            hue="choice", kind="line", data=dots);
```


```
sns.relplot(x="time", y="firing_rate",
            hue="align", kind="line", data=dots);
```

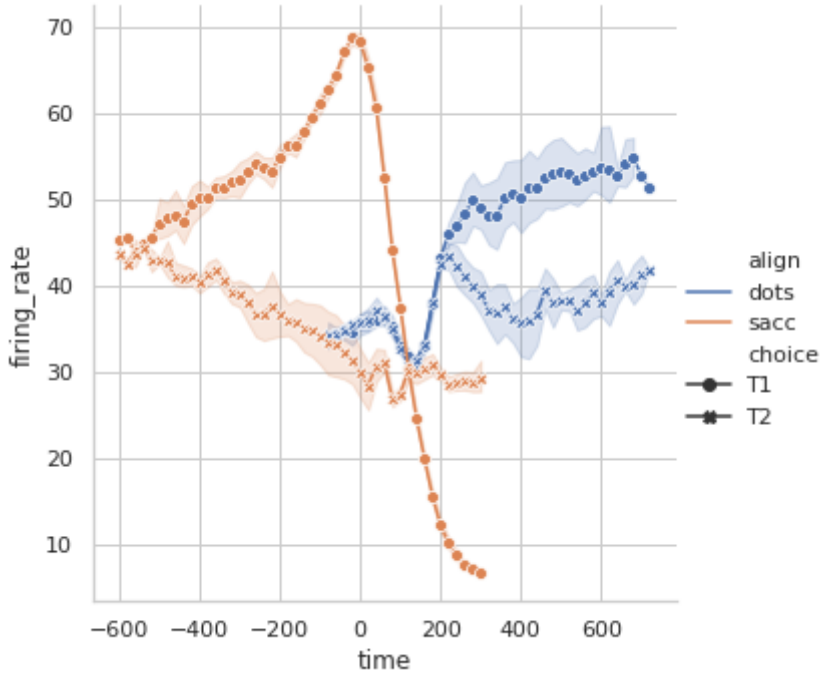



```
sns.relplot(x="time", y="firing_rate",
            hue="align", style="choice",
            kind="line", data=dots);
```

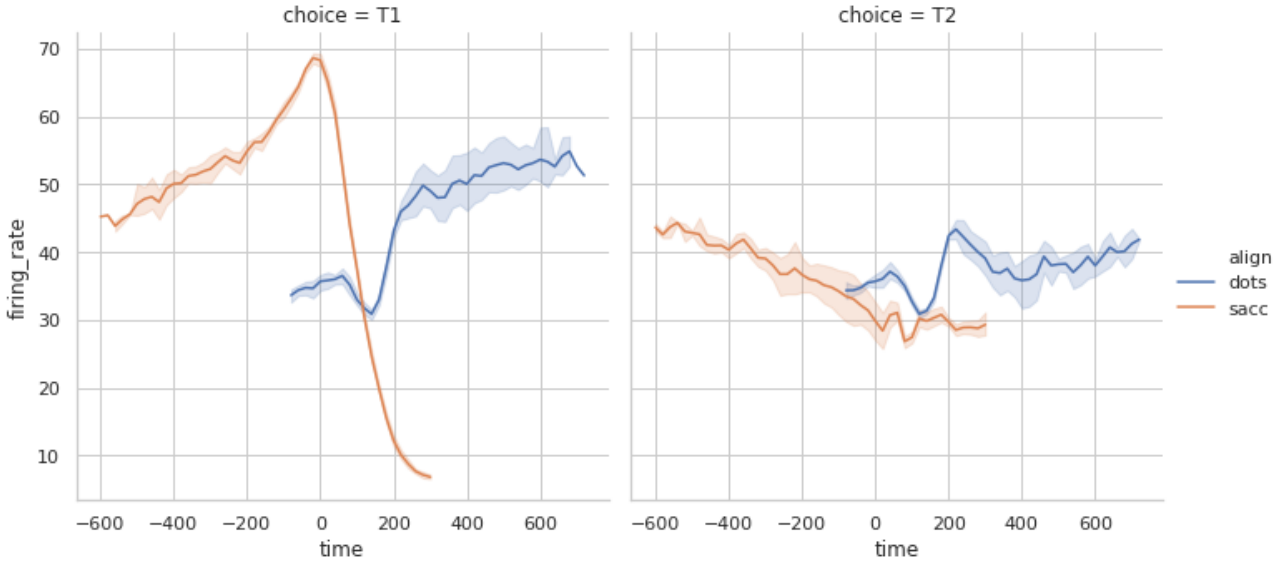



```
sns.relplot(x="time", y="firing_rate",
            hue="align", style="choice",
            dashes=False, markers=True,
```

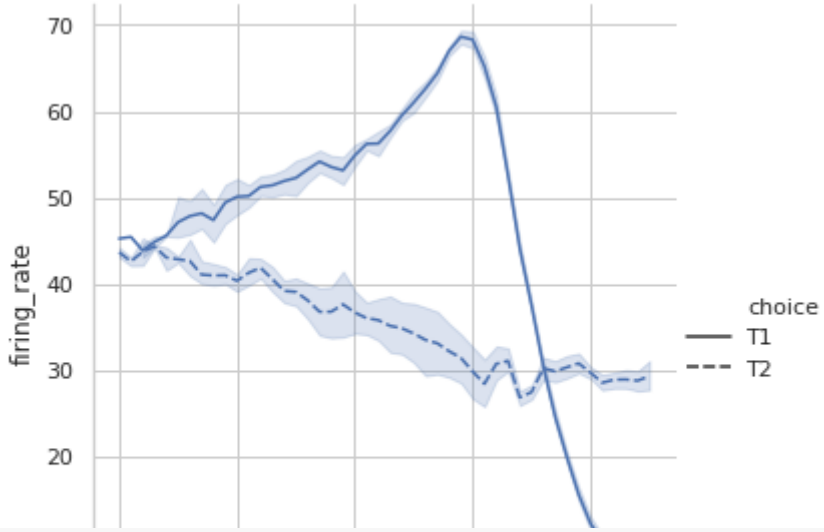
```
            kind="line", data=dots);
```



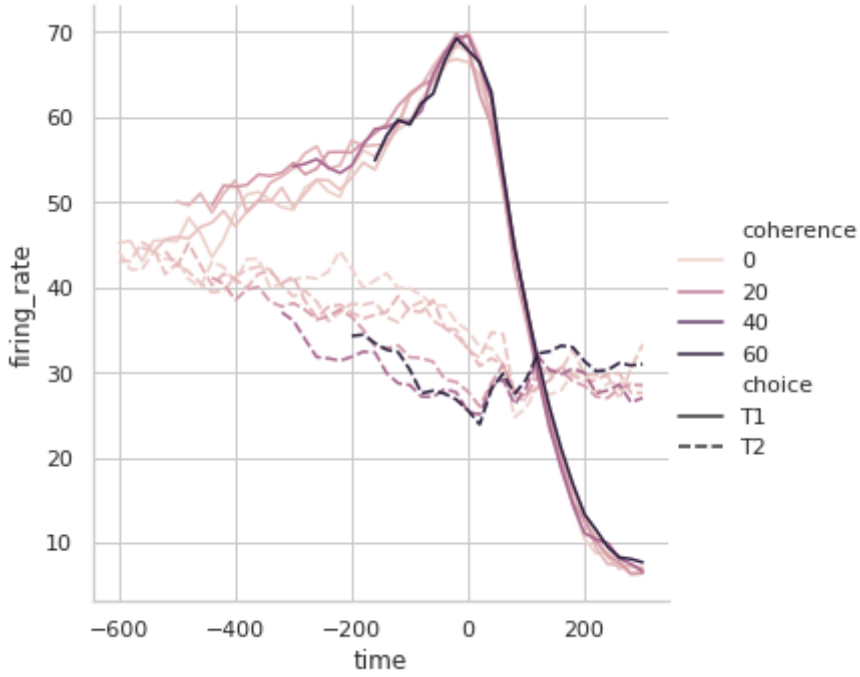

```
sns.relplot(x="time", y="firing_rate",
            hue="align", col="choice",
            kind="line", data=dots);
```



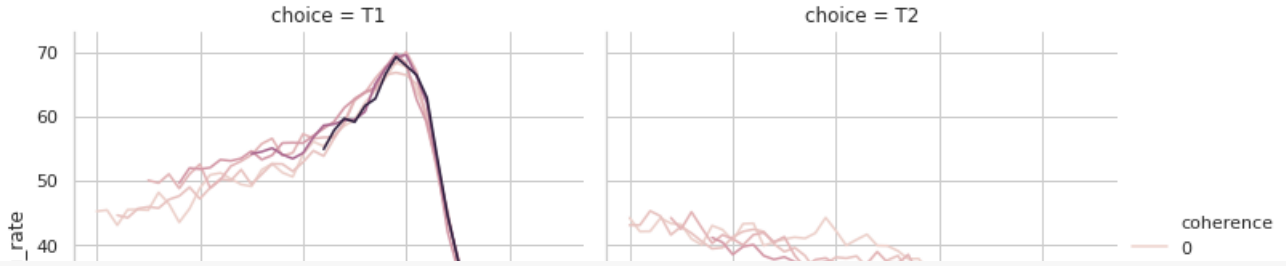

```
sns.relplot(x="time", y="firing_rate",
            style="choice",
            kind="line", data=dots.query("align == 'sacc'"));
```




```
sns.relplot(x="time", y="firing_rate",
            hue="coherence", style="choice",
            kind="line", data=dots.query("align == 'sacc'"));
```




```
sns.relplot(x="time", y="firing_rate",
            hue="coherence", col="choice",
            kind="line", data=dots.query("align == 'sacc'"));
```



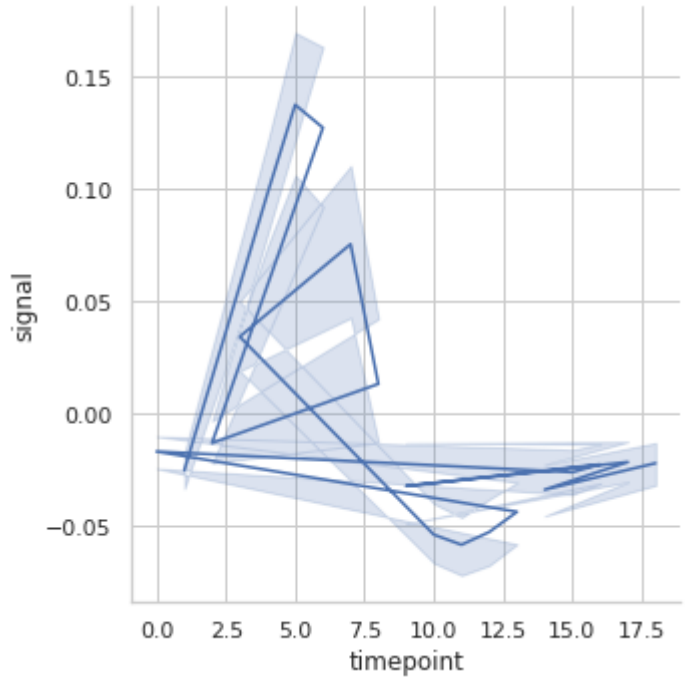

```
fmri = sns.load_dataset("fmri")
fmri
```

	subject	timepoint	event	region	signal
0	s13	18	stim	parietal	-0.017552
1	s5	14	stim	parietal	-0.080883
2	s12	18	stim	parietal	-0.081033
3	s11	18	stim	parietal	-0.046134
4	s10	18	stim	parietal	-0.037970
...	...	...	...	...	...
1059	s0	8	cue	frontal	0.018165
1060	s13	7	cue	frontal	-0.029130
1061	s12	7	cue	frontal	-0.004939
1062	s11	7	cue	frontal	-0.025367
1063	s0	0	cue	parietal	-0.006899

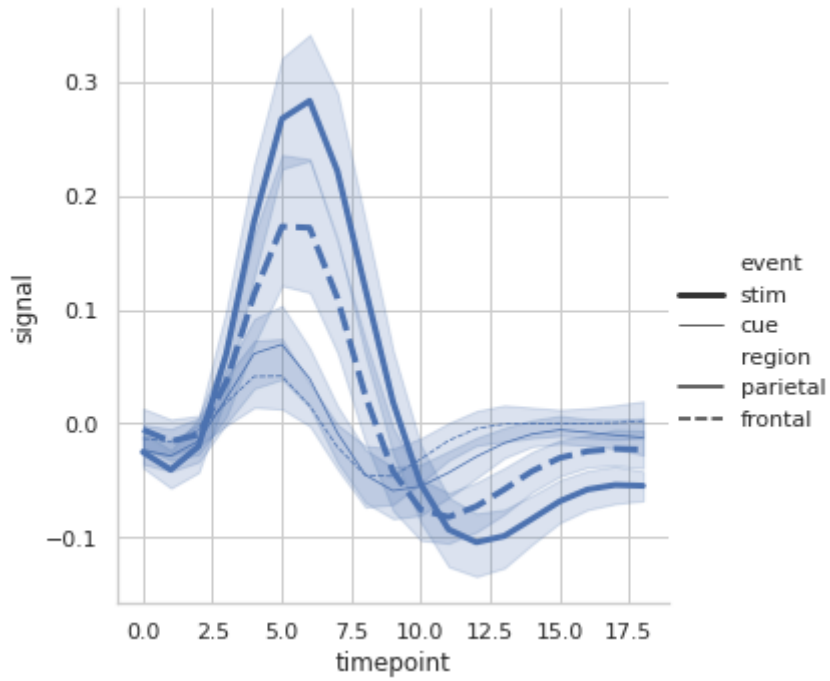
1064 rows × 5 columns

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

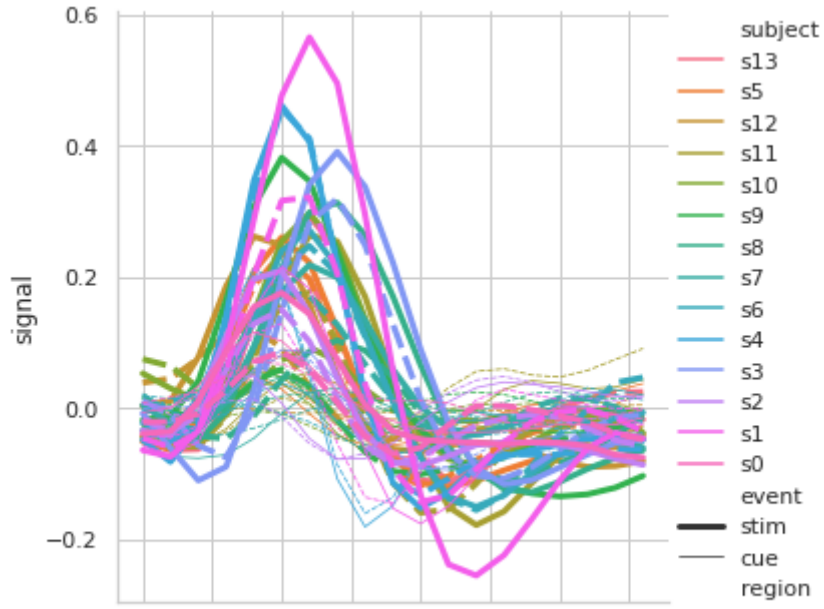
```
sns.relplot(x="timepoint", y="signal",
            sort=False, kind="line", data=fmri);
```



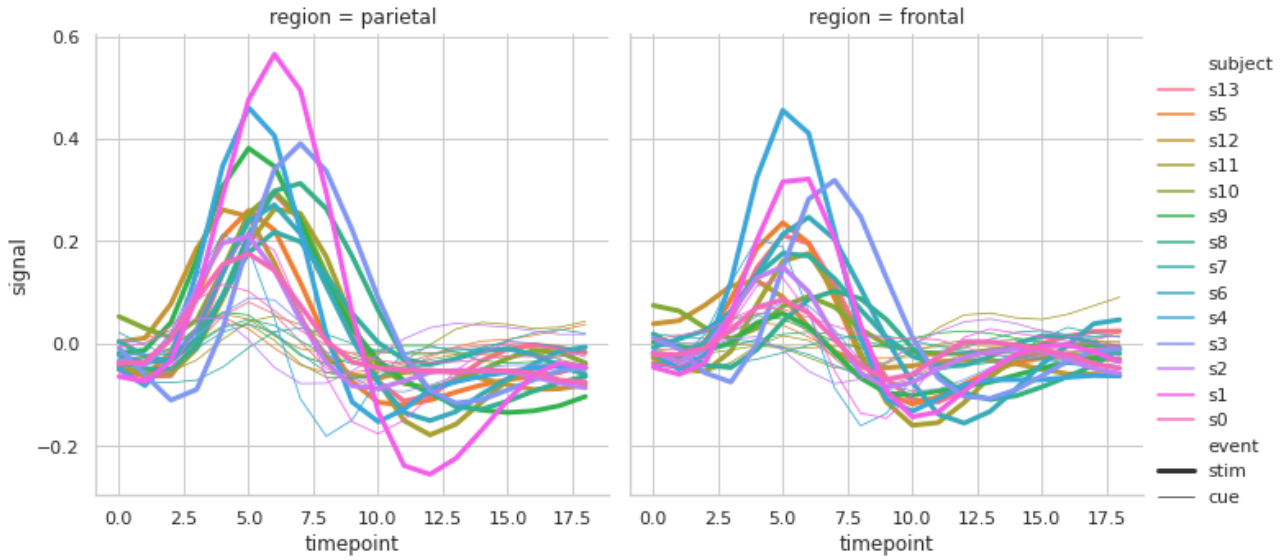

```
sns.relplot(x="timepoint", y="signal",
            style="region", size="event",
            kind="line", data=fmri);
```



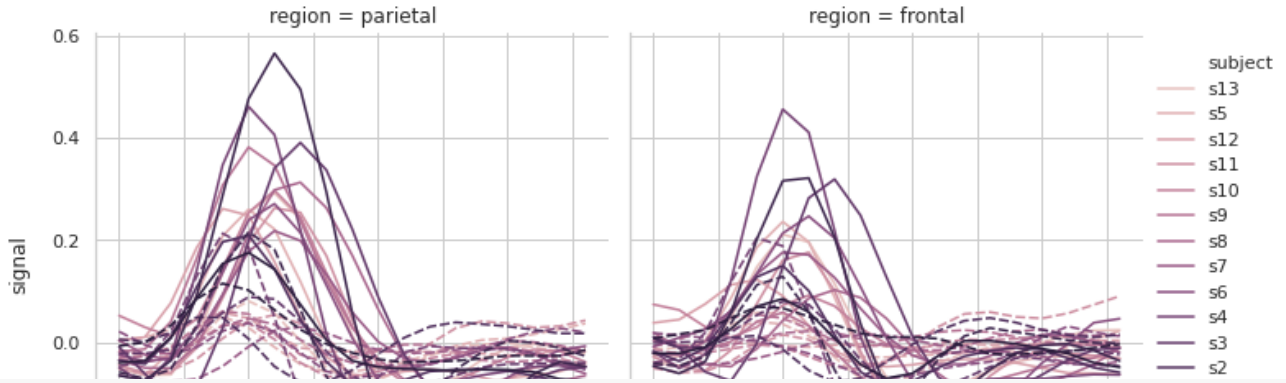

```
sns.relplot(x="timepoint", y="signal",
            hue="subject", style="region", size="event",
            kind="line", data=fmri);
```



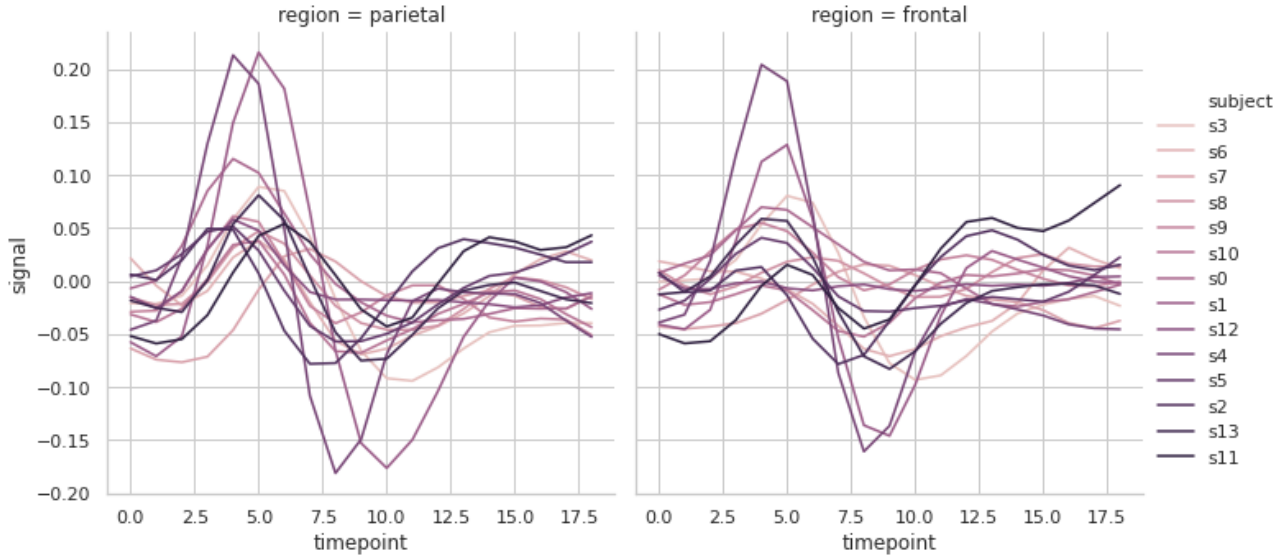

```
sns.relplot(x="timepoint", y="signal",
            hue="subject", col="region", size="event",
            kind="line", data=fmri);
```



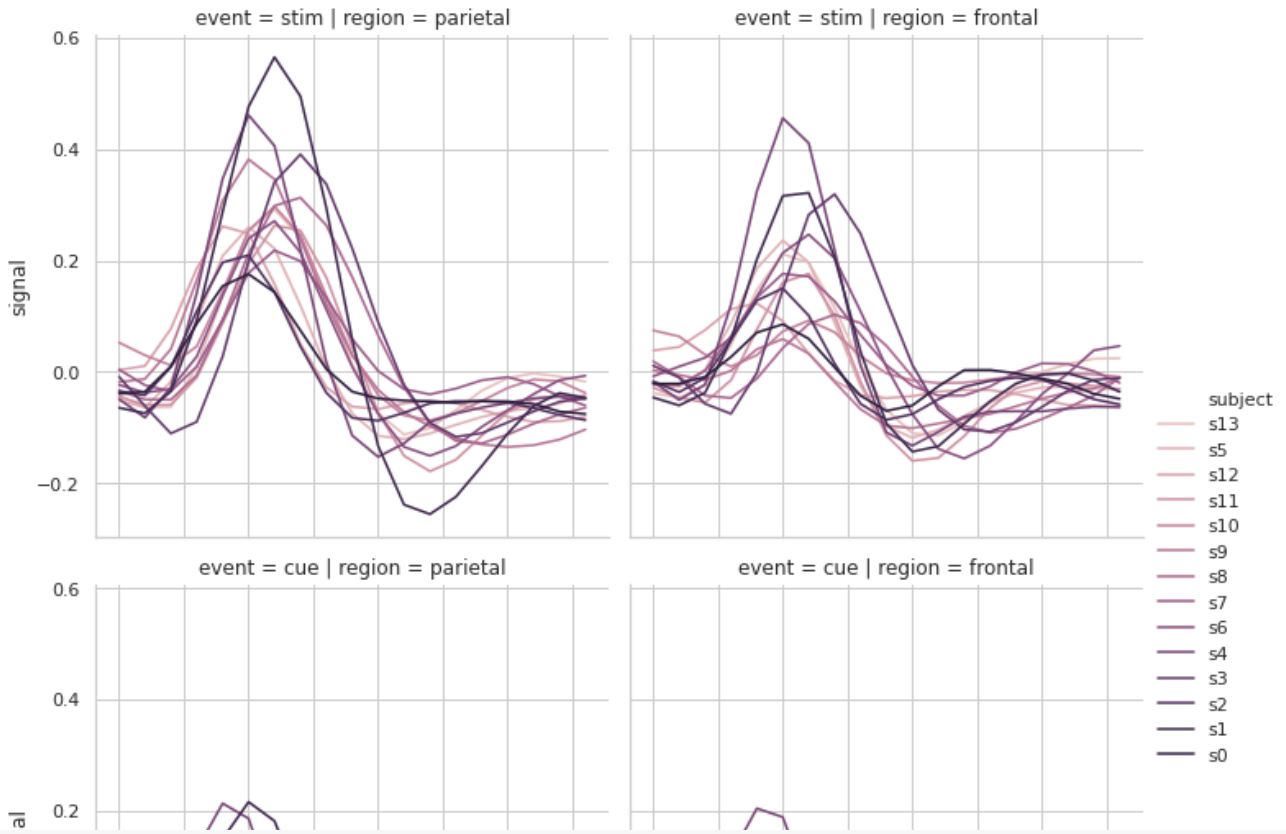

```
palette = sns.cubehelix_palette(n_colors=14, light=0.8)
sns.relplot(x="timepoint", y="signal",
            hue="subject", col="region", style="event",
            palette=palette, kind="line", data=fmri);
```



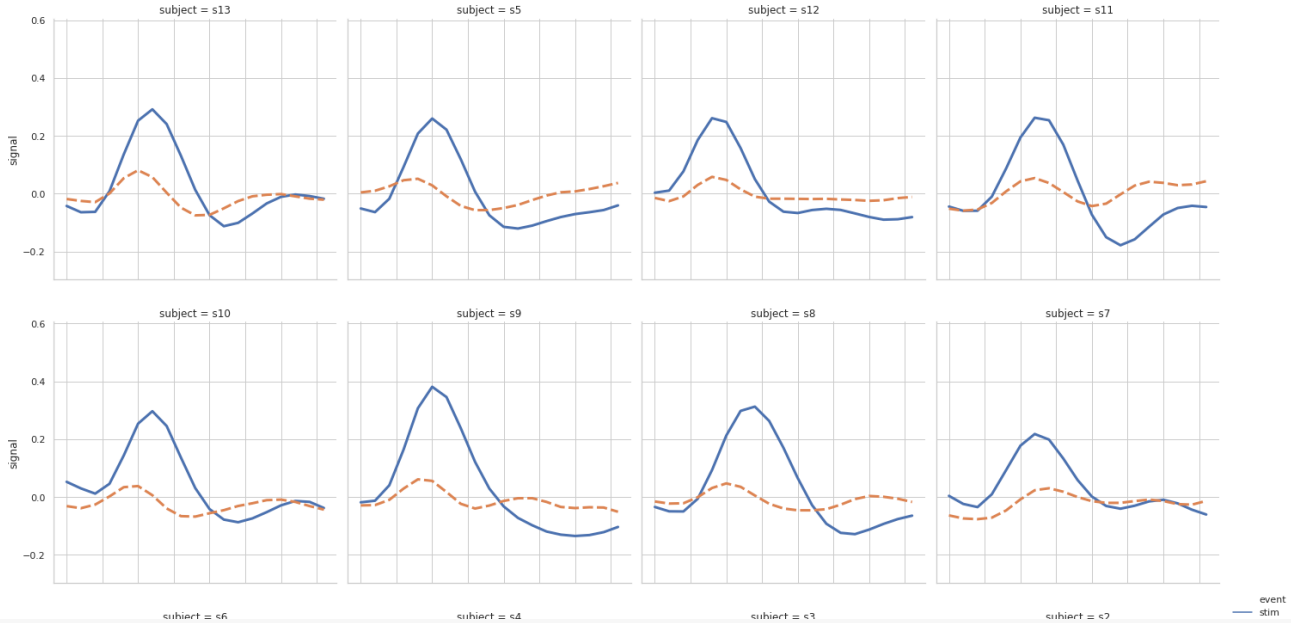

```
sns.relplot(x="timepoint", y="signal",
            hue="subject", col="region",
            palette=palette, kind="line", data=fmri.query("event == 'cue'"));
```




```
sns.relplot(x="timepoint", y="signal",
            hue="subject", col="region", row="event",
            palette=palette, kind="line", data=fmri);
```



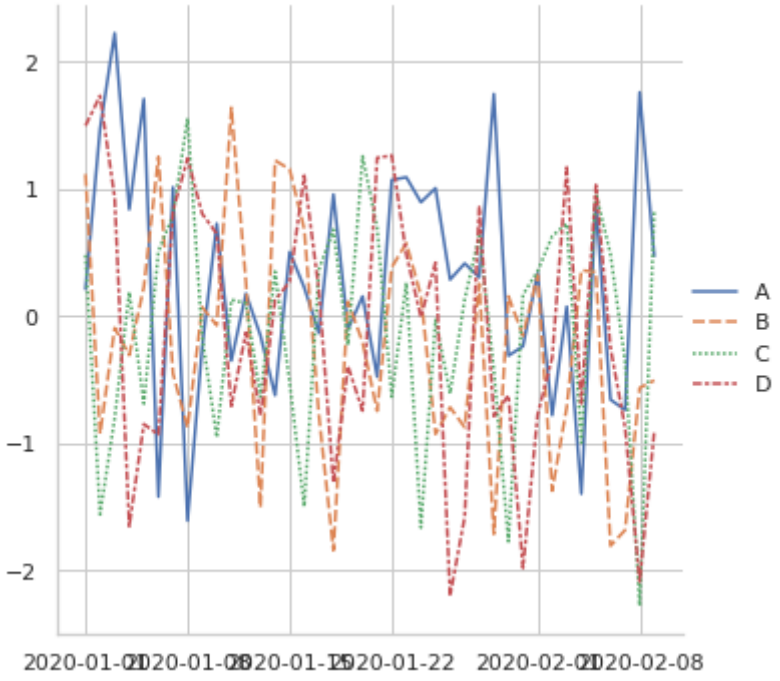

```
sns.relplot(x="timepoint", y="signal",
            hue="event", style="event",
            col="subject", col_wrap=4, linewidth=3,
            kind="line", data=fmri.query("region == 'parietal'"));
```



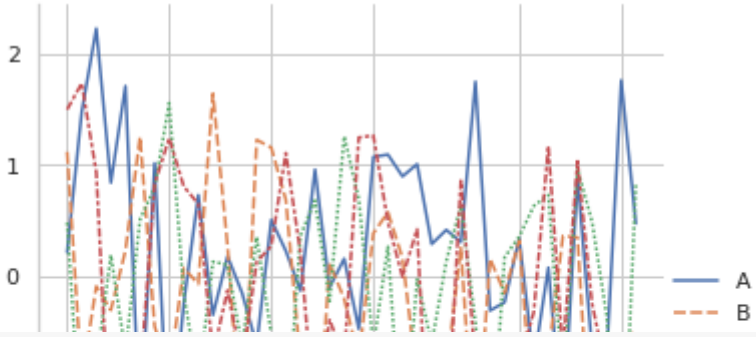

```
tdf = pd.DataFrame(np.random.randn(40, 4),
                   index=pd.date_range('2020-01-01', periods=40),
                   columns=['A', 'B', 'C', 'D'])
tdf
```

	A	B	C	D
2020-01-01	0.212307	1.116571	0.479458	1.497874
2020-01-02	1.475882	-0.928822	-1.563284	1.733448
2020-01-03	2.227370	-0.094513	-0.805349	0.932671
2020-01-04	0.838842	-0.308714	0.187207	-1.655722
2020-01-05	1.708836	0.228033	-0.690887	-0.851920
2020-01-06	-1.423851	1.252746	0.508675	-0.934869
2020-01-07	1.014049	-0.454399	0.794062	0.827190
2020-01-08	-1.612318	-0.878517	1.554968	1.240540
2020-01-09	-0.245562	0.069111	-0.164191	0.804465
2020-01-10	0.727454	-0.074143	-0.957051	0.647150
2020-01-11	-0.352806	1.640270	0.127467	-0.709762

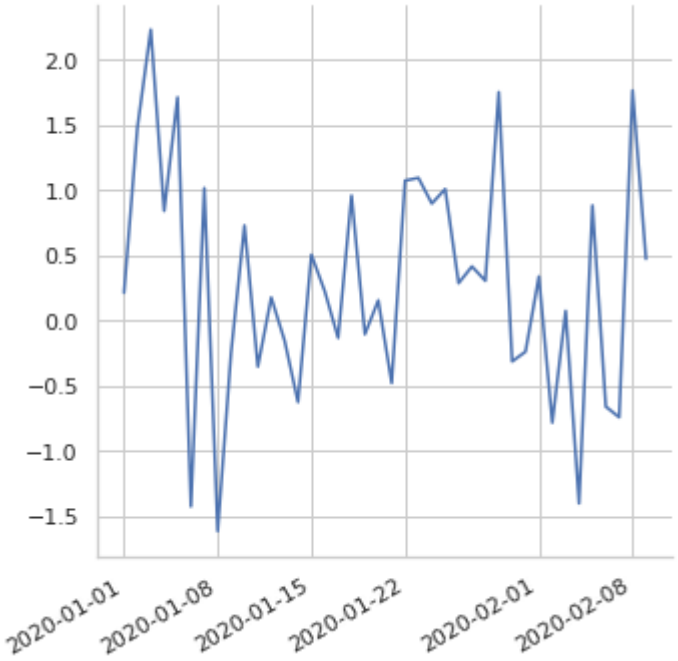
```
sns.relplot(kind="line", data=tdf);
```




```
g = sns.relplot(kind="line", data=tdf)
g.fig.autofmt_xdate()
```




```
g = sns.relplot(kind="line", data=tdf['A'])
g.fig.autofmt_xdate()
```



범주형 데이터(Categorical Data)

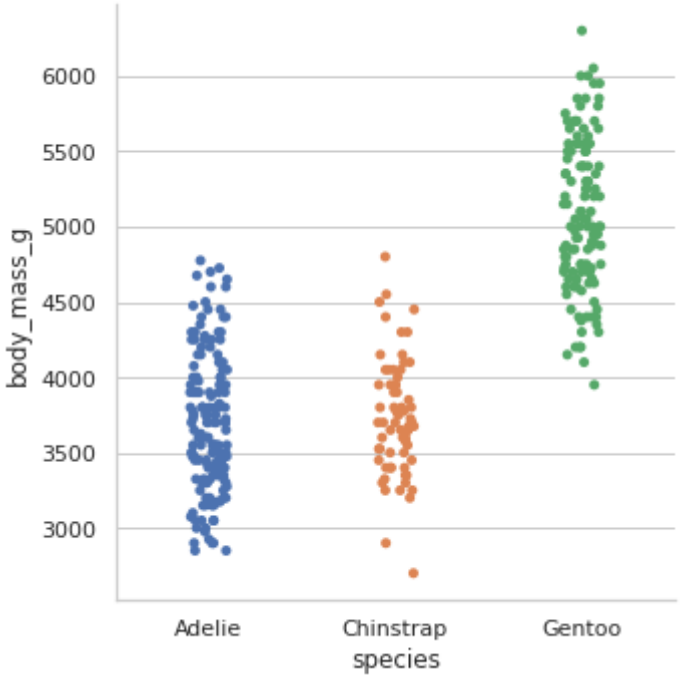
범주형 산점도(Categorical scatterplots)

- `stripplot()` (with `kind="strip"`; the default)
- `swarmplot()` (with `kind="swarm"`)

```
penguins
```

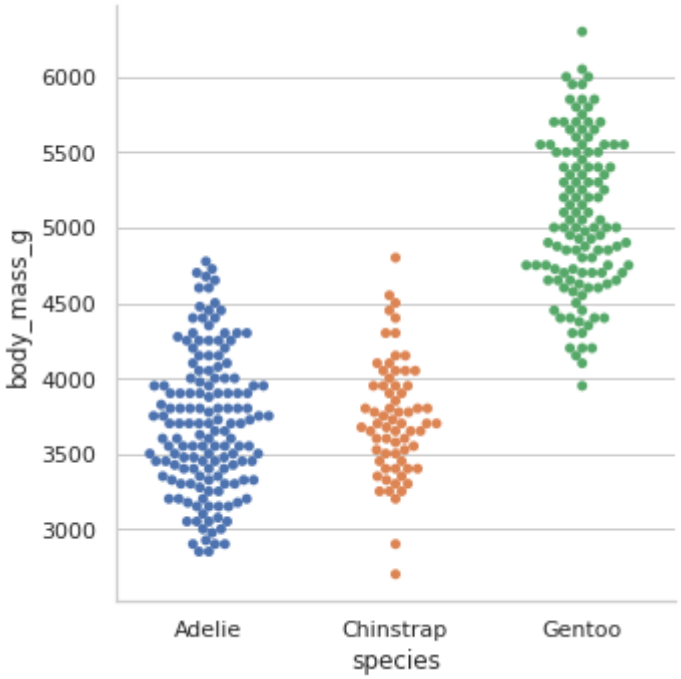
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	

```
sns.catplot(x="species", y="body_mass_g", data=penguins);
```

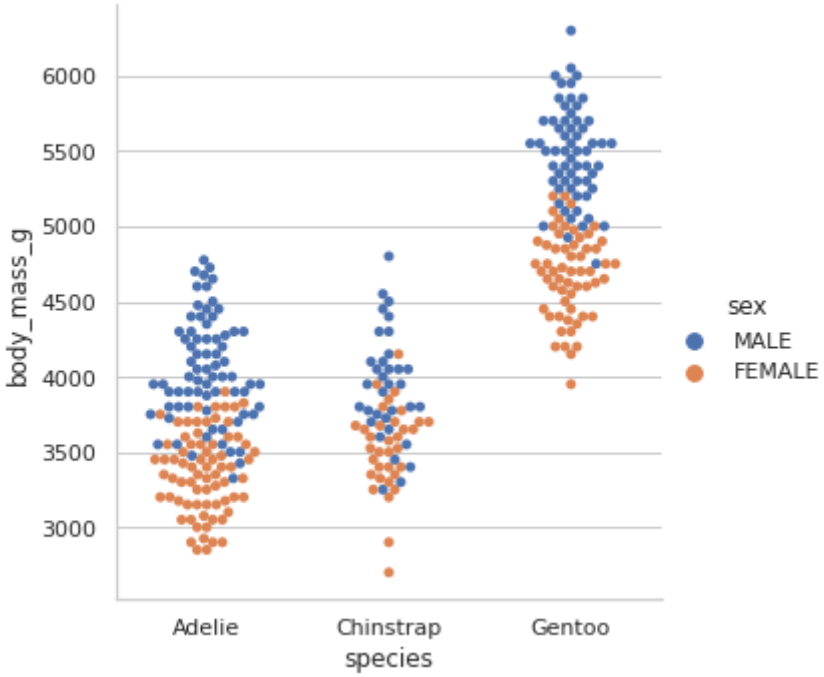



```
sns.catplot(x="species", y="body_mass_g",
            jitter=False, data=penguins);
```

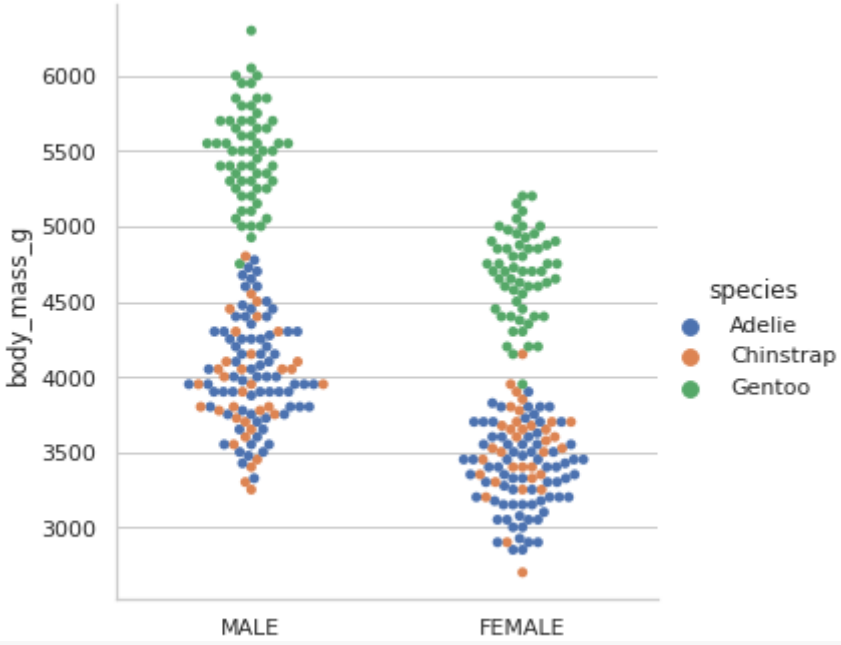
```
sns.catplot(x="species", y="body_mass_g",
            kind="swarm", data=penguins);
```



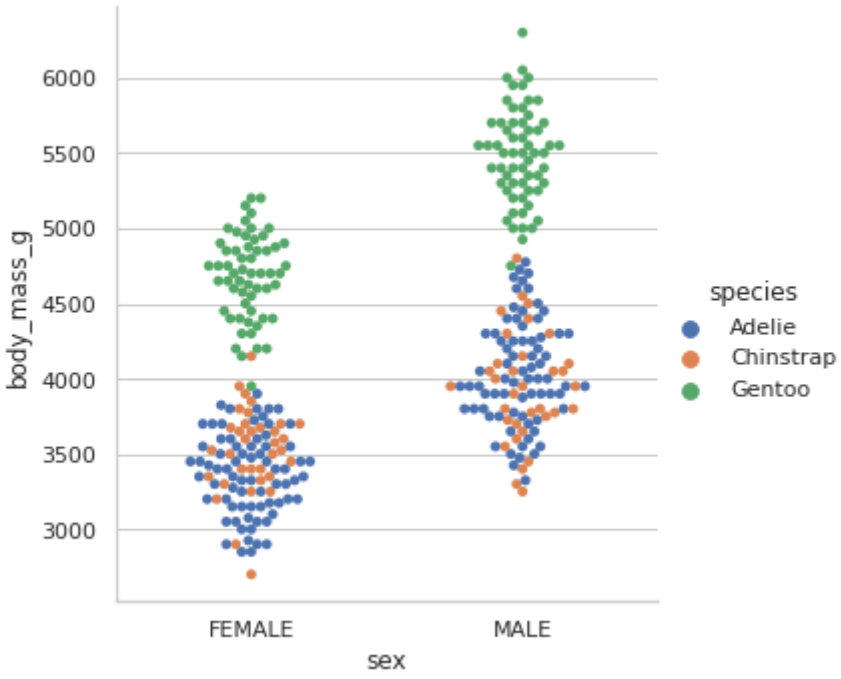

```
sns.catplot(x="species", y="body_mass_g",
            hue="sex",
            kind="swarm", data=penguins);
```



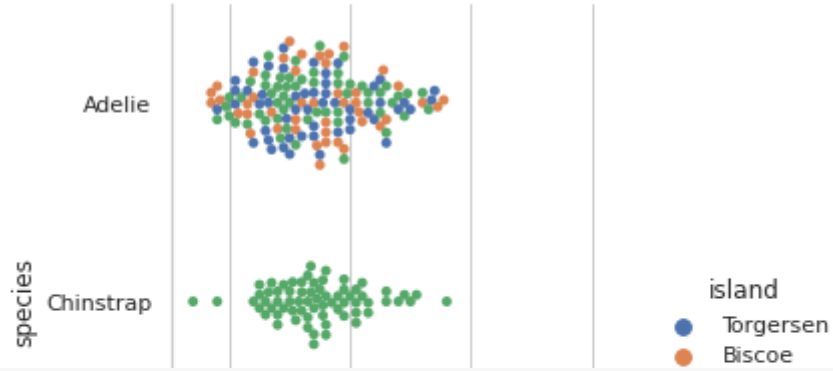

```
sns.catplot(x="sex", y="body_mass_g",
            hue="species",
            kind="swarm", data=penguins);
```



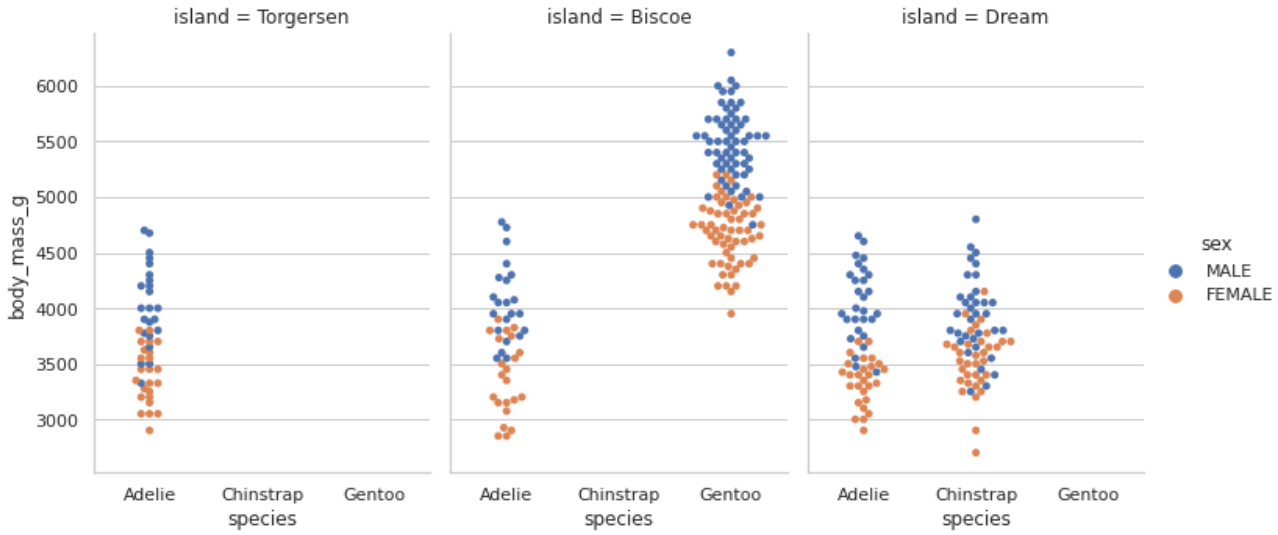

```
sns.catplot(x="sex", y="body_mass_g",
            hue="species", kind="swarm",
            order=["FEMALE", "MALE"], data=penguins);
```




```
sns.catplot(x="body_mass_g", y="species",
            hue="island", kind="swarm",
            data=penguins);
```




```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", col="island", aspect=.7,
            kind="swarm", data=penguins);
```

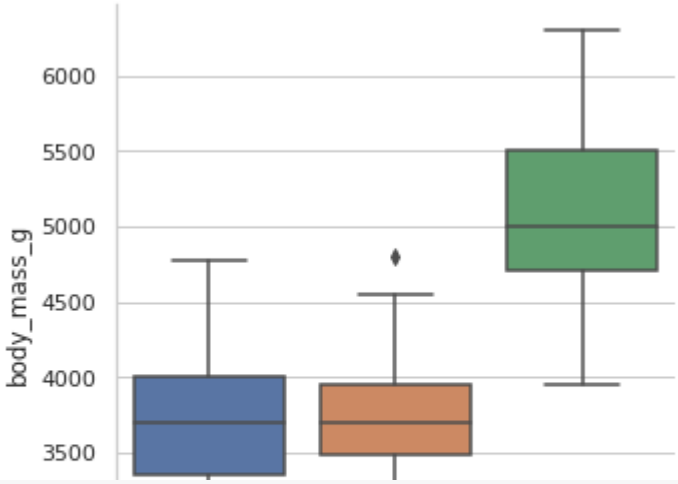


범주형 분포도(Categorical distribution plots):

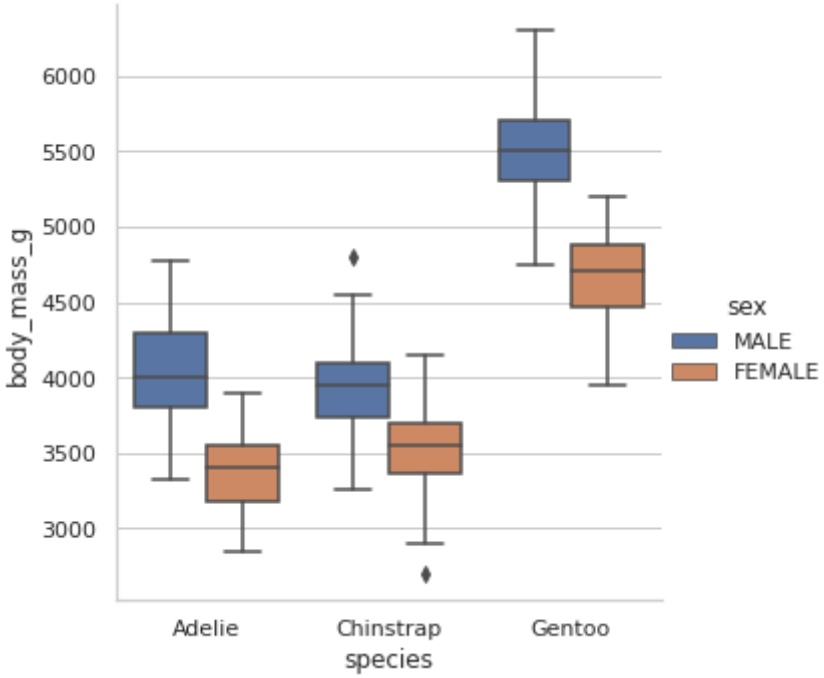
- `boxplot()` (with `kind="box"`)
- `boxenplot()` (with `kind="boxen"`)
- `violinplot()` (with `kind="violin"`)

박스 플롯(Box plots)

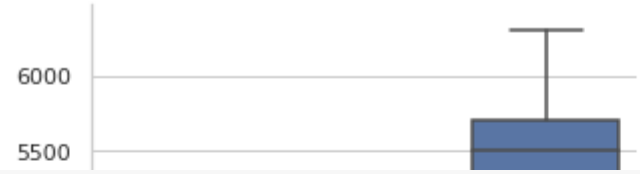
```
sns.catplot(x="species", y="body_mass_g",
            kind="box", data=penguins);
```



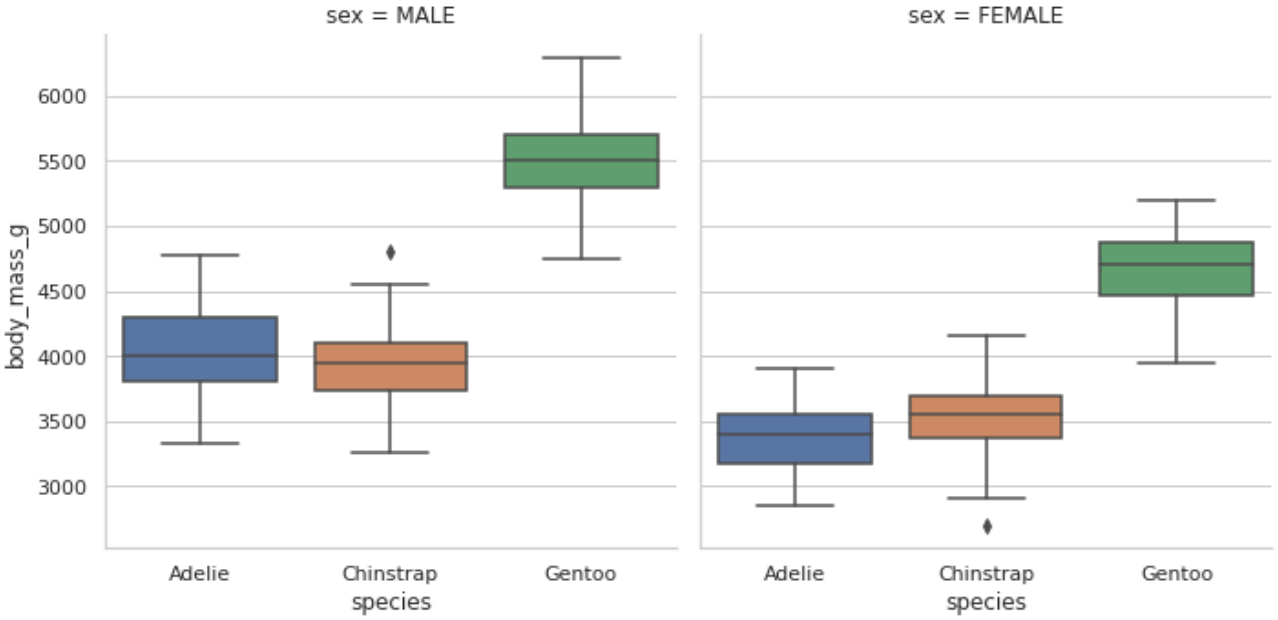

```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="box", data=penguins);
```



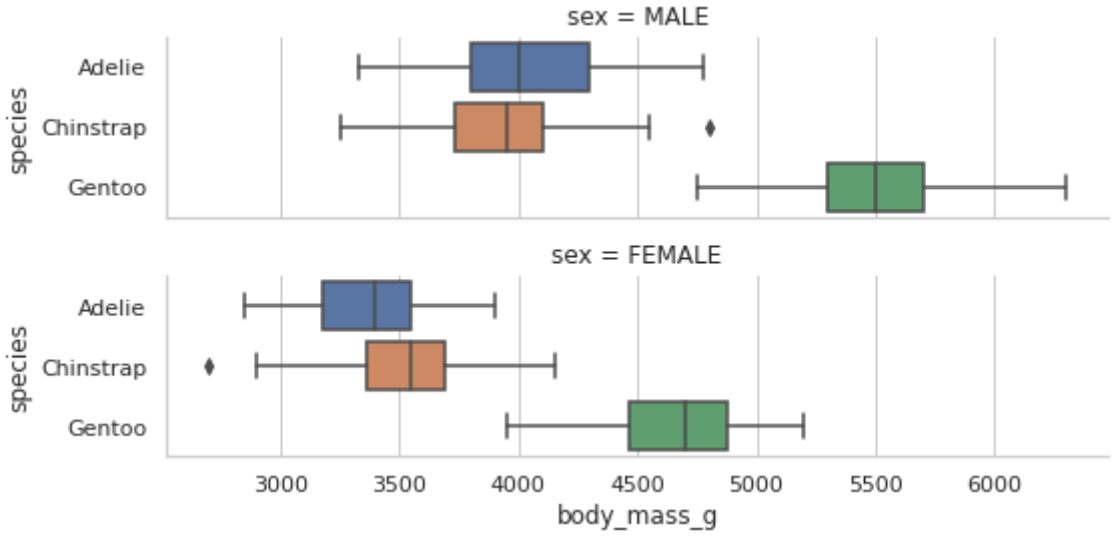

```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="box",
            dodge=False, data=penguins);
```




```
sns.catplot(x="species", y="body_mass_g",
            col="sex", kind="box",
            data=penguins);
```




```
sns.catplot(x="body_mass_g", y="species",
            row="sex", kind="box",
            height=2, aspect=4,
            data=penguins);
```

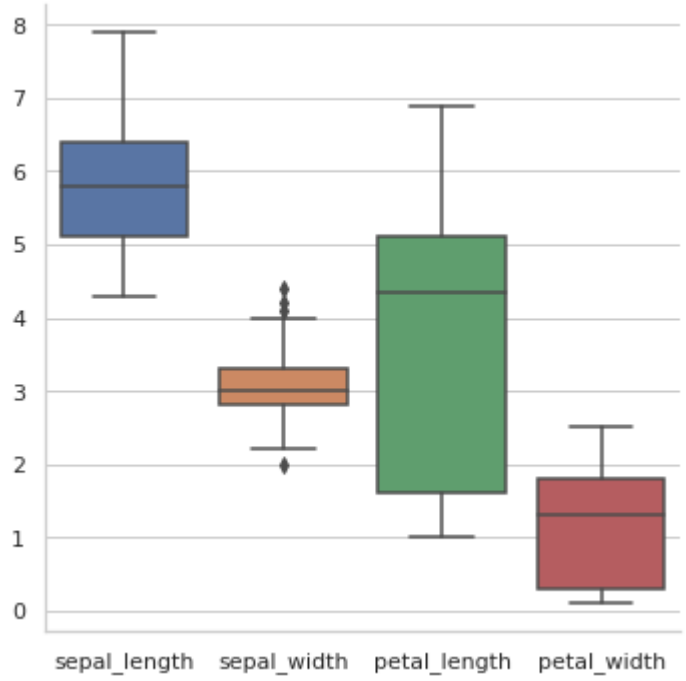



```
iris = sns.load_dataset("iris")
iris
```

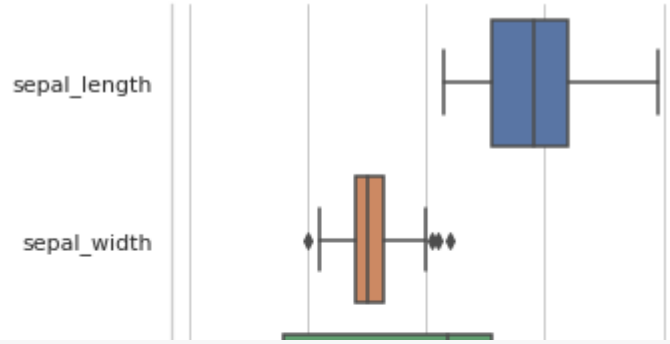
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

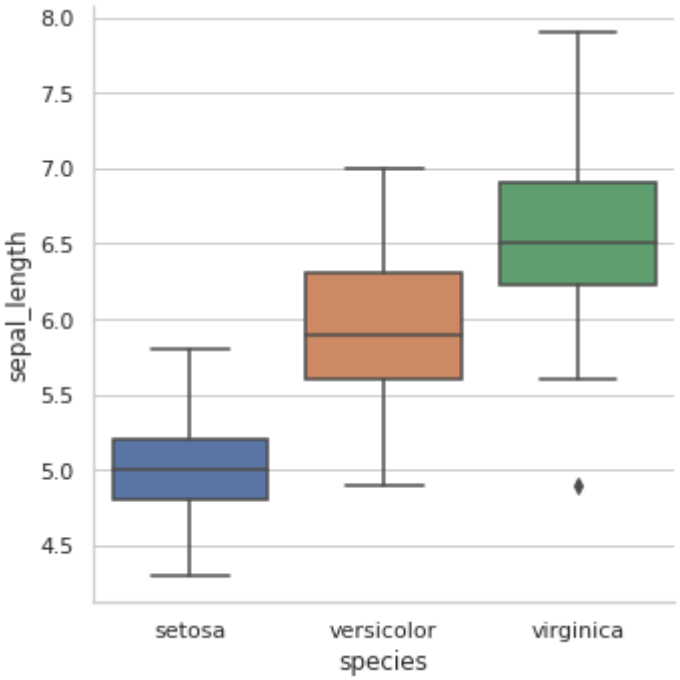
```
sns.catplot(kind="box", data=iris);
```



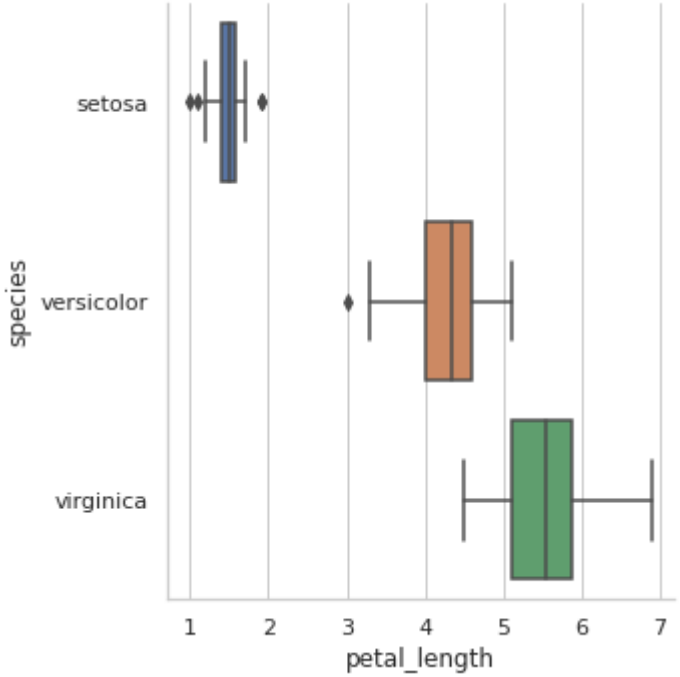

```
sns.catplot(kind="box", orient='h', data=iris);
```




```
sns.catplot(x="species", y="sepal_length", kind="box", data=iris);
```




```
sns.catplot(x="petal_length", y="species", kind="box", data=iris);
```



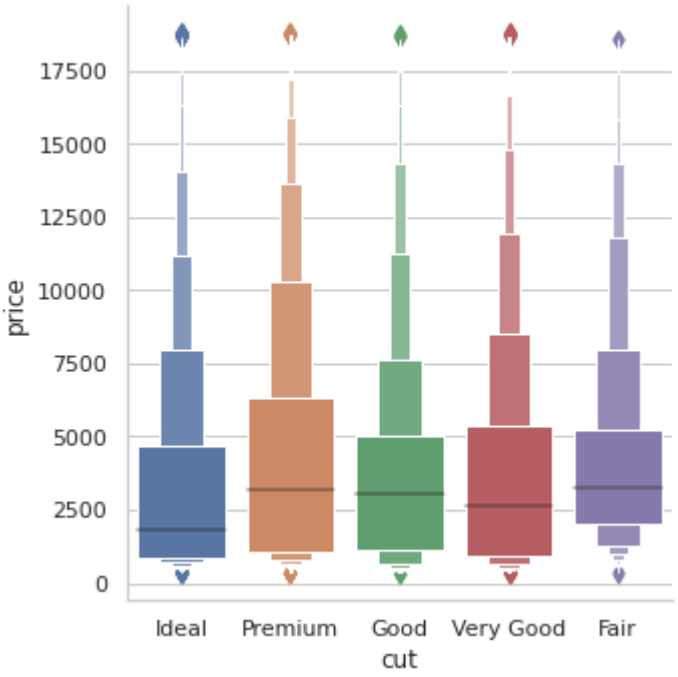
박슨 플롯(Boxen plots)

```
diamonds = sns.load_dataset("diamonds")
diamonds
```

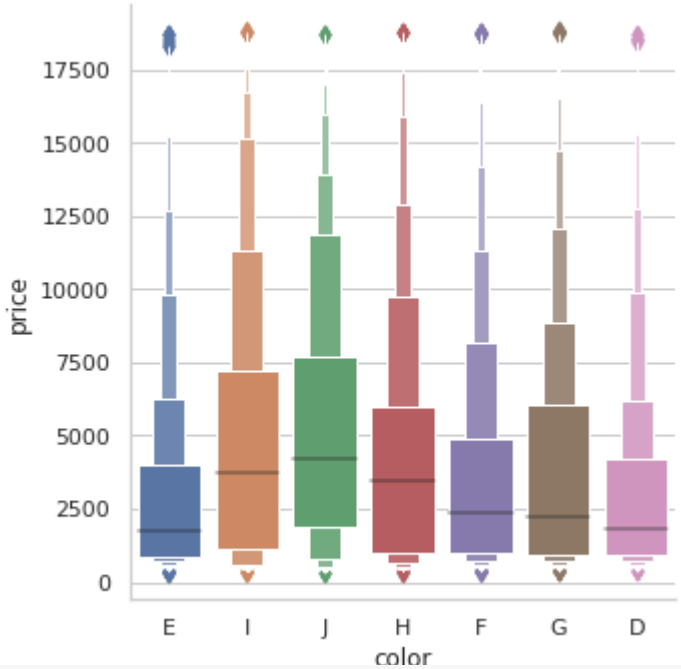
	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...	...	...
53935	0.72	Ideal	D	SI1	60.8	57.0	2757	5.75	5.76	3.50
53936	0.72	Good	D	SI1	63.1	55.0	2757	5.69	5.75	3.61
53937	0.70	Very Good	D	SI1	62.8	60.0	2757	5.66	5.68	3.56
53938	0.86	Premium	H	SI2	61.0	58.0	2757	6.15	6.12	3.74
53939	0.75	Ideal	D	SI2	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 10 columns

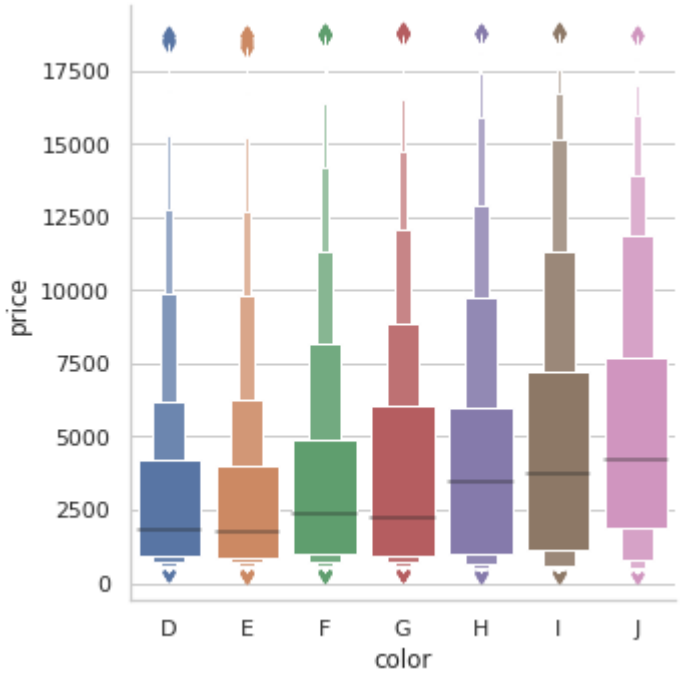
```
sns.catplot(x="cut", y="price",
            kind="boxen", data=diamonds);
```



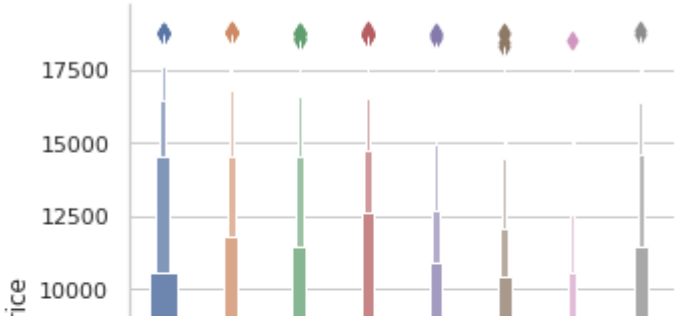

```
sns.catplot(x="color", y="price",
            kind="boxen", data=diamonds);
```




```
sns.catplot(x="color", y="price",
            kind="boxen", data=diamonds.sort_values("color"));
```




```
sns.catplot(x="clarity", y="price",
            kind="boxen", data=diamonds);
```



바이올린 플롯(Violin plots)

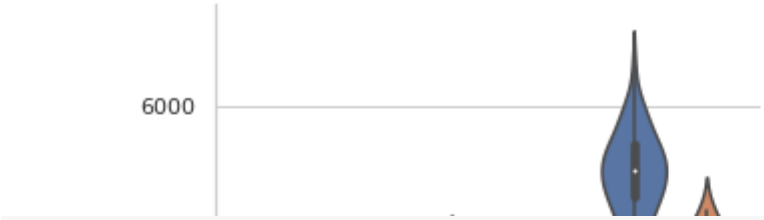
- `violinplot` : 커널 밀도 추정과 상자 도표 결합

```
penguins
```

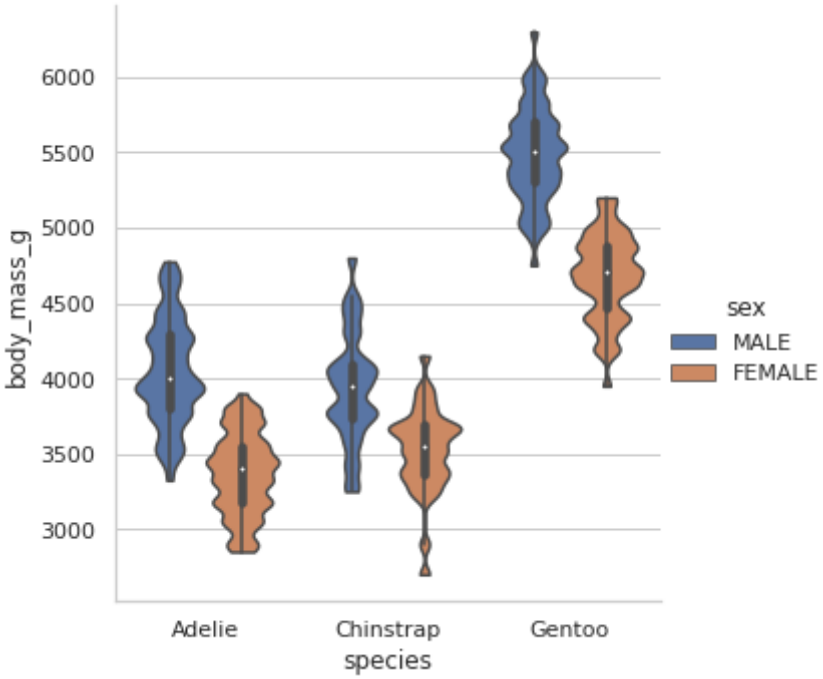
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	
340	Gentoo	Biscoe	46.8	14.3	215.0	
341	Gentoo	Biscoe	50.4	15.7	222.0	
342	Gentoo	Biscoe	45.2	14.8	212.0	
343	Gentoo	Biscoe	49.9	16.1	213.0	

344 rows × 7 columns

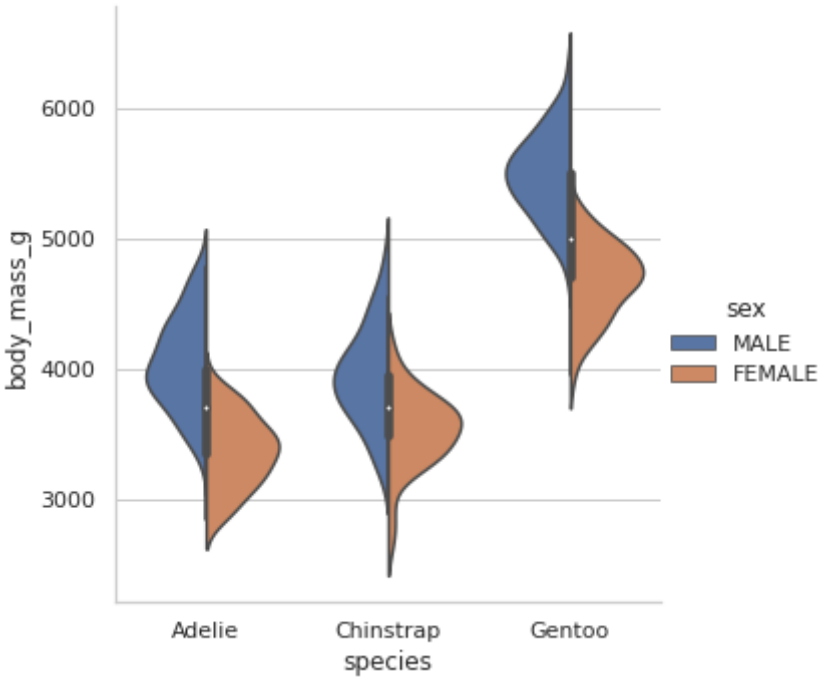
```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="violin", data=penguins);
```



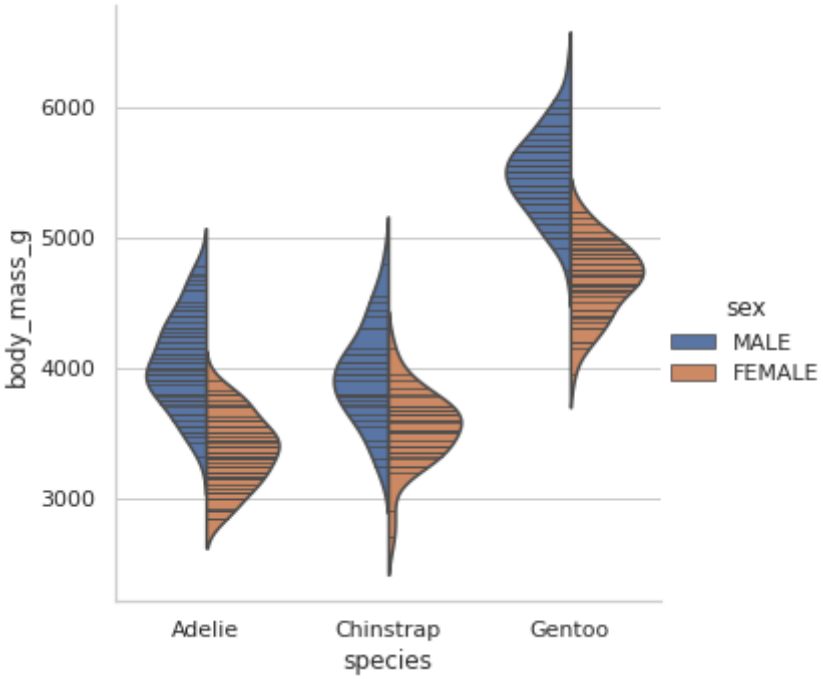

```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="violin",
            bw=.15, cut=0,
            data=penguins);
```



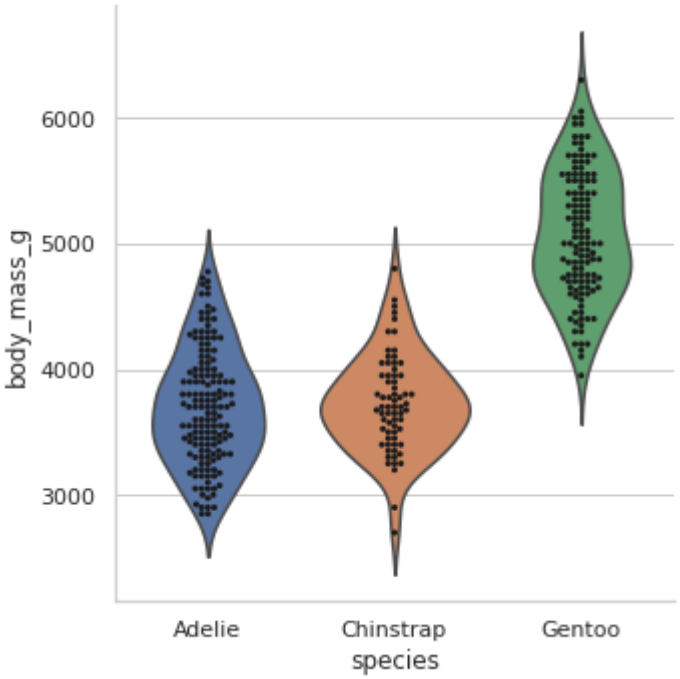

```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="violin",
            split=True, data=penguins);
```



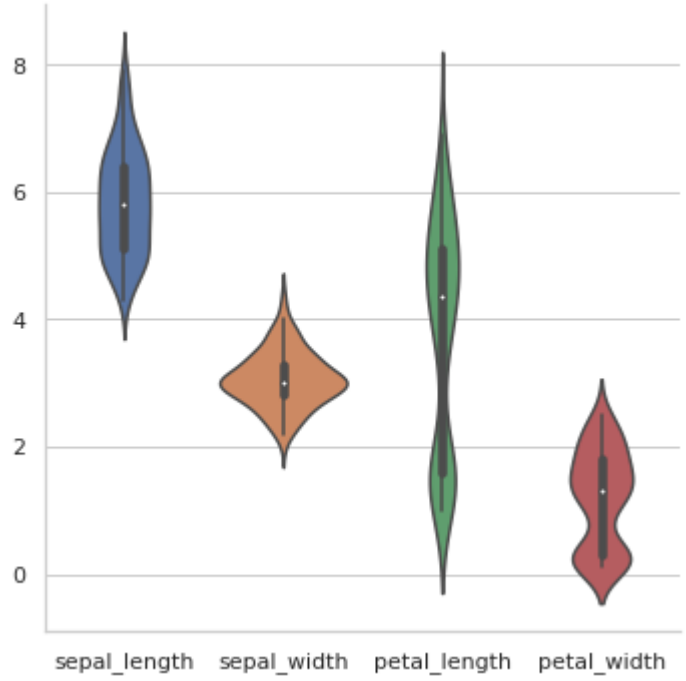
```
sns.catplot(x="species", y="body_mass_g",
            hue="sex", kind="violin",
            inner="stick", split=True,
            data=penguins);
```



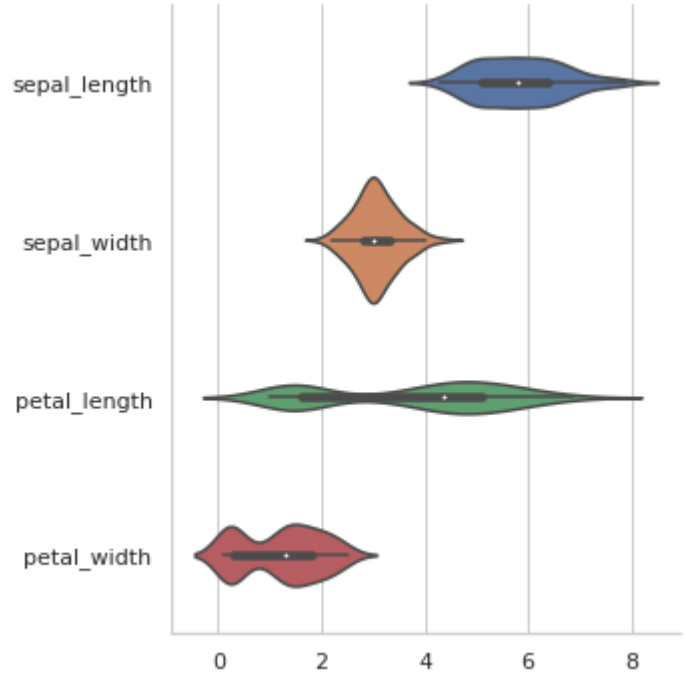

```
g = sns.catplot(x="species", y="body_mass_g",
                kind="violin",
                inner=None, data=penguins)
sns.swarmplot(x="species", y="body_mass_g",
              color="k", size=3,
              data=penguins, ax=g.ax);
```



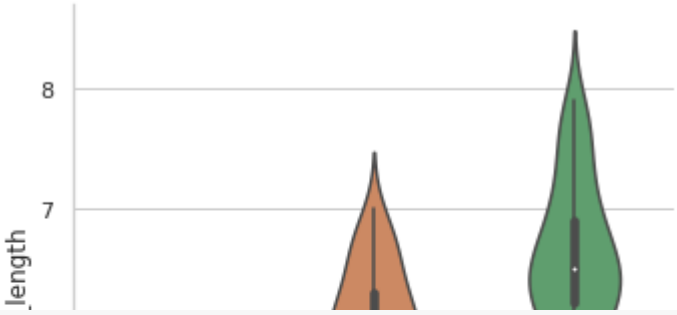

```
sns.catplot(kind="violin", data=iris);
```



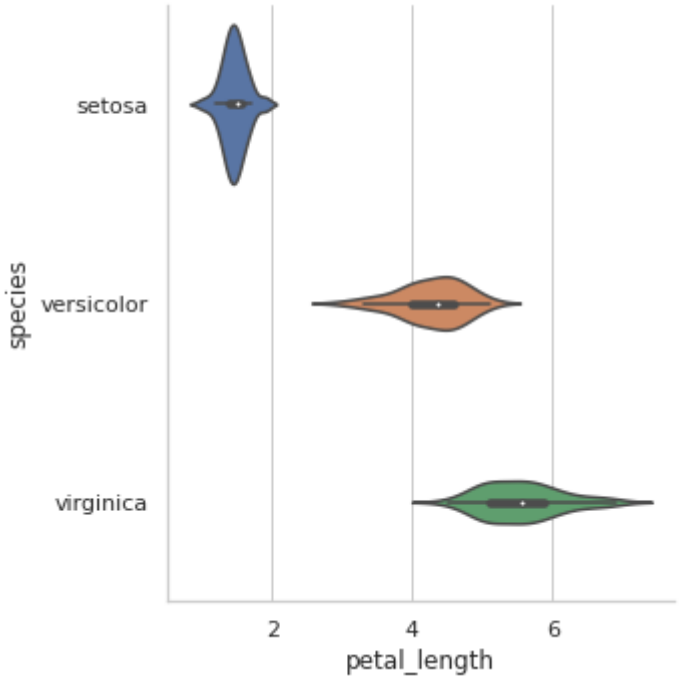

```
sns.catplot(kind="violin", orient='h', data=iris);
```




```
sns.catplot(x="species", y="sepal_length",
            kind="violin", data=iris);
```




```
sns.catplot(x="petal_length", y="species",
            kind="violin", data=iris);
```



범주형 추정치 도표(Categorical estimate plots)

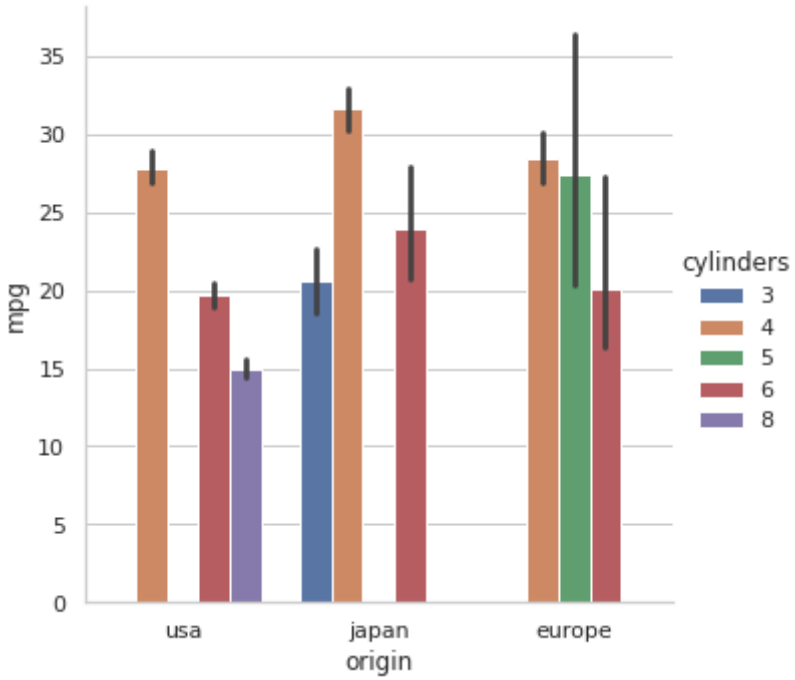
- `barplot()` (with `kind="bar"`)
- `pointplot()` (with `kind="point"`)
- `countplot()` (with `kind="count"`)

막대 플롯(Bar plots)

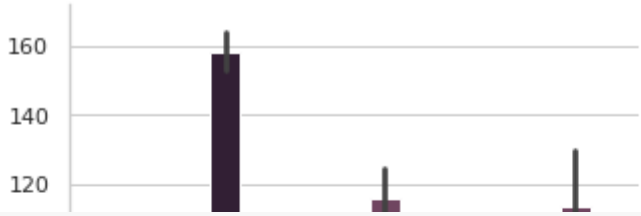
```
mpg = sns.load_dataset("mpg")
mpg
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	or
0	18.0	8	307.0	130.0	3504	12.0	70	
1	15.0	8	350.0	165.0	3693	11.5	70	
2	18.0	8	318.0	150.0	3436	11.0	70	
3	16.0	8	304.0	150.0	3433	12.0	70	

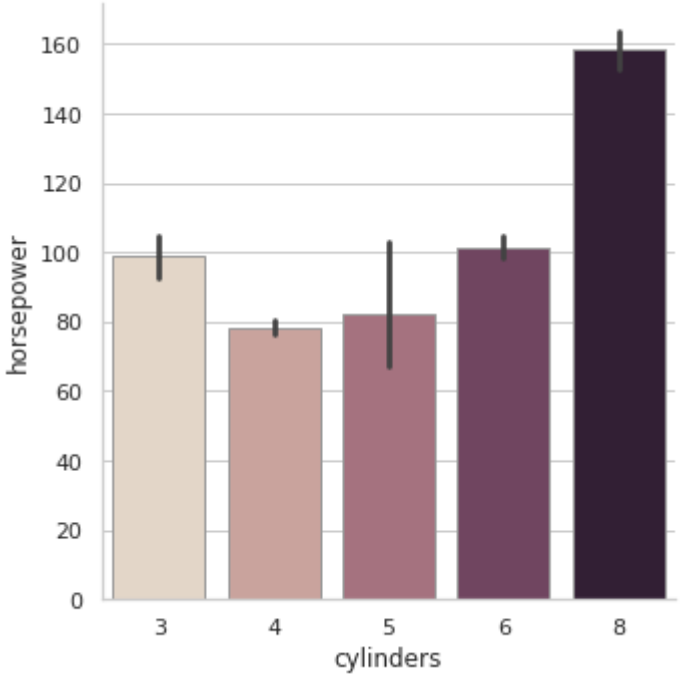
```
sns.catplot(x="origin", y="mpg",
            hue="cylinders", kind="bar",
            data=mpg);
```




```
sns.catplot(x="origin", y="horsepower",
            hue="cylinders", kind="bar",
            palette="ch:.20", data=mpg);
```




```
sns.catplot(x="cylinders", y="horsepower",
            kind="bar", palette="ch:.20", edgecolor=".6",
            data=mpg);
```

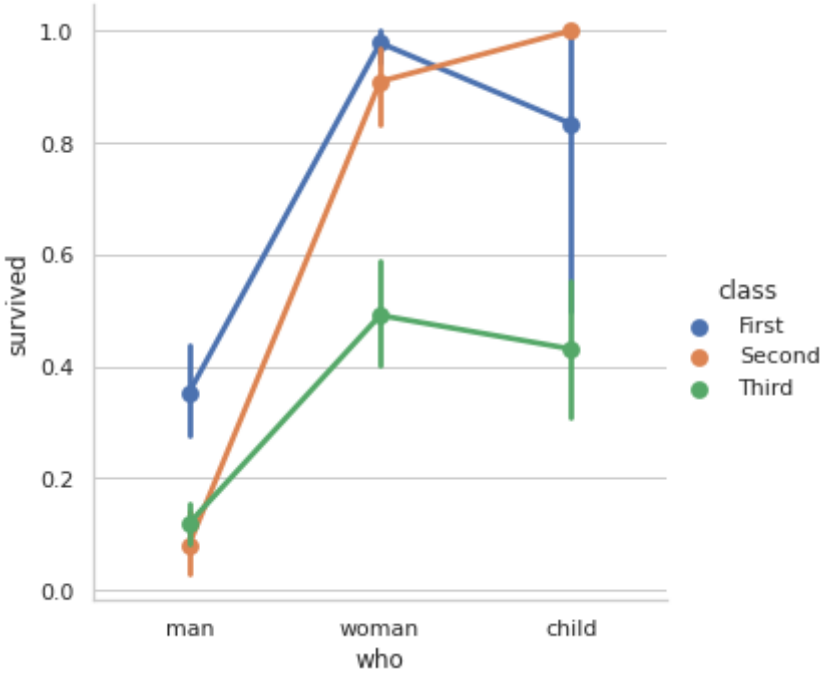


- 포인트 플롯(Point plots)
  - 축의 높이를 사용하여 추정값을 인코딩하여 점 추정값과 신뢰 구간 표시

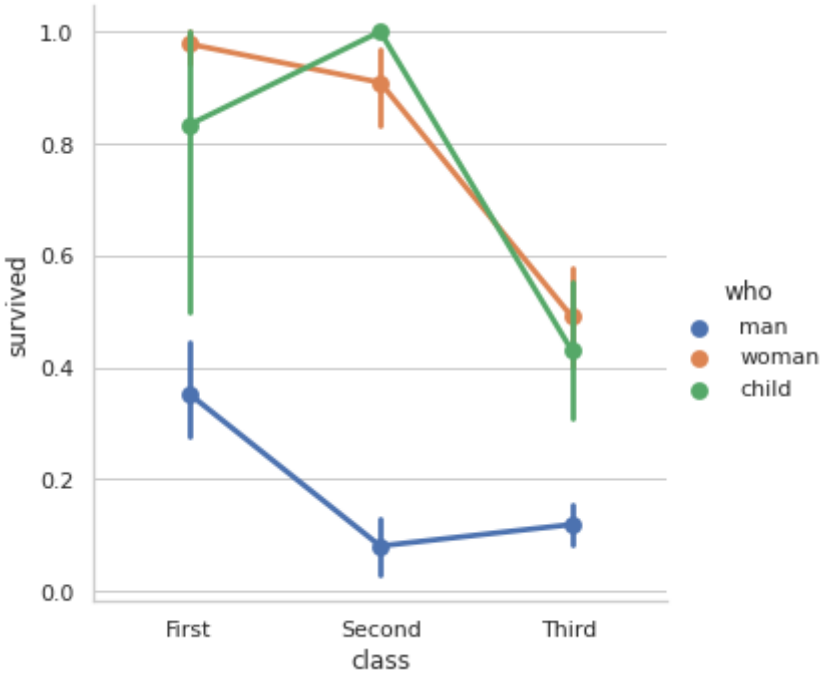
```
titanic = sns.load_dataset("titanic")
titanic
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman

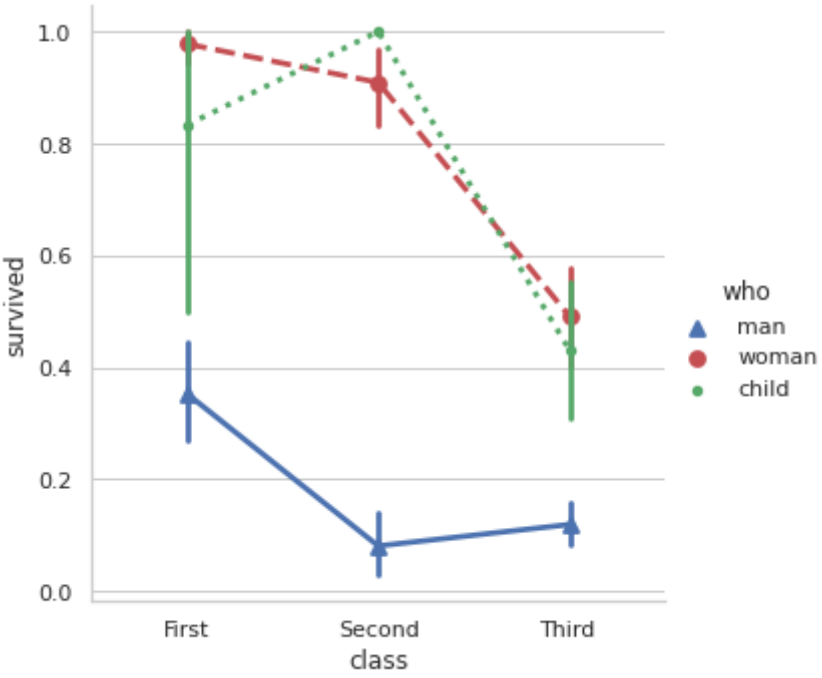
```
sns.catplot(x="who", y="survived",
            hue="class", kind="point",
            data=titanic);
```



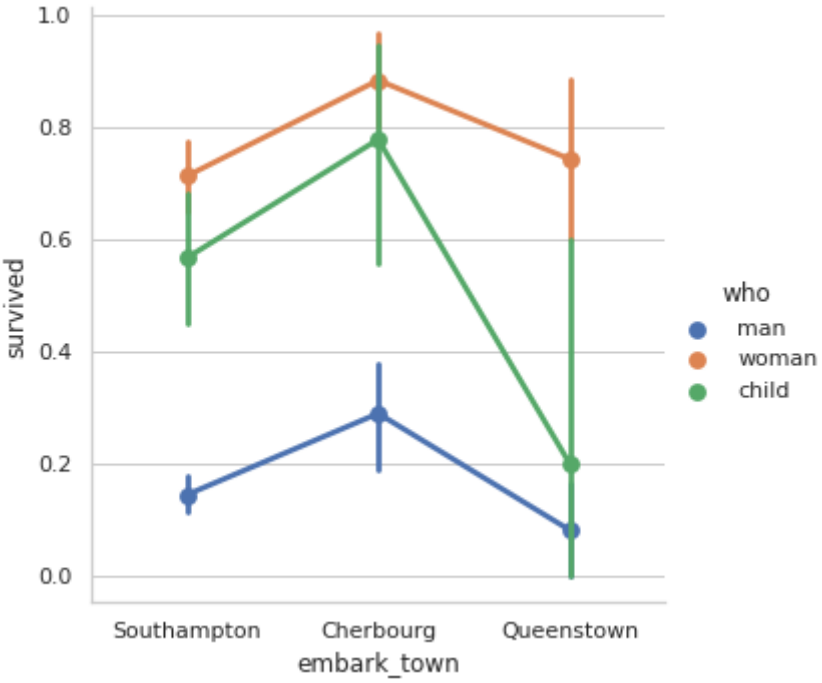

```
sns.catplot(x="class", y="survived",
            hue="who", kind="point",
            data=titanic);
```



```
sns.catplot(x="class", y="survived", hue="who",
            palette={"man": "b", "woman": "r", "child": "g"},
            markers=["^", "o", "."], linestyles=["-", "--", ":"],
            kind="point", data=titanic);
```

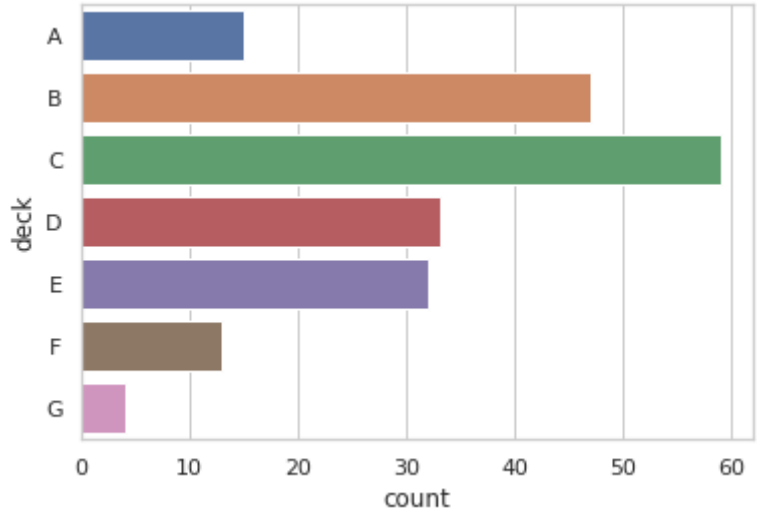



```
sns.catplot(x="embark_town", y="survived",
            hue="who", kind="point",
            data=titanic);
```

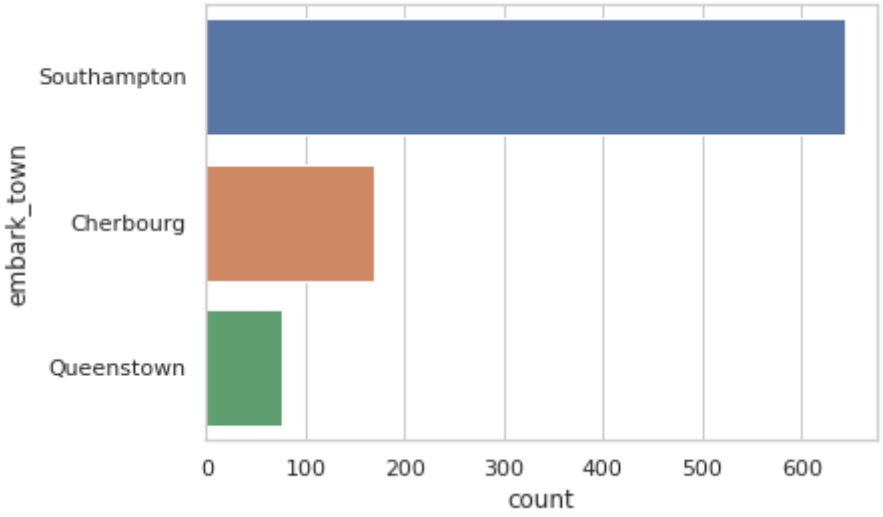


카운트 플롯(Count plots)

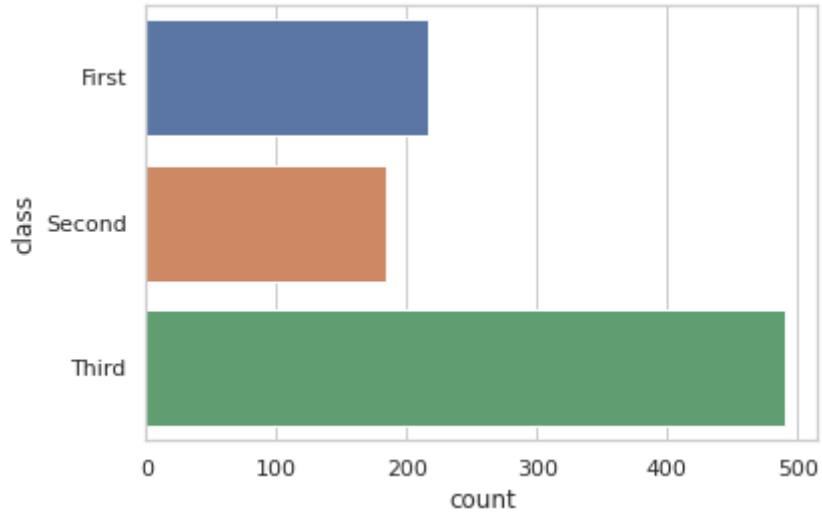
```
sns.countplot(y="deck", data=titanic);
```




```
sns.countplot(y="embark_town", data=titanic);
```



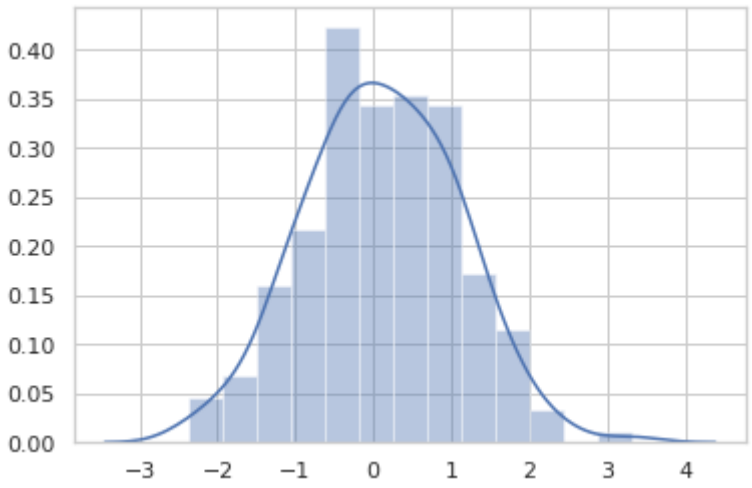

```
sns.countplot(y="class", data=titanic);
```



분포 시각화(Distribution Visualization)

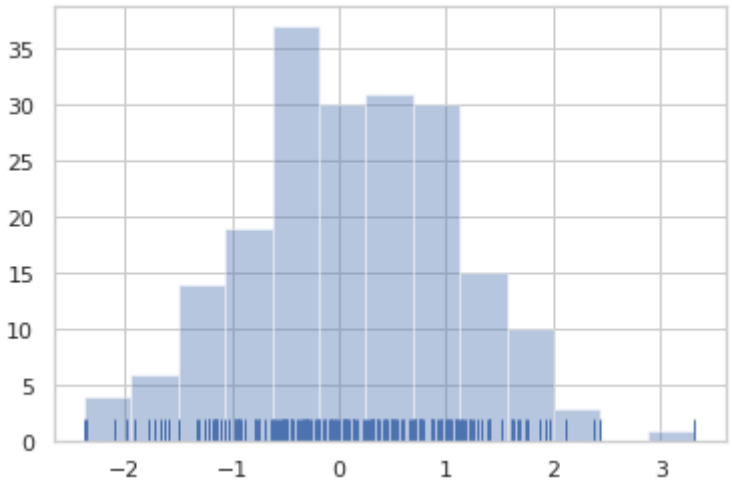
일변량 분포(Univariate distributions)

```
x = np.random.randn(200)
sns.distplot(x);
```



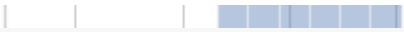
히스토그램(Histograms)

```
sns.distplot(x, kde=False, rug=True);
```

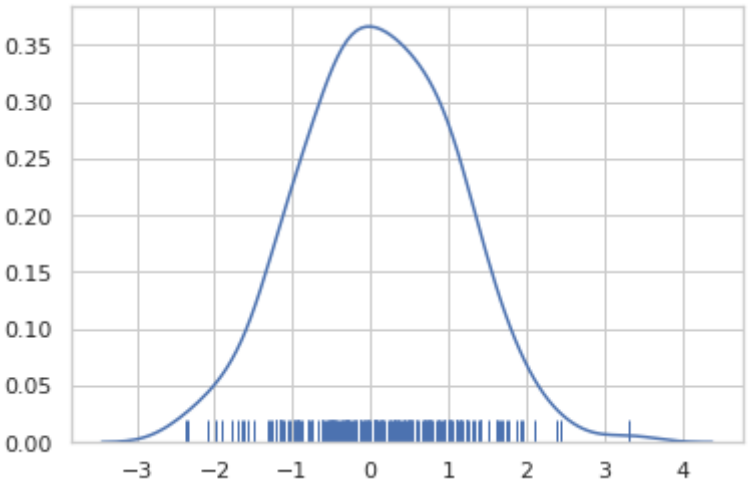



```
sns.distplot(x, bins=20, kde=False, rug=True);
```

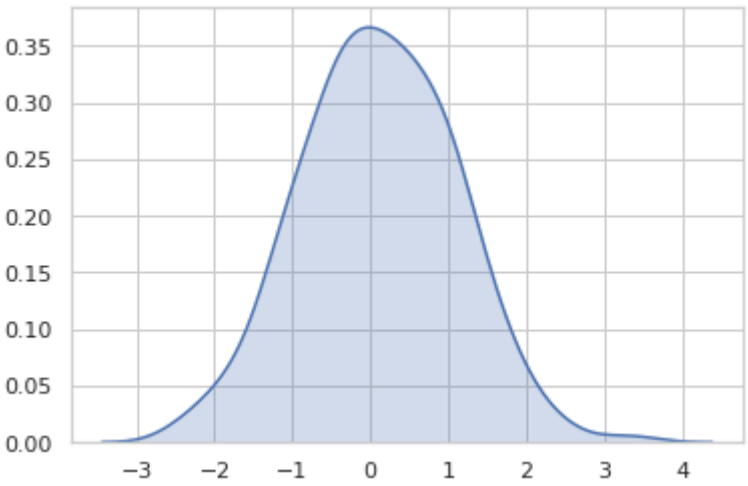
커널 밀도 추정(Kernel density estimation)



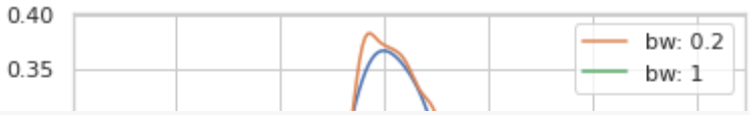
```
sns.distplot(x, hist=False, rug=True);
```



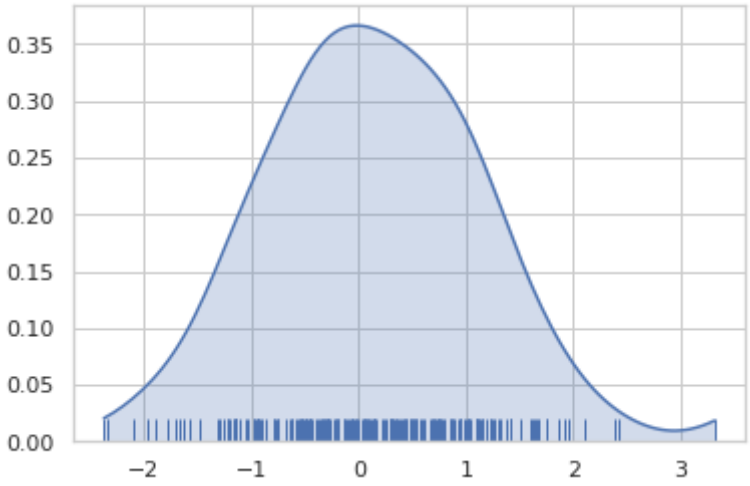

```
sns.kdeplot(x, shade=True);
```



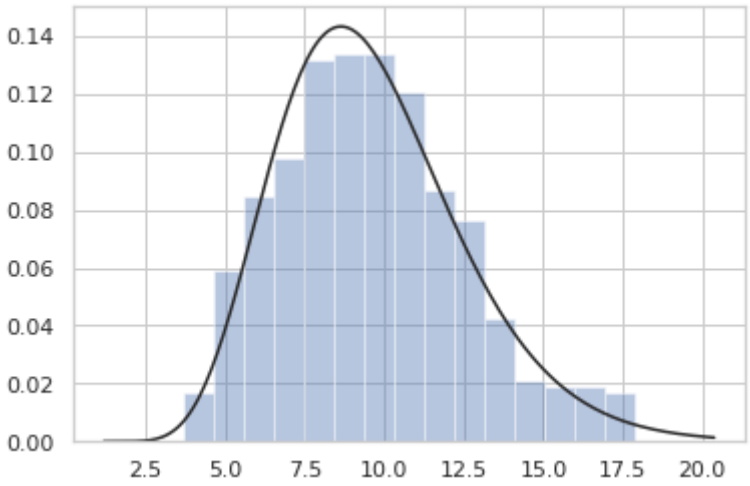

```
sns.kdeplot(x)
sns.kdeplot(x, bw=.2, label="bw: 0.2")
sns.kdeplot(x, bw=1, label="bw: 1")
plt.legend();
```




```
sns.kdeplot(x, shade=True, cut=0)
sns.rugplot(x);
```




```
x = np.random.gamma(10, size=500)
sns.distplot(x, kde=False, fit=stats.gamma);
```



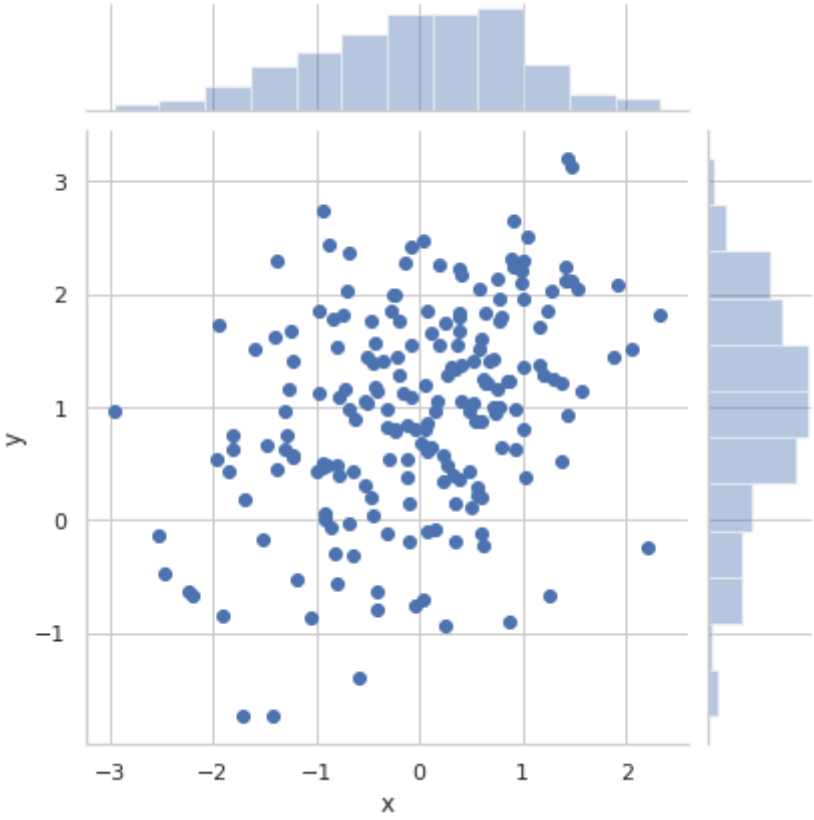
이변량 분포(Bivariate distributions)

산점도(Scatterplots)

- `jointplot` : 두 개의 변수 간의 이변량(또는 joint) 관계와 별도의 축에 각각의 일변량(또는 marginal) 분포가 모두 표시되는 다중 패널 플롯 생성

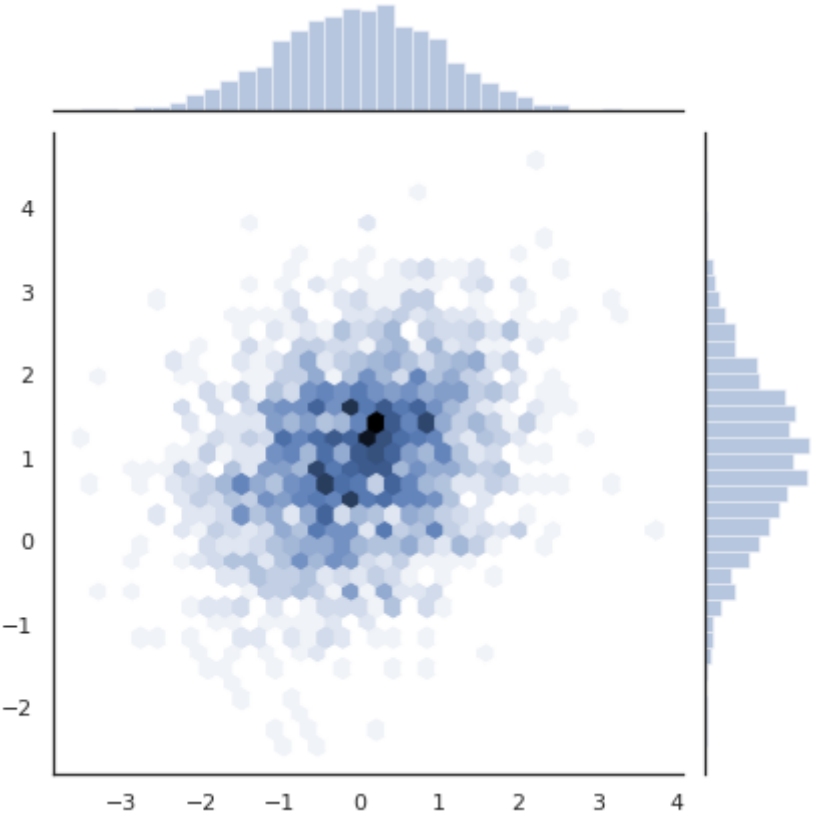
```
mean = [0, 1]
cov = [(1, .3), (.3, 1)]
data = np.random.multivariate_normal(mean, cov, 200)
df = pd.DataFrame(data, columns=["x", "y"])₩

sns.jointplot(x="x", y="y", data=df);
```

```
sns.jointplot("x", "y", data=df);
```



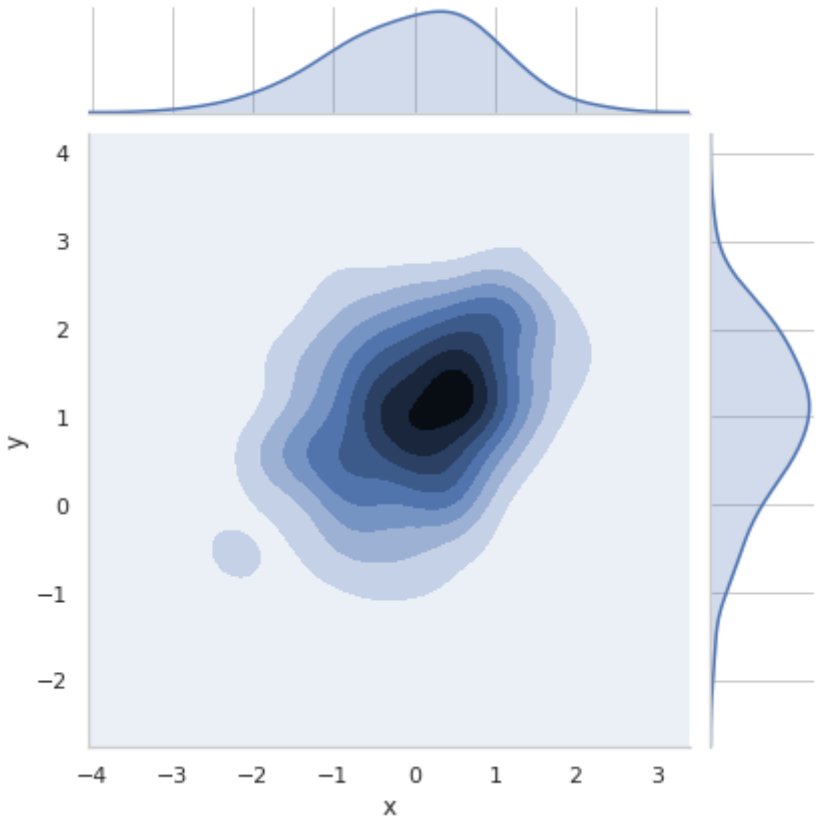
육각 빈 플롯(Hexbin plots)

```
x, y = np.random.multivariate_normal(mean, cov, 2000).T
with sns.axes_style("white"):
  sns.jointplot(x=x, y=y, kind="hex")
```

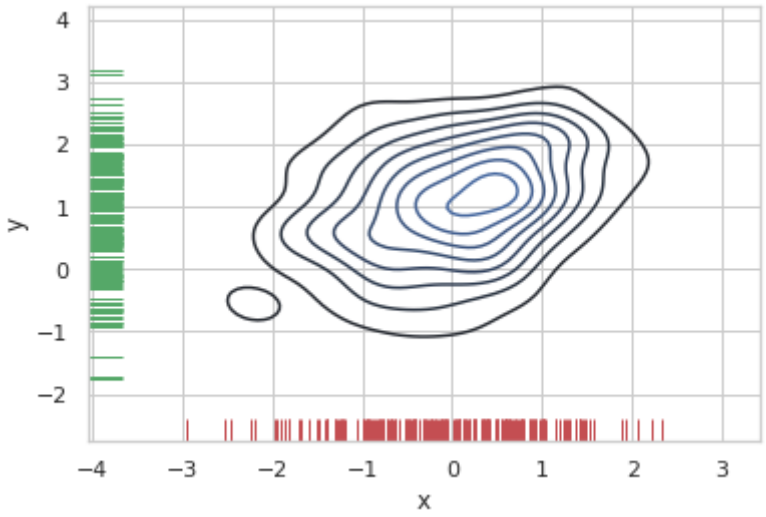


커널 밀도 추정(Kernel density estimation)

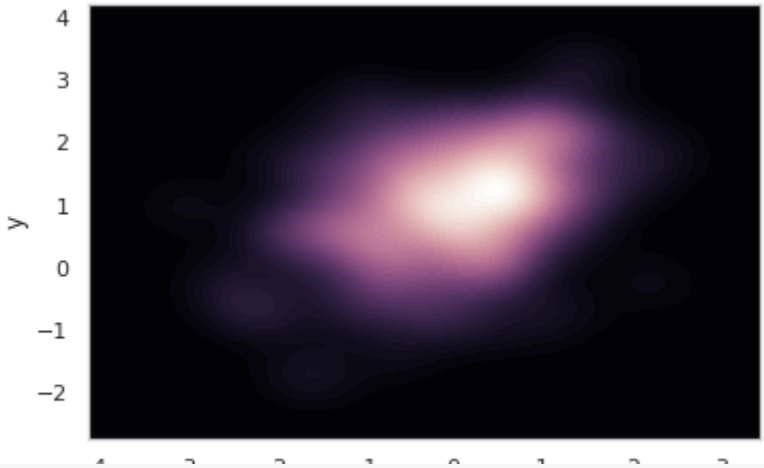
```
sns.jointplot(x="x", y="y", data=df, kind="kde");
```



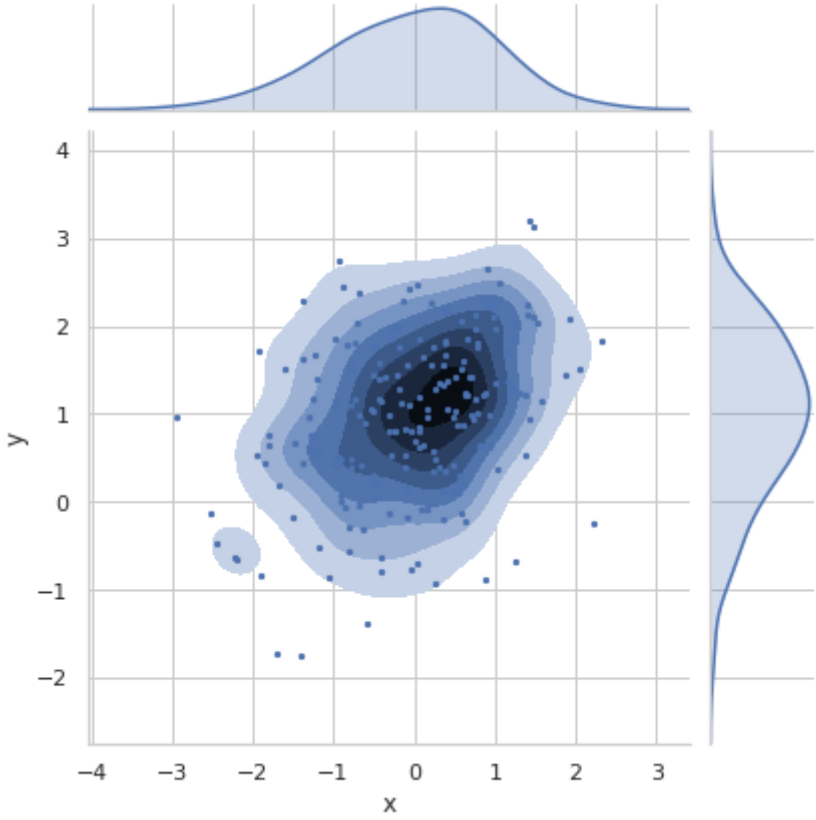

```
sns.kdeplot(df.x, df.y)
sns.rugplot(df.x, color='r')
sns.rugplot(df.y, color='g', vertical=True);
```




```
cmap = sns.cubehelix_palette(as_cmap=True, dark=0, light=1, reverse=True)
sns.kdeplot(df.x, df.y, cmap=cmap, n_levels=60, shade=True);
```




```
g = sns.jointplot(x="x", y="y", data=df, kind="kde")
g.plot_joint(plt.scatter, s=20, linewidth=1, marker=".")
g.ax_joint.collections[0].set_alpha(0)
```



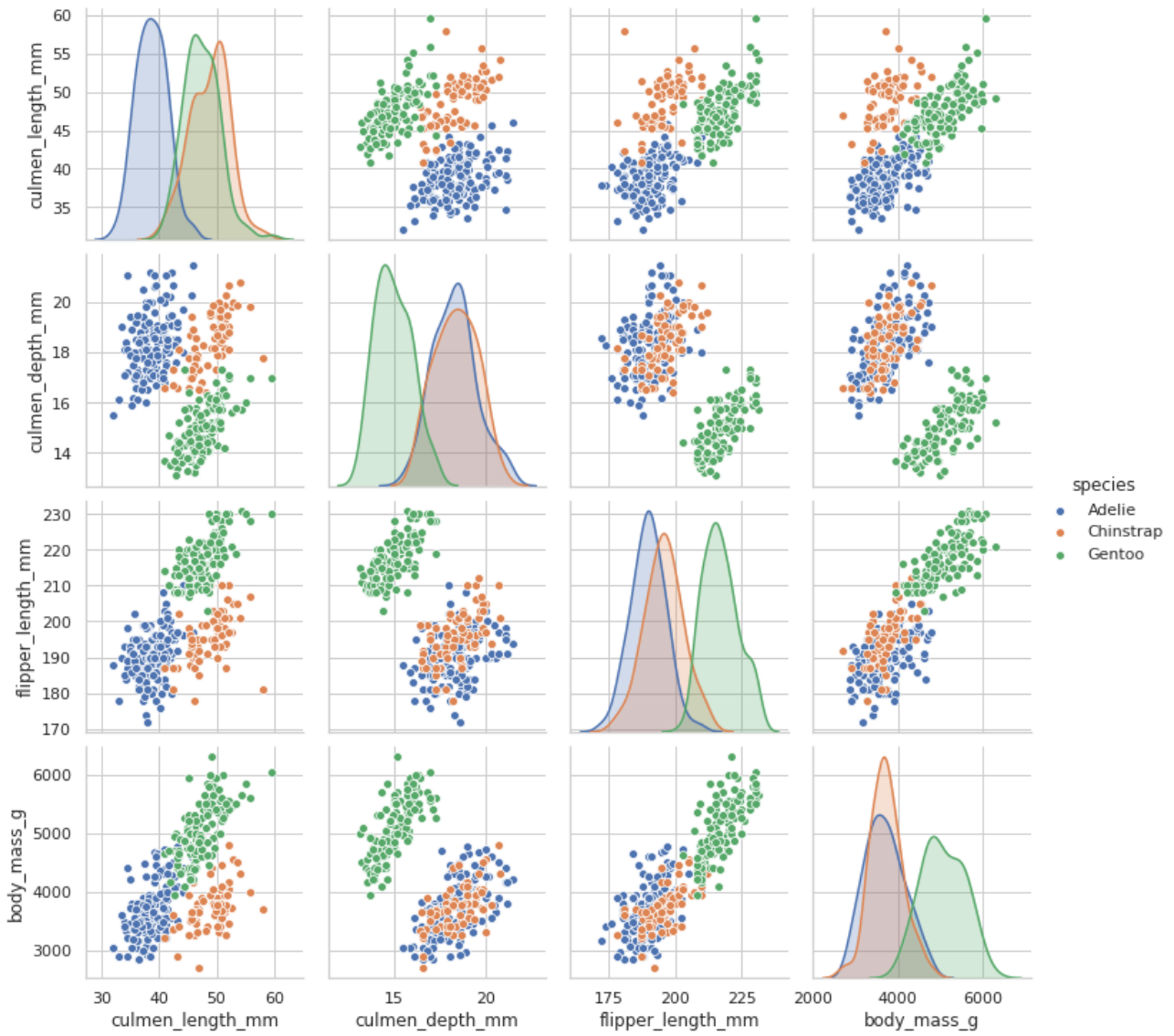
페어와이즈 관계 시각화(Visualizing pairwise relationships)

```
penguins
```

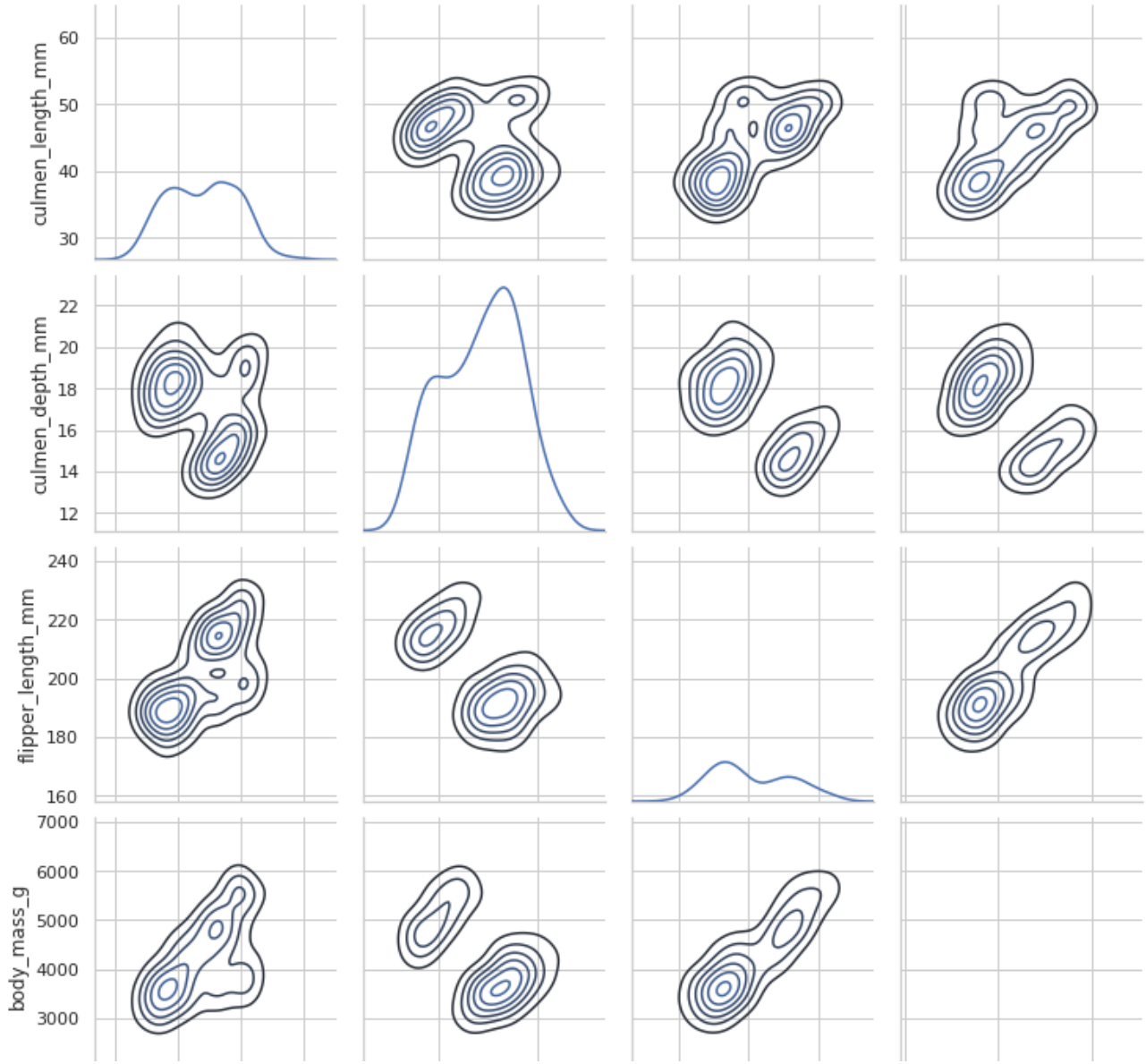
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	

```
sns.pairplot(penguins);
```

```
sns.pairplot(penguins, hue="species");
```

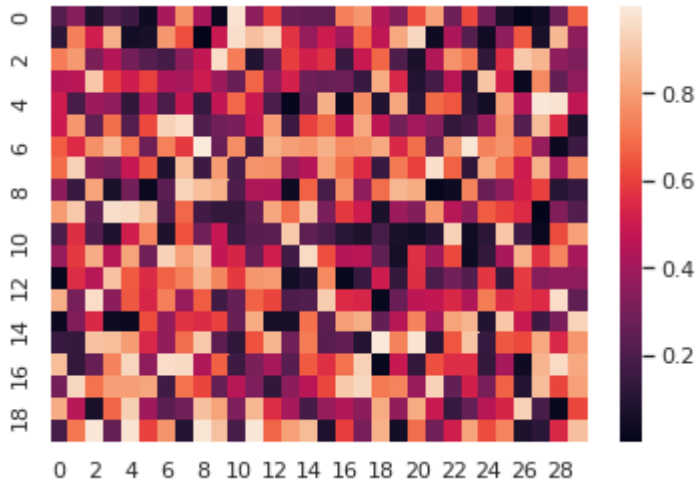



```
g = sns.PairGrid(penguins)
g.map_diag(sns.kdeplot)
g.map_offdiag(sns.kdeplot, n_levels=6);
```

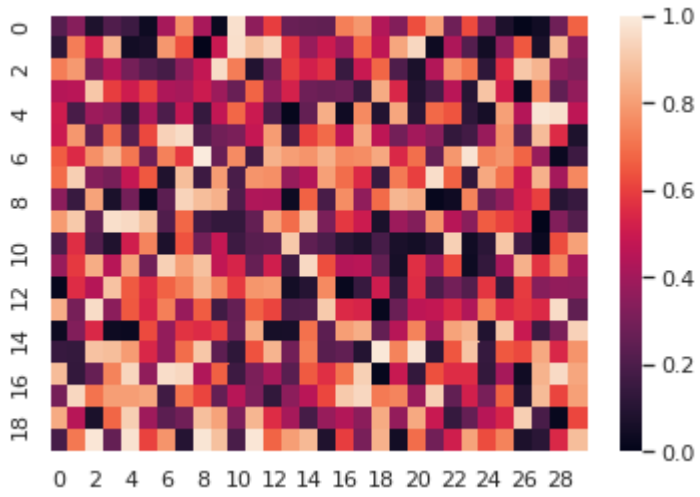


히트맵(Heat Map) & 클러스터맵(Cluster Map)

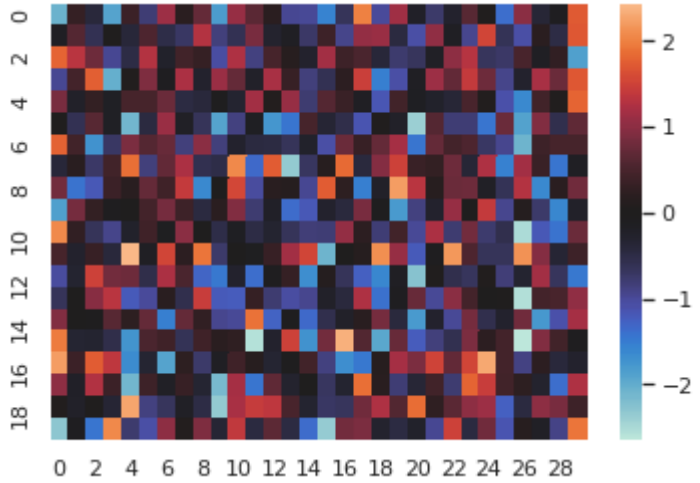
```
udata = np.random.rand(20, 30)
sns.heatmap(udata);
```



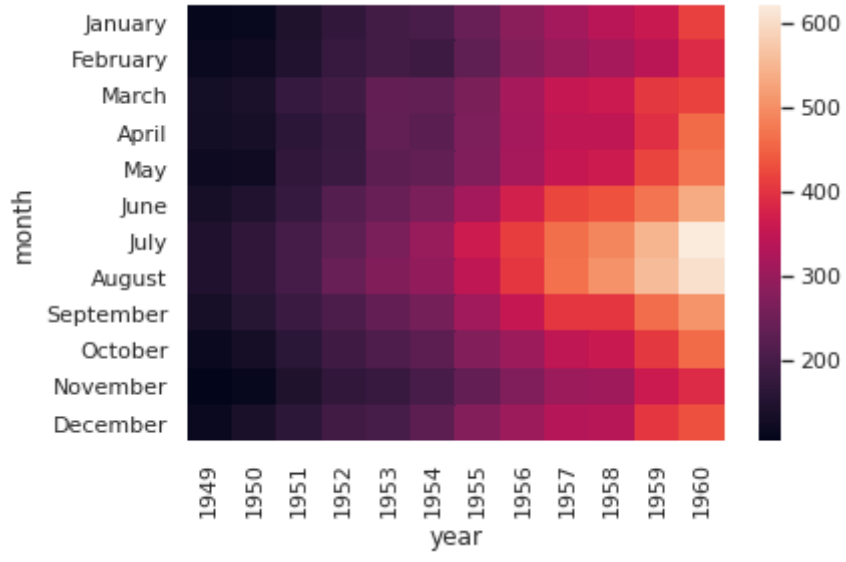
```
sns.heatmap(udata, vmin=0, vmax=1);
```



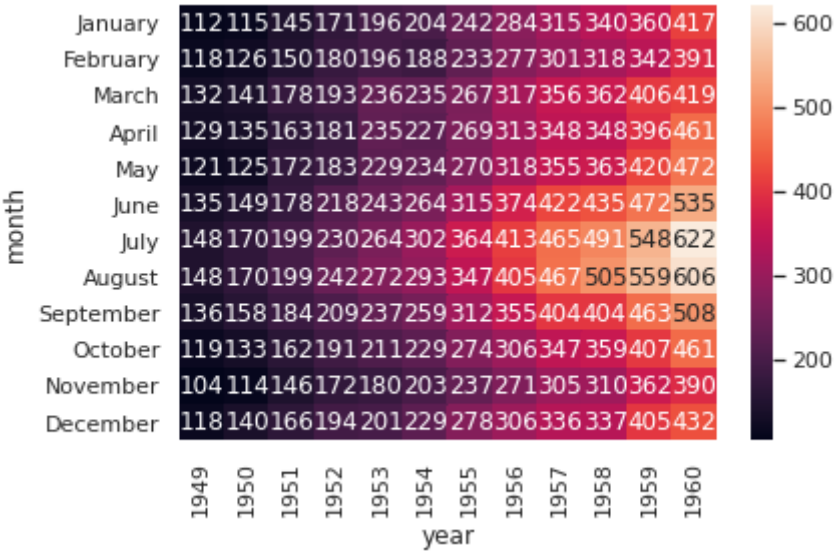

```
ndata = np.random.randn(20, 30)
sns.heatmap(ndata, center=0);
```



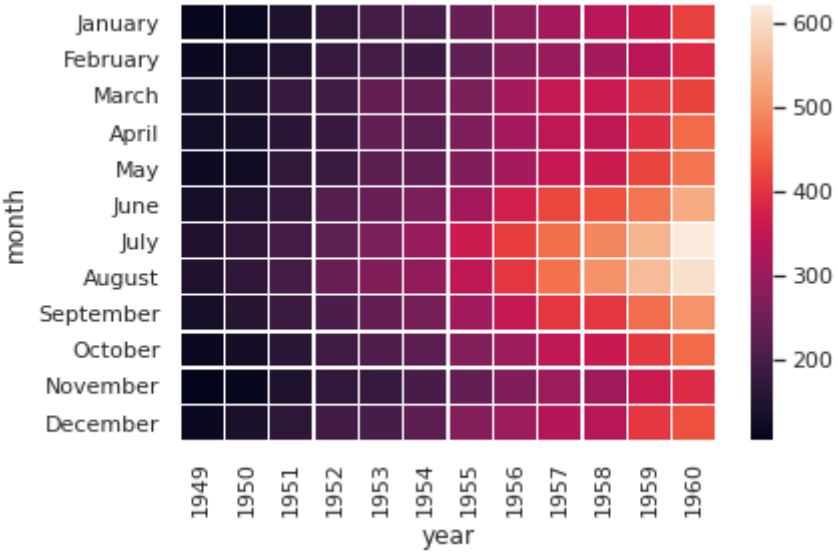

```
flights = sns.load_dataset("flights")
flights = flights.pivot("month", "year", "passengers")
sns.heatmap(flights);
```



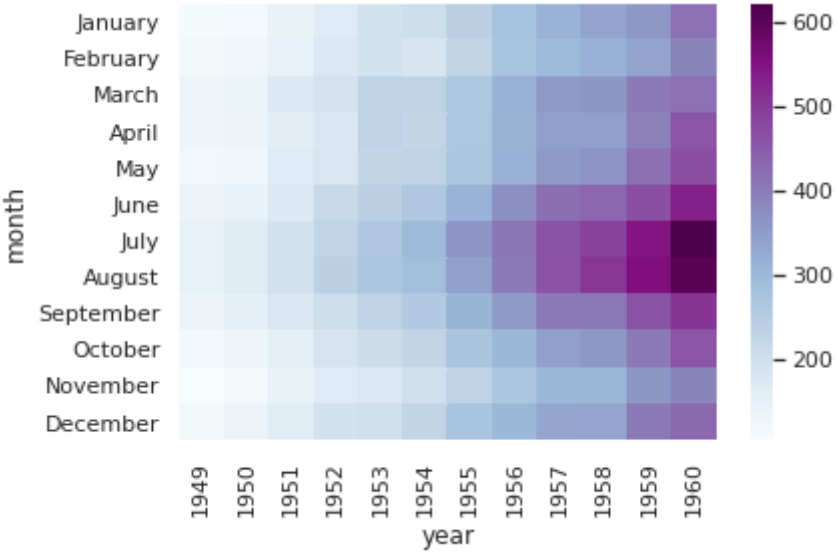

```
sns.heatmap(flights, annot=True, fmt='d');
```



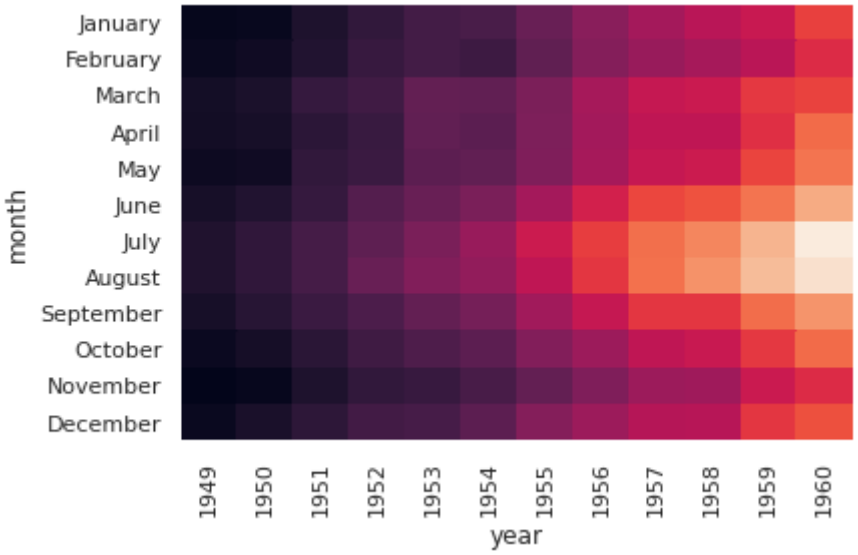

```
sns.heatmap(flights, linewidths=.2);
```



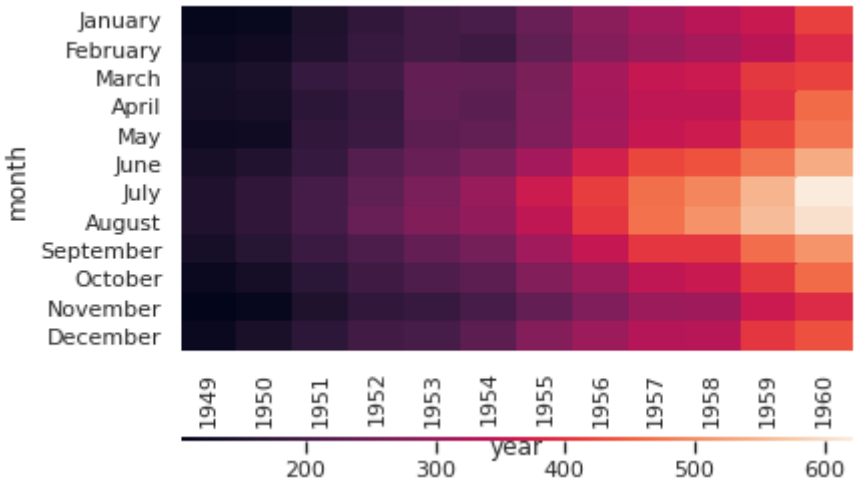

```
sns.heatmap(flights, cmap="BuPu");
```



```
sns.heatmap(flights, cbar=False);
```



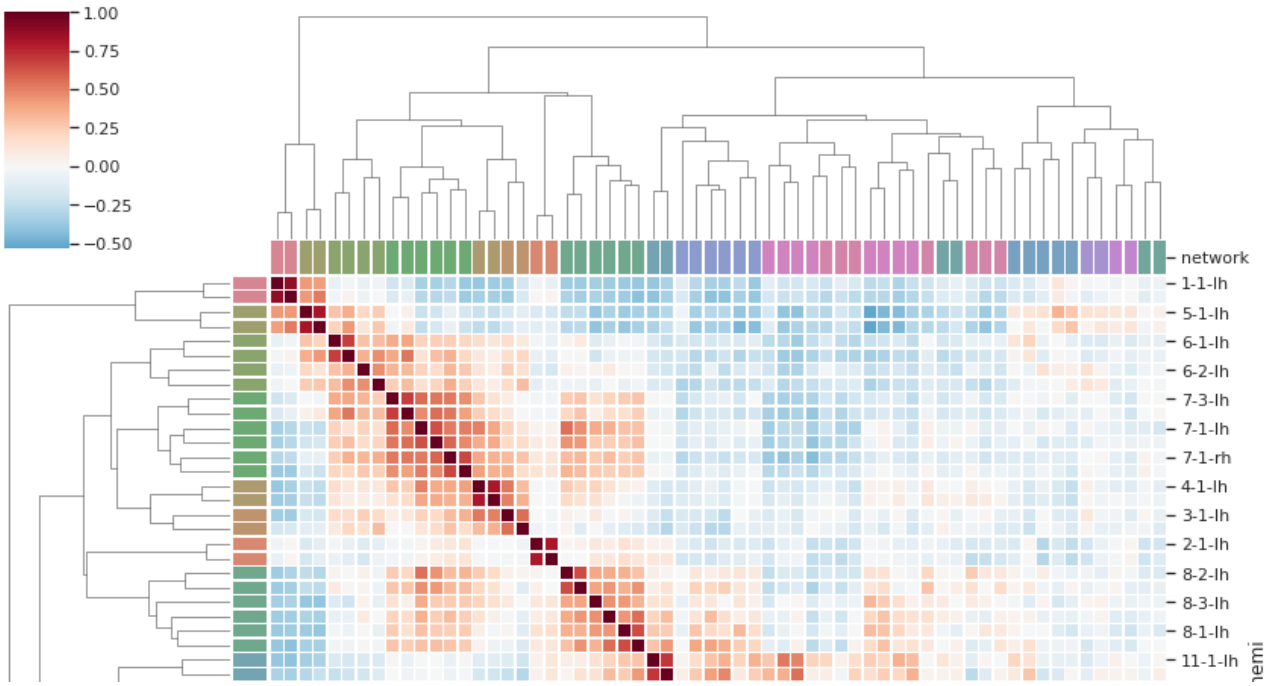

```
grid_kws = {"height_ratios": (.9, 0.01), "hspace": .5}
f, (ax, cbar_ax) = plt.subplots(2, gridspec_kw=grid_kws)
ax = sns.heatmap(flights, ax=ax,
                 cbar_ax=cbar_ax,
                 cbar_kws={"orientation":"horizontal"})
```




```
brain_networks = sns.load_dataset("brain_networks", header=[0, 1, 2], index_col=0)
brain_networks
```

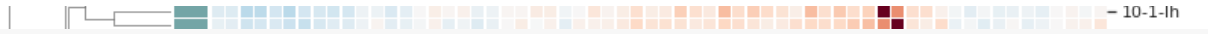
network	1		2		3		4
node	1		1		1		1
hemi	lh	rh	lh	rh	lh	rh	lh
0	56.055744	92.031036	3.391576	38.659683	26.203819	-49.715569	47.461(
1	55.547253	43.690075	-65.495987	-13.974523	-28.274963	-39.050129	-1.2106
2	60.997768	63.438793	-51.108582	-13.561346	-18.842947	-1.214659	-65.575(
3	18.514868	12.657158	-34.576603	-32.665958	-7.420454	17.119448	-41.800(
4	-2.527392	-63.104668	-13.814151	-15.837989	-45.216927	3.483550	-62.613(

```
networks = brain_networks.columns.get_level_values("network")
used_networks = np.arange(1, 18)
used_columns = (networks.astype(int).isin(used_networks))
brain_networks = brain_networks.loc[:, used_columns]

network_pal = sns.husl_palette(17, s=.5)
network_lut = dict(zip(map(str, used_networks), network_pal))
network_colors = pd.Series(networks, index=brain_networks.columns).map(network_lut)

sns.clustermap(brain_networks.corr(), center=0, cmap="RdBu_r",
               row_colors=network_colors, col_colors=network_colors,
               linewidth=.5, figsize=(12, 12));
```



선형 관계 시각화(Visualizing linear relationships)

선형 회귀 모델 시각화 함수

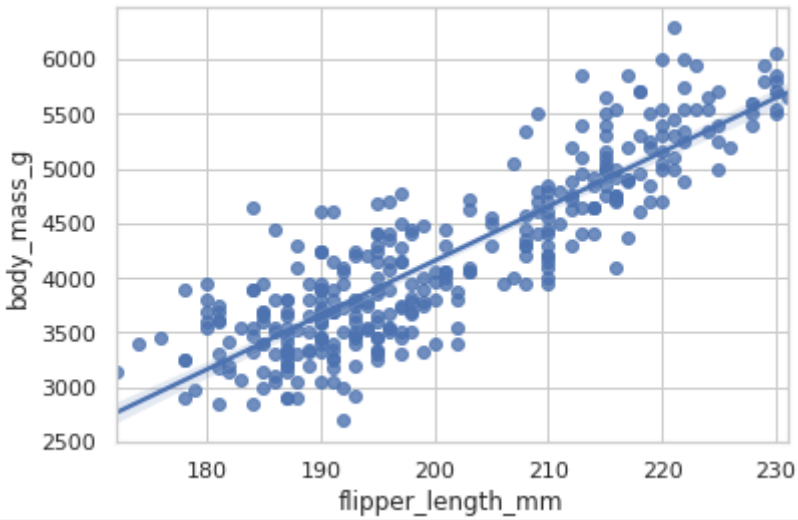



```
penguins
```

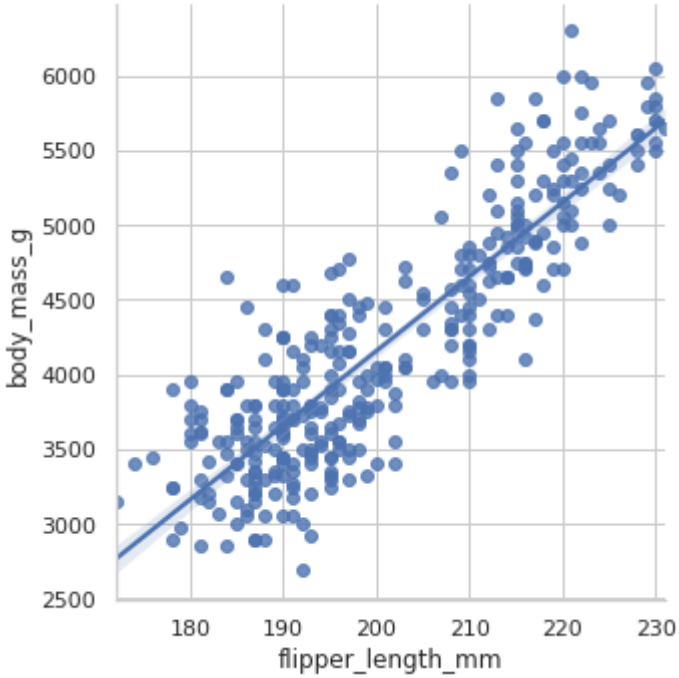
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	
340	Gentoo	Biscoe	46.8	14.3	215.0	
341	Gentoo	Biscoe	50.4	15.7	222.0	
342	Gentoo	Biscoe	45.2	14.8	212.0	
343	Gentoo	Biscoe	49.9	16.1	213.0	

344 rows × 7 columns

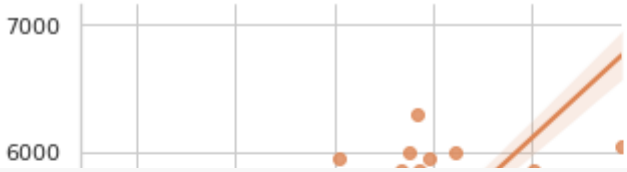
```
sns.regplot(x="flipper_length_mm", y="body_mass_g",
            data=penguins);
```



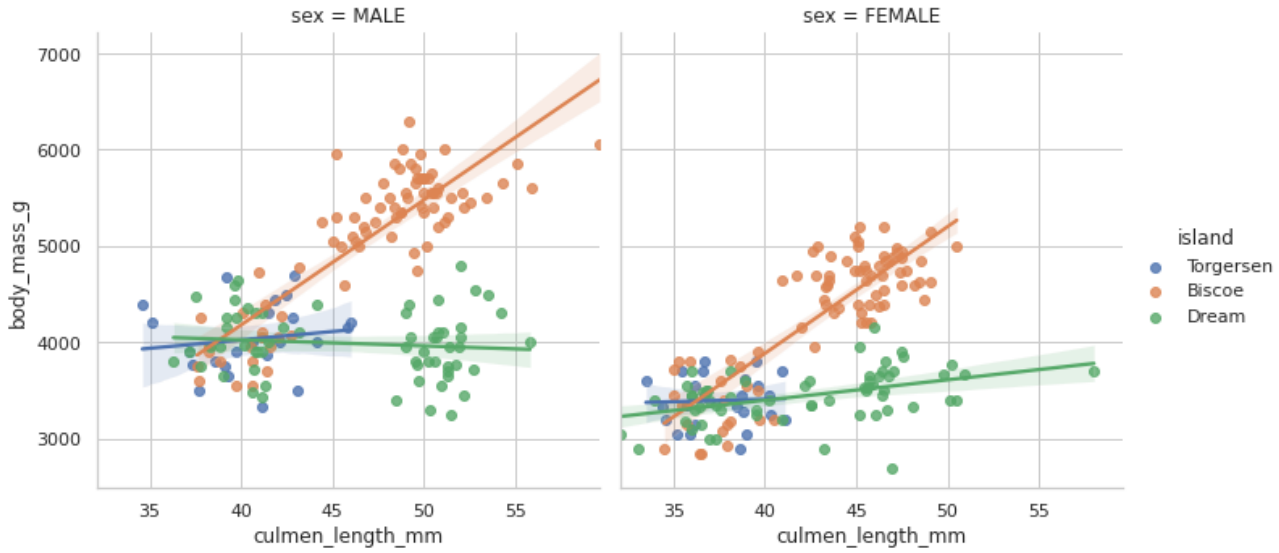

```
sns.lmplot(x="flipper_length_mm", y="body_mass_g",
           data=penguins);
```



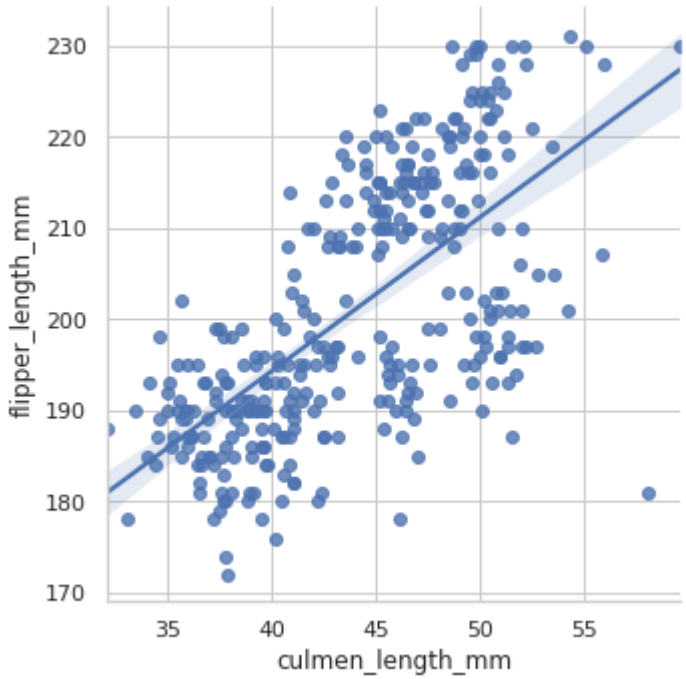

```
sns.lmplot(x="culmen_length_mm", y="body_mass_g",
           hue="island", data=penguins);
```



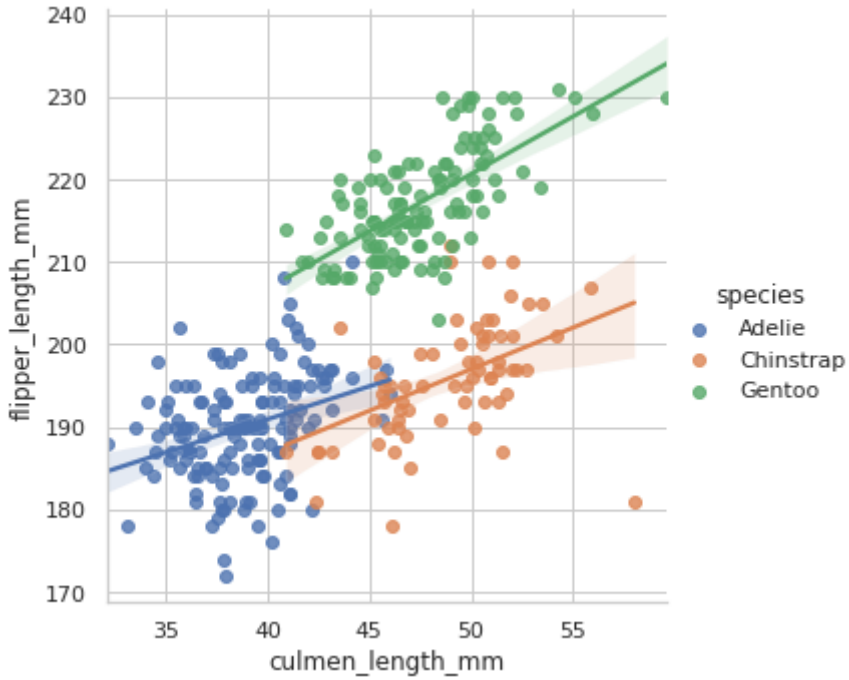

```
sns.lmplot(x="culmen_length_mm", y="body_mass_g",
           col="sex", hue="island", data=penguins);
```



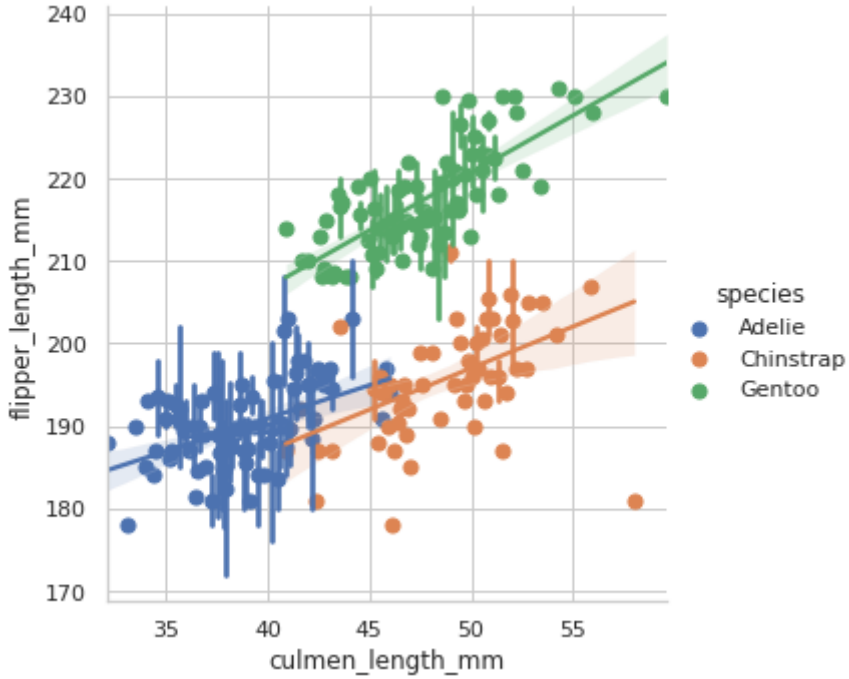

```
sns.lmplot(x="culmen_length_mm", y="flipper_length_mm",
           data=penguins);
```



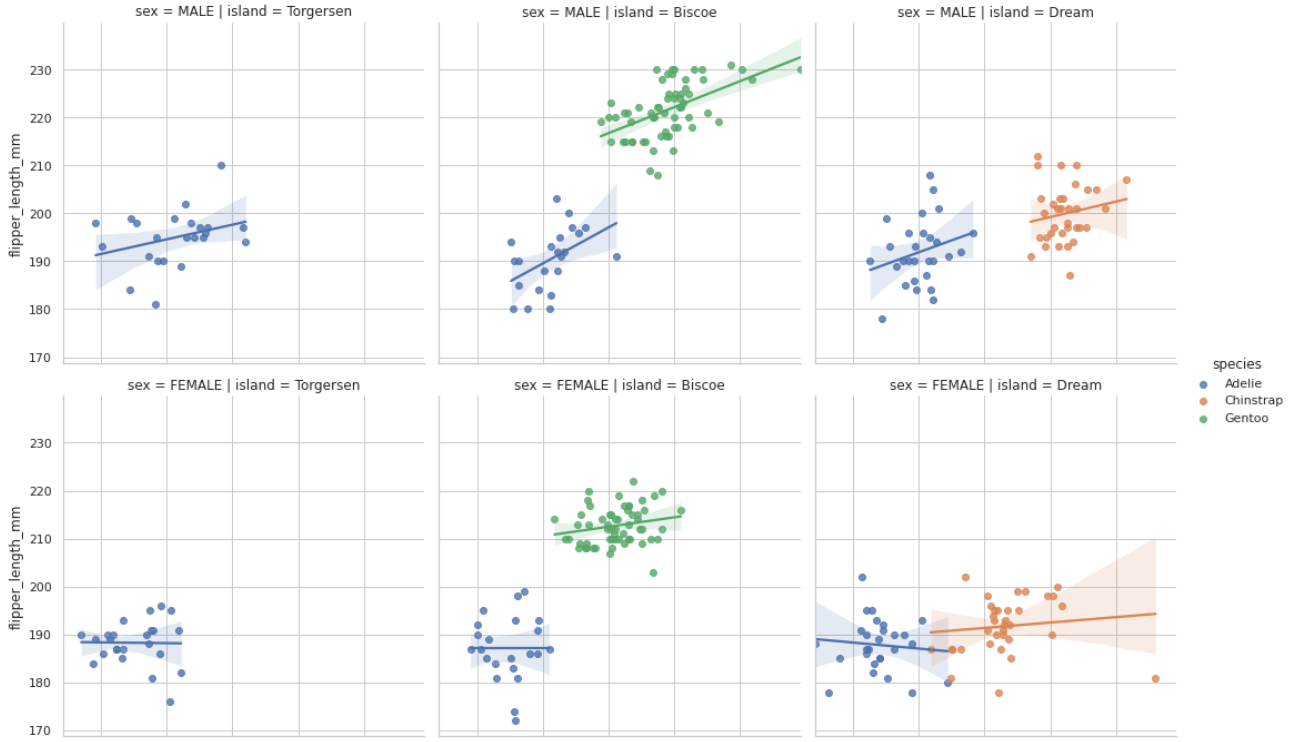

```
sns.lmplot(x="culmen_length_mm", y="flipper_length_mm",
           hue="species", data=penguins);
```




```
sns.lmplot(x="culmen_length_mm", y="flipper_length_mm",
           hue="species", x_estimator=np.mean, data=penguins);
```




```
sns.lmplot(x="culmen_length_mm", y="flipper_length_mm",
           col="island", row="sex", hue="species",
           data=penguins);
```

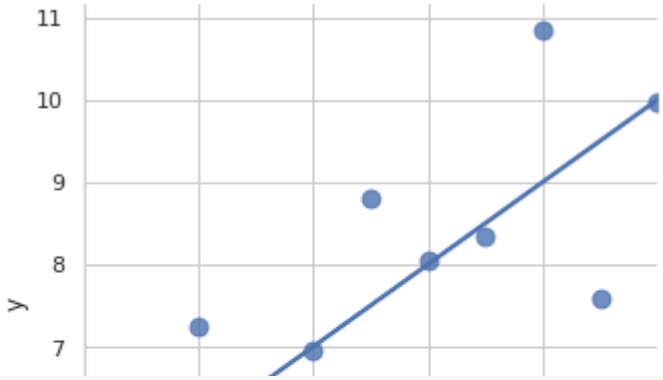


다른 종류의 모델

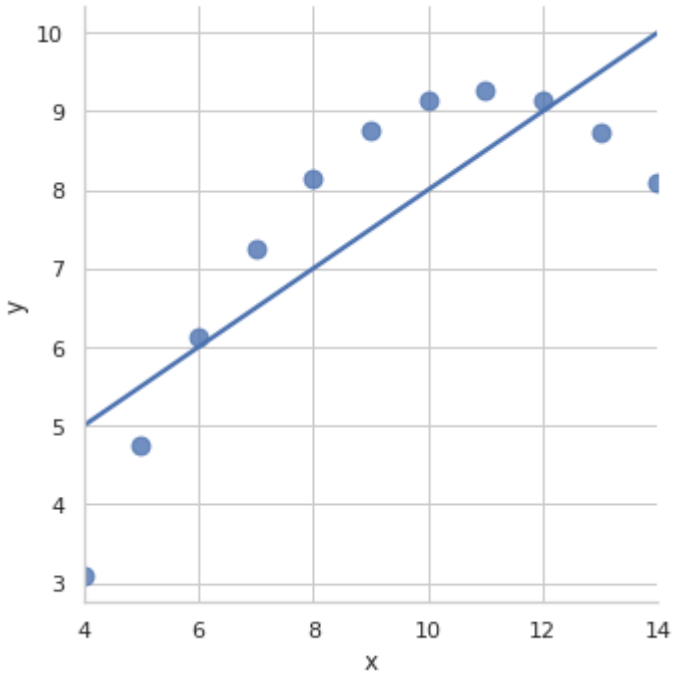
```
anscombe = sns.load_dataset("anscombe")
anscombe.describe()
```

	x	y
count	44.000000	44.000000
mean	9.000000	7.500682
std	3.198837	1.958925
min	4.000000	3.100000
25%	7.000000	6.117500
50%	8.000000	7.520000
75%	11.000000	8.747500
max	19.000000	12.740000

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'I'"),
           ci=None, scatter_kws={"s": 80});
```

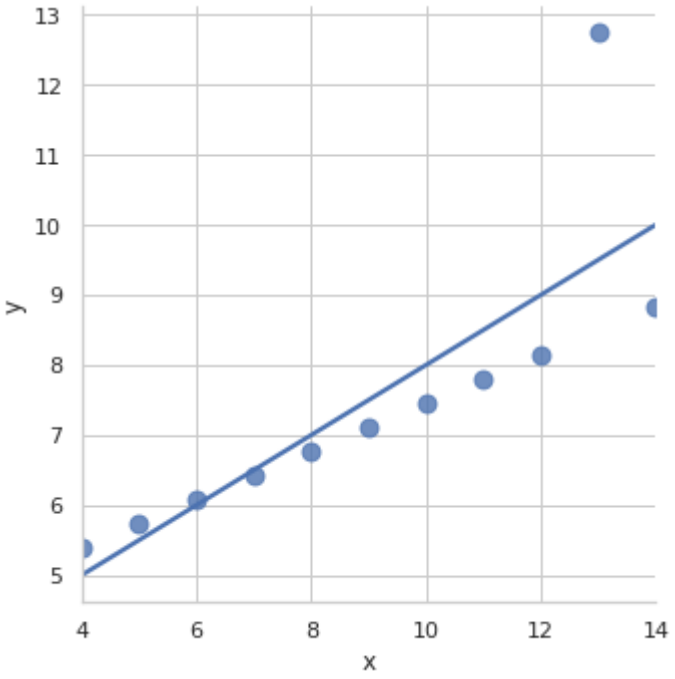



```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
           ci=None, scatter_kws={"s": 80});
```

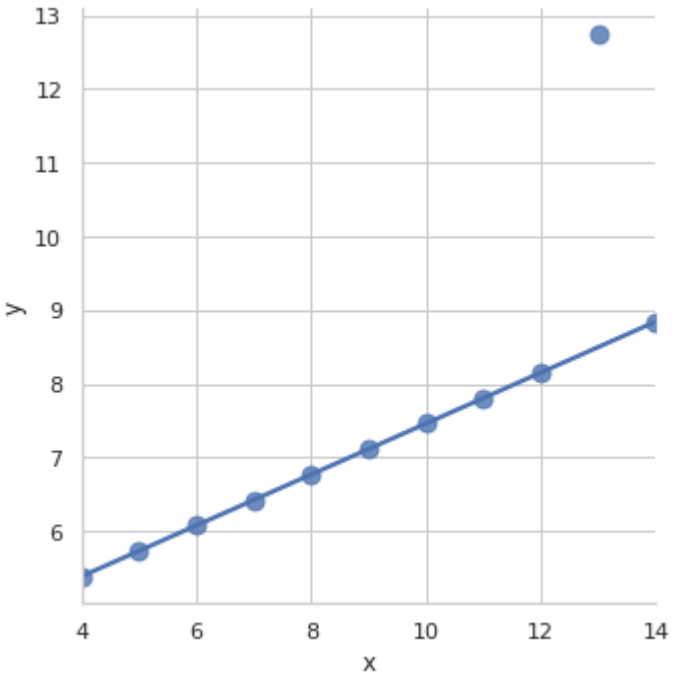



```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
           order=2, ci=None, scatter_kws={"s": 80});
```

```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'III'"),
           ci=None, scatter_kws={"s": 80});
```




```
sns.lmplot(x="x", y="y", data=anscombe.query("dataset == 'III'"),
           robust=True, ci=None, scatter_kws={"s": 80});
```



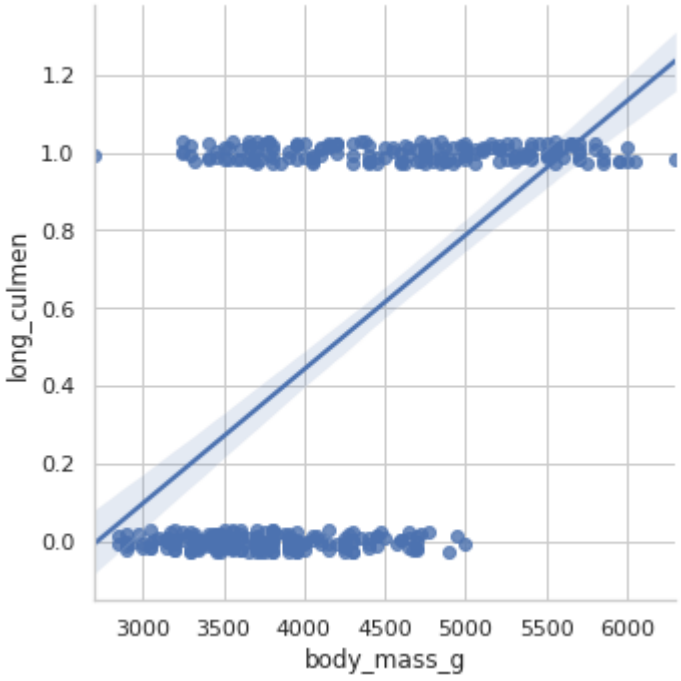

```
penguins
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	
340	Gentoo	Biscoe	46.8	14.3	215.0	
341	Gentoo	Biscoe	50.4	15.7	222.0	

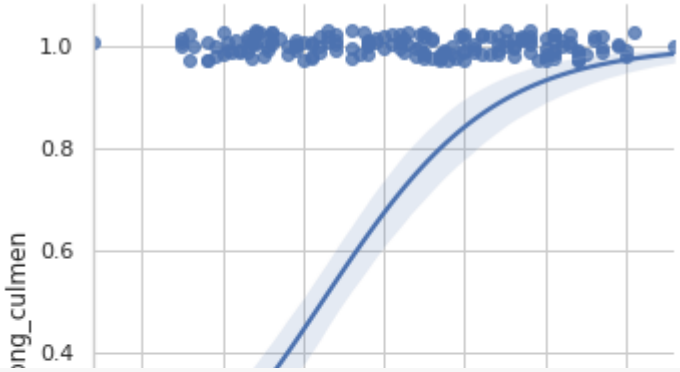
```
penguins["long_culmen"] = (penguins.culmen_length_mm > penguins['culmen_length_mm'].mean())
```

343	Gentoo	Biscoe	49.9	16.1	213.0	

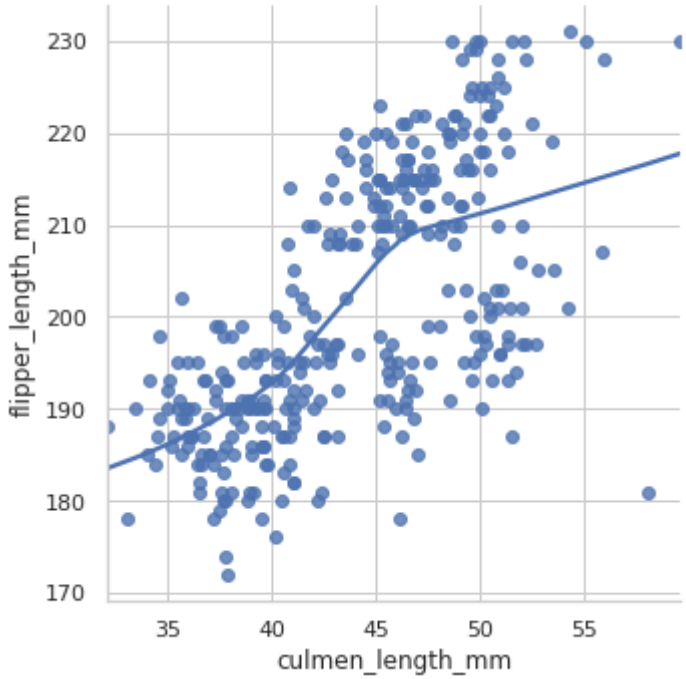
```
sns.lmplot(x="body_mass_g", y="long_culmen",
           y_jitter=.03, data=penguins);
```



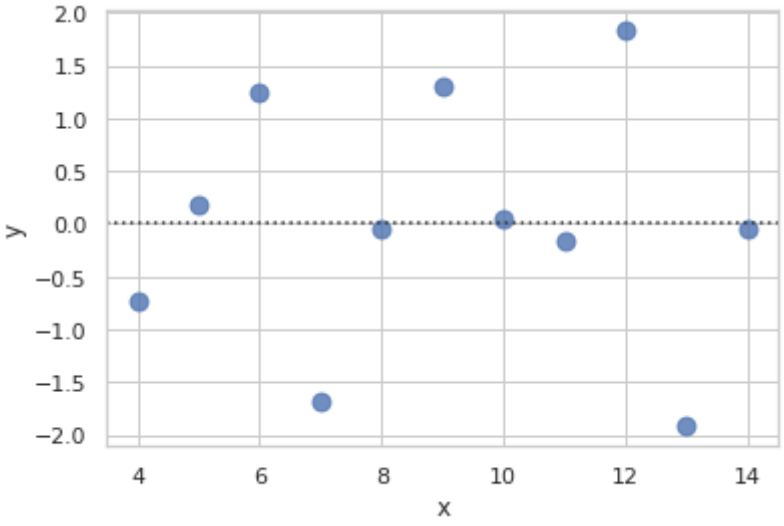

```
sns.lmplot(x="body_mass_g", y="long_culmen",
           logistic=True, y_jitter=.03, data=penguins);
```



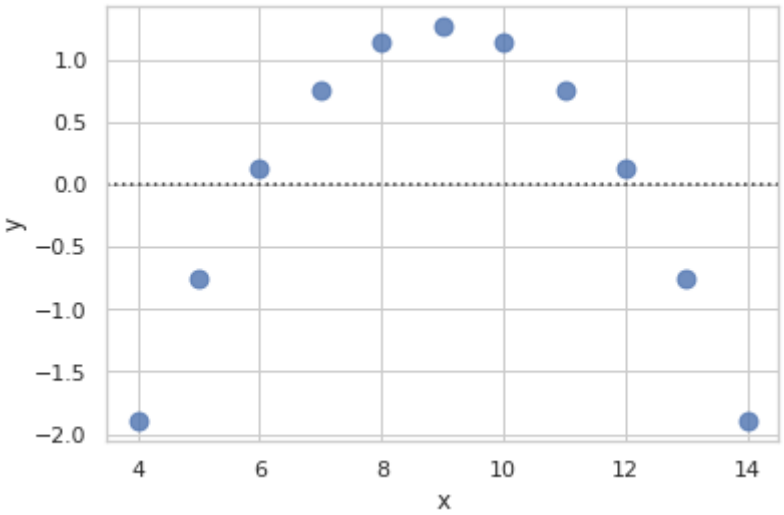

```
sns.lmplot(x="culmen_length_mm", y="flipper_length_mm",
           lowess=True, data=penguins);
```




```
sns.residplot(x="x", y="y", data=anscombe.query("dataset == 'I'"),
              scatter_kws={"s": 80});
```




```
sns.residplot(x="x", y="y", data=anscombe.query("dataset == 'II'"),
              scatter_kws={"s": 80});
```



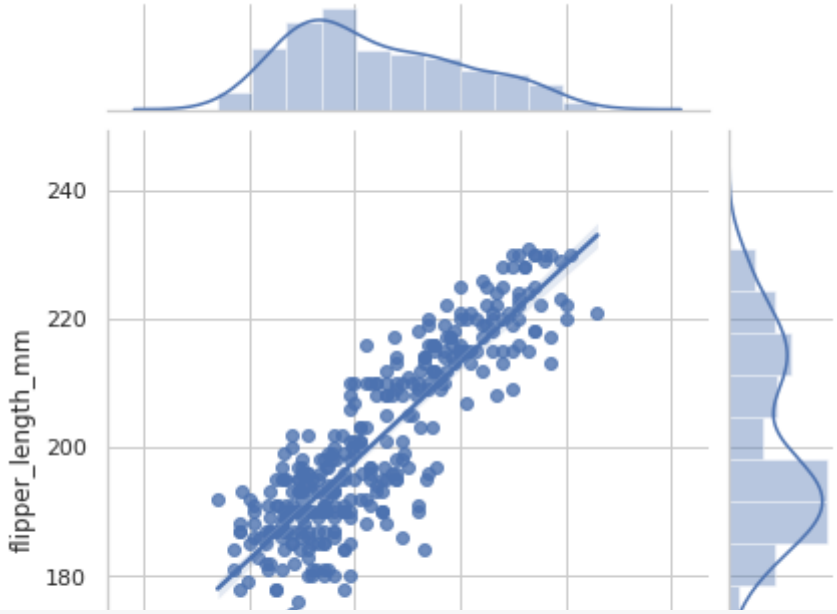
다른 상황의 회귀

```
penguins
```

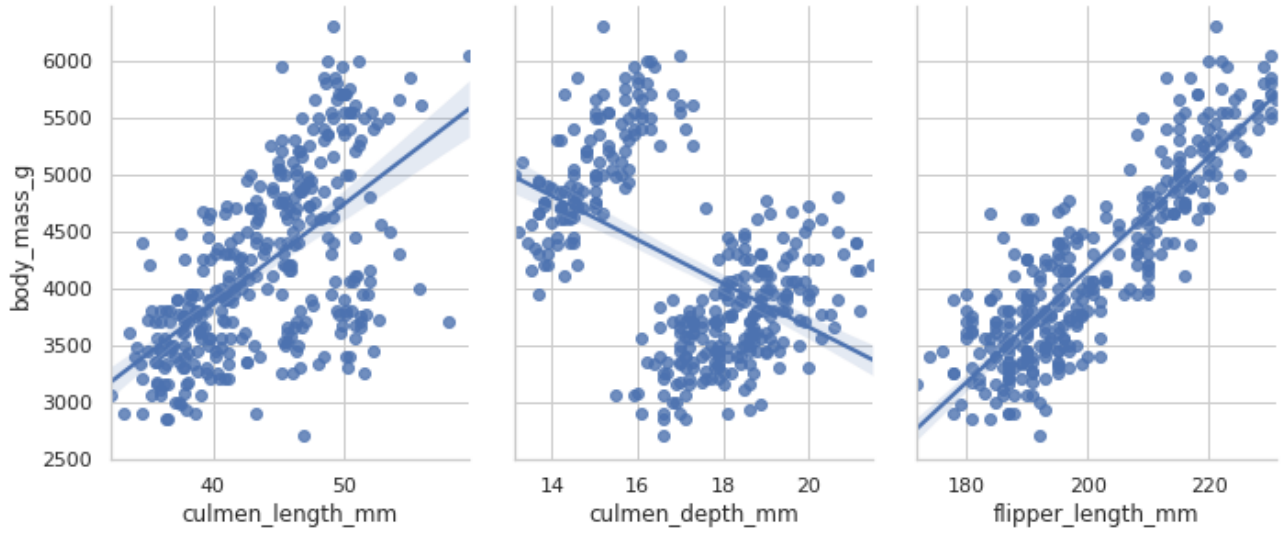
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	
340	Gentoo	Biscoe	46.8	14.3	215.0	
341	Gentoo	Biscoe	50.4	15.7	222.0	
342	Gentoo	Biscoe	45.2	14.8	212.0	
343	Gentoo	Biscoe	49.9	16.1	213.0	

344 rows × 8 columns

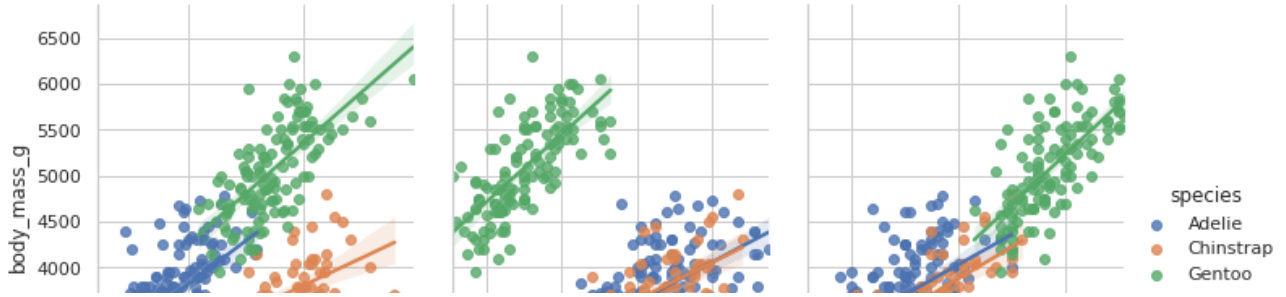
```
sns.jointplot(x="body_mass_g", y="flipper_length_mm",
              kind="reg", data=penguins);
```




```
sns.pairplot(penguins,
             x_vars=["culmen_length_mm", "culmen_depth_mm", "flipper_length_mm"],
             y_vars=["body_mass_g"],
             height=4, aspect=.8,
             kind="reg");
```




```
sns.pairplot(penguins,
             x_vars=["culmen_length_mm", "culmen_depth_mm", "flipper_length_mm"],
             y_vars=["body_mass_g"], hue="species",
             height=4, aspect=.8,
             kind="reg");
```



구조화된 다중 플롯 그리드

FacetGrid

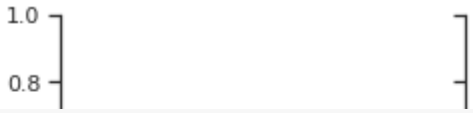
```
penguins
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_m
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	
...	...	...	...	...	...	
339	Gentoo	Biscoe	NaN	NaN	NaN	
340	Gentoo	Biscoe	46.8	14.3	215.0	
341	Gentoo	Biscoe	50.4	15.7	222.0	
342	Gentoo	Biscoe	45.2	14.8	212.0	
343	Gentoo	Biscoe	49.9	16.1	213.0	

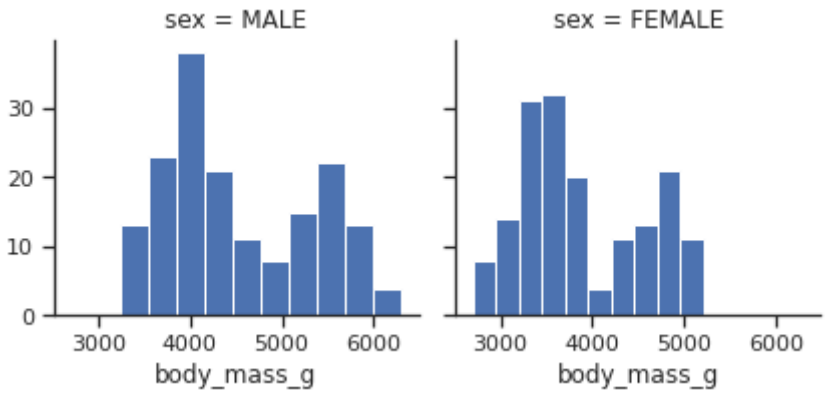
344 rows × 8 columns

```
sns.set(style="ticks")
```

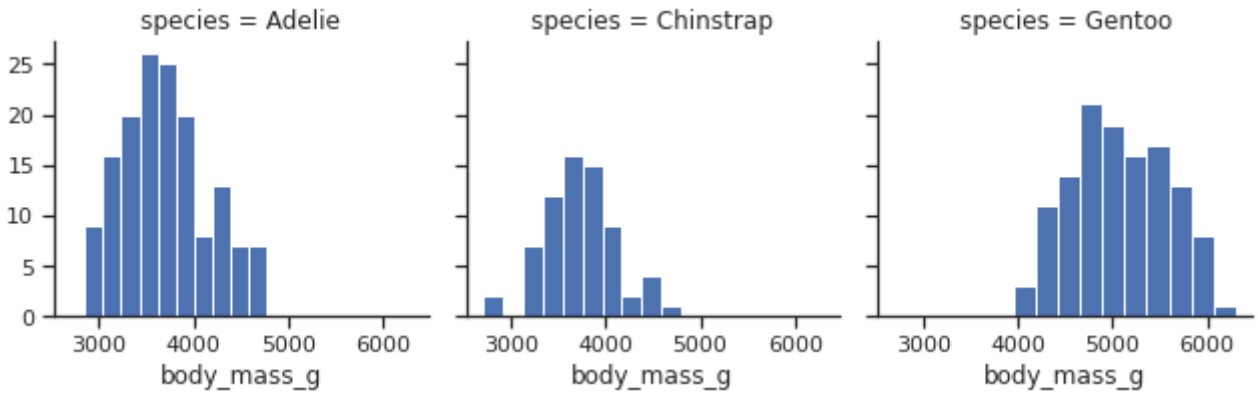
```
g = sns.FacetGrid(penguins, col="sex")
```



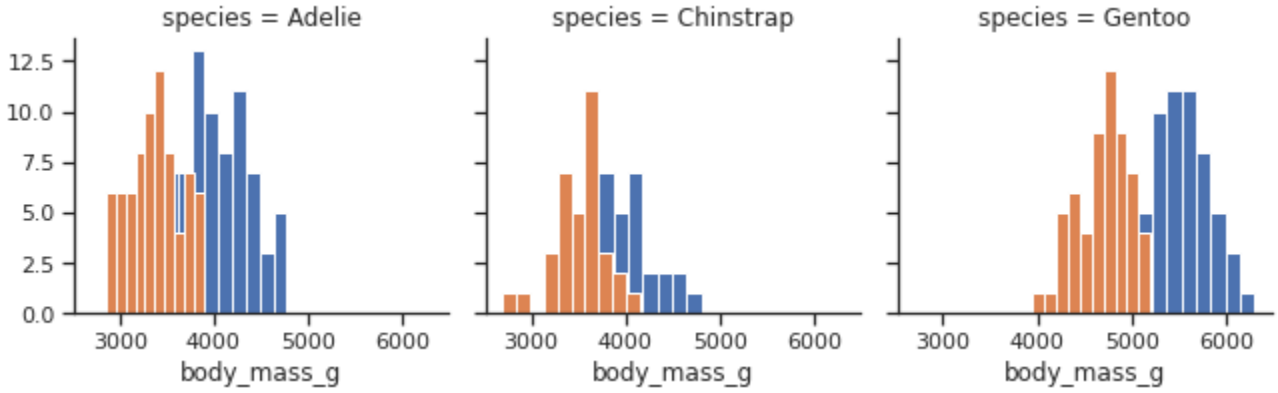

```
g = sns.FacetGrid(penguins, col="sex")
g.map(plt.hist, "body_mass_g");
```



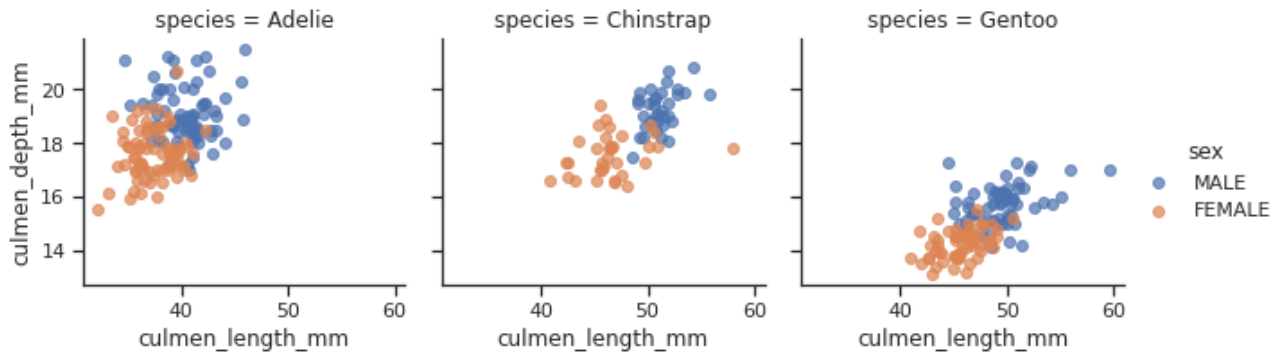

```
g = sns.FacetGrid(penguins, col="species")
g.map(plt.hist, "body_mass_g");
```



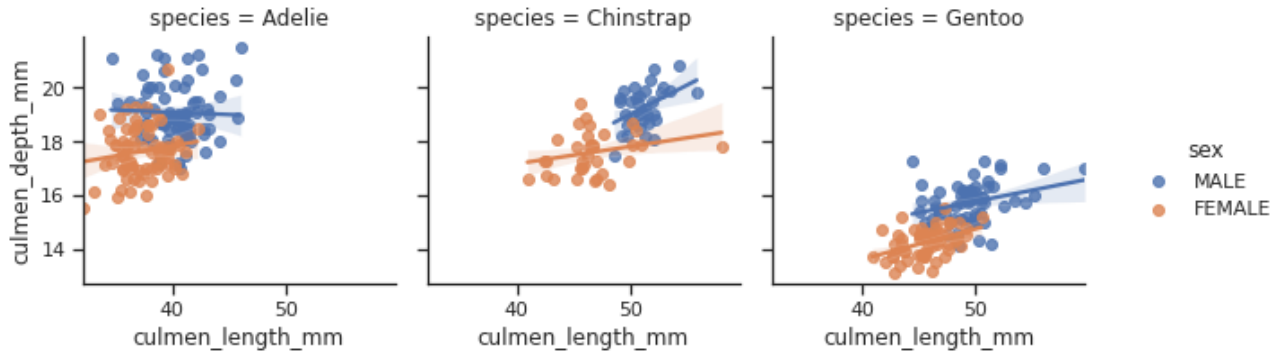

```
g = sns.FacetGrid(penguins, col="species", hue="sex")
g.map(plt.hist, "body_mass_g");
```



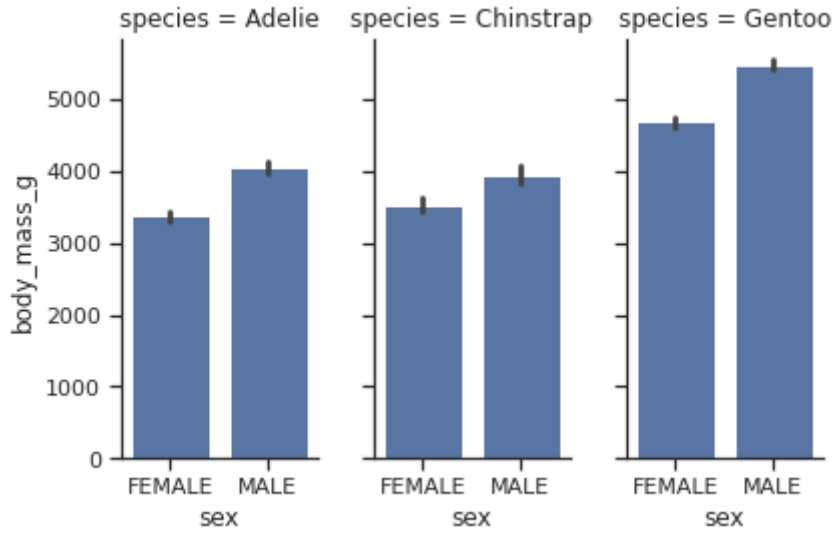

```
g = sns.FacetGrid(penguins, col="species", hue="sex")
g.map(plt.scatter, "culmen_length_mm", "culmen_depth_mm", alpha=.7);
g.add_legend();
```




```
g = sns.FacetGrid(penguins, col="species", hue="sex", margin_titles=True)
g.map(sns.regplot, "culmen_length_mm", "culmen_depth_mm");
g.add_legend();
```



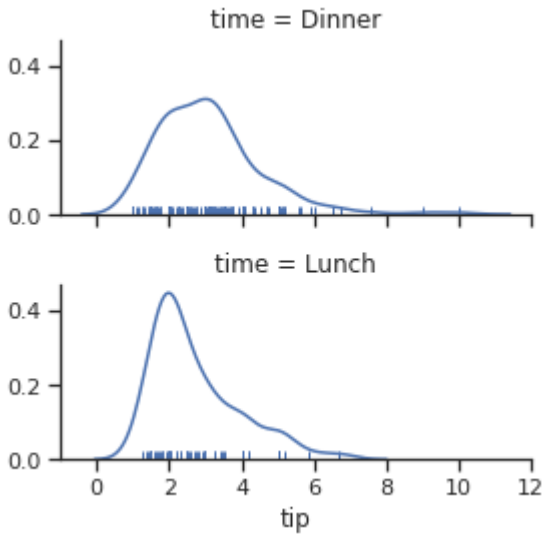

```
g = sns.FacetGrid(penguins, col="species", height=4, aspect=.5)
g.map(sns.barplot, "sex", "body_mass_g", order=["FEMALE", "MALE"]);
```




```
tips = sns.load_dataset("tips")
tips
```

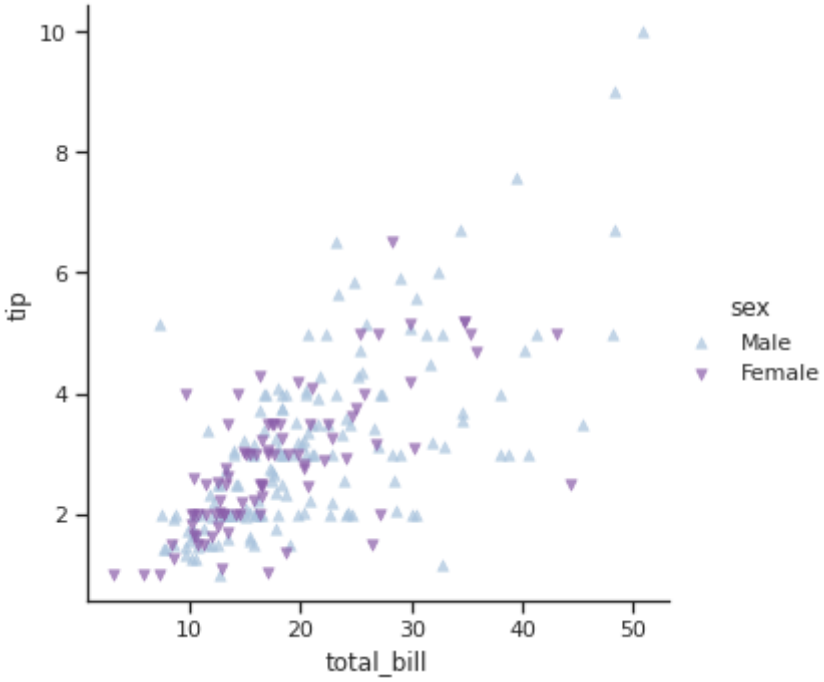
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2

```
ordered_times = tips.time.value_counts().index
g = sns.FacetGrid(tips, row="time", row_order=ordered_times,
                  height=2, aspect=2,)
g.map(sns.distplot, "tip", hist=False, rug=True);
```

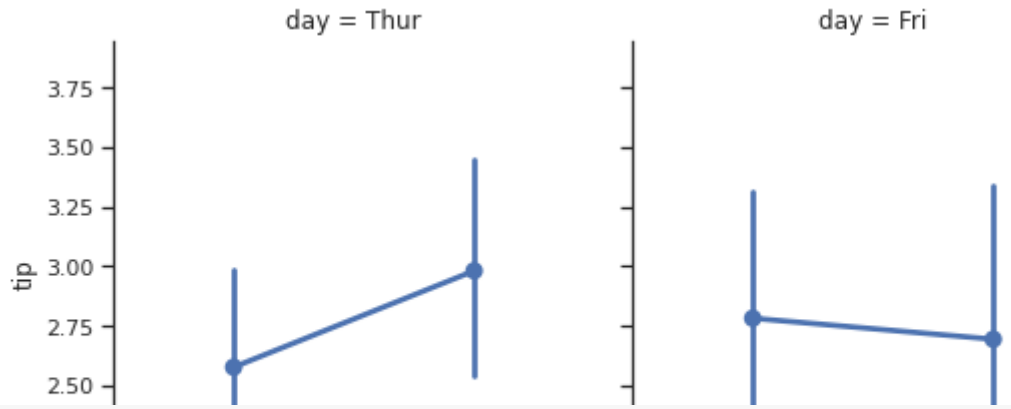



```
g = sns.FacetGrid(tips, hue="day", height=5)
g.map(plt.scatter, "total_bill", "tip", s=30, alpha=.7, linewidth=.5)
g.add_legend();
```

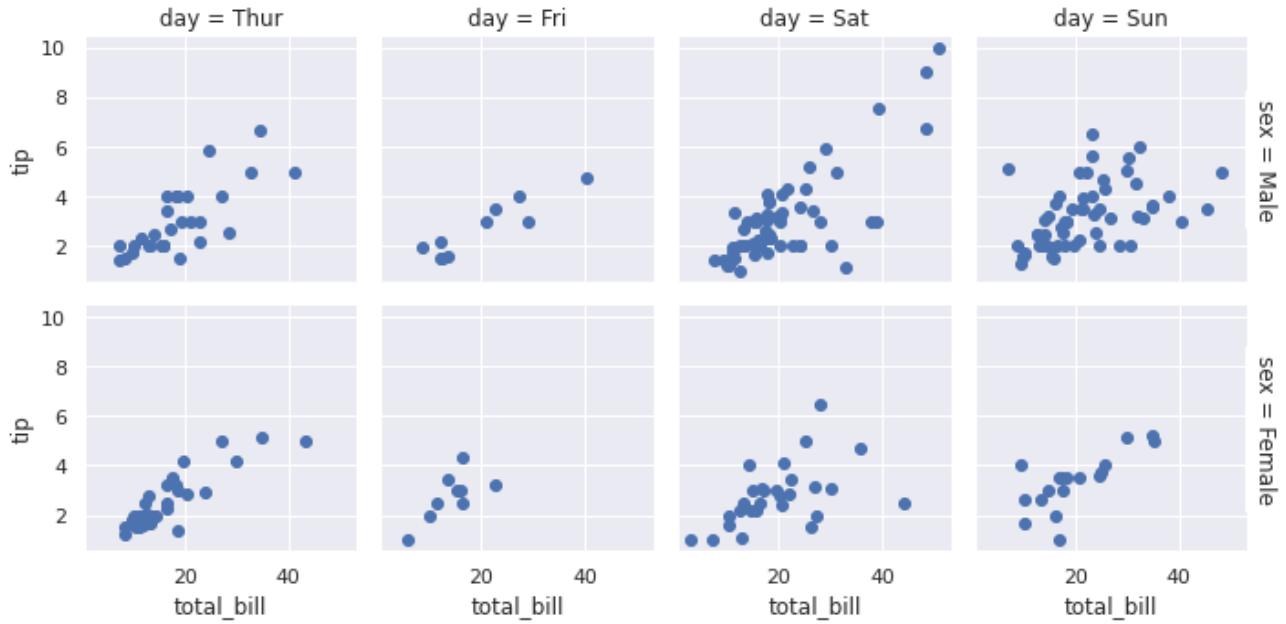

```
g = sns.FacetGrid(tips, hue="sex", palette="BuPu",
                  height=5, hue_kws={"marker": ["^", "v"]})
g.map(plt.scatter, "total_bill", "tip", s=30, alpha=.7, linewidth=.5,)
g.add_legend();
```




```
g = sns.FacetGrid(tips, col="day", col_wrap=2, height=4)
g.map(sns.pointplot, "sex", "tip", order=["Female", "Male"]);
```



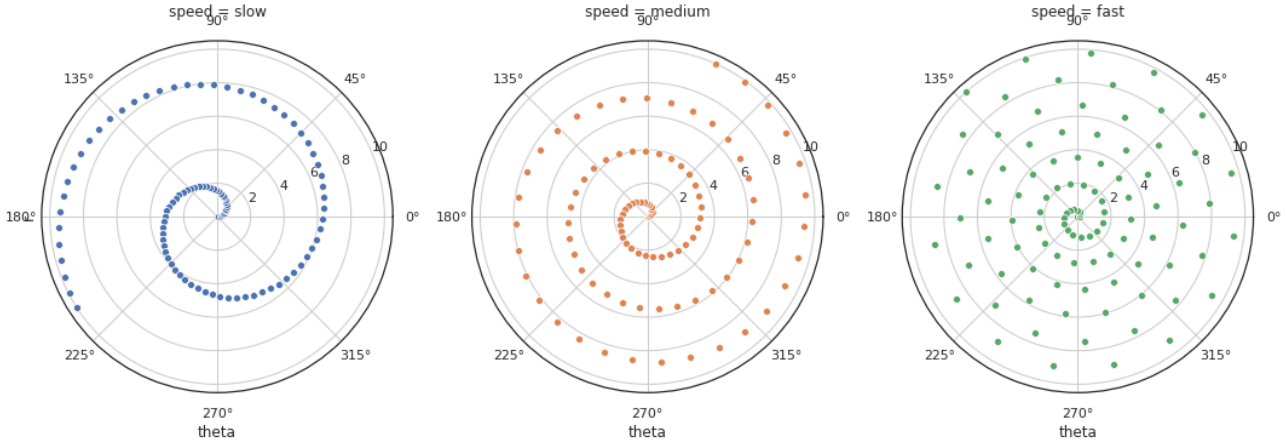

```
with sns.axes_style("darkgrid"):
  g = sns.FacetGrid(tips, row="sex", col="day", margin_titles=True, height=2.5)
g.map(plt.scatter, "total_bill", "tip");
```




```
g = sns.FacetGrid(tips, col="time", margin_titles=True, height=4)
g.map(plt.scatter, "total_bill", "tip")
for ax in g.axes.flat:
  ax.plot((0, 50), (0, .2 * 50), c=".2", ls=":")
```

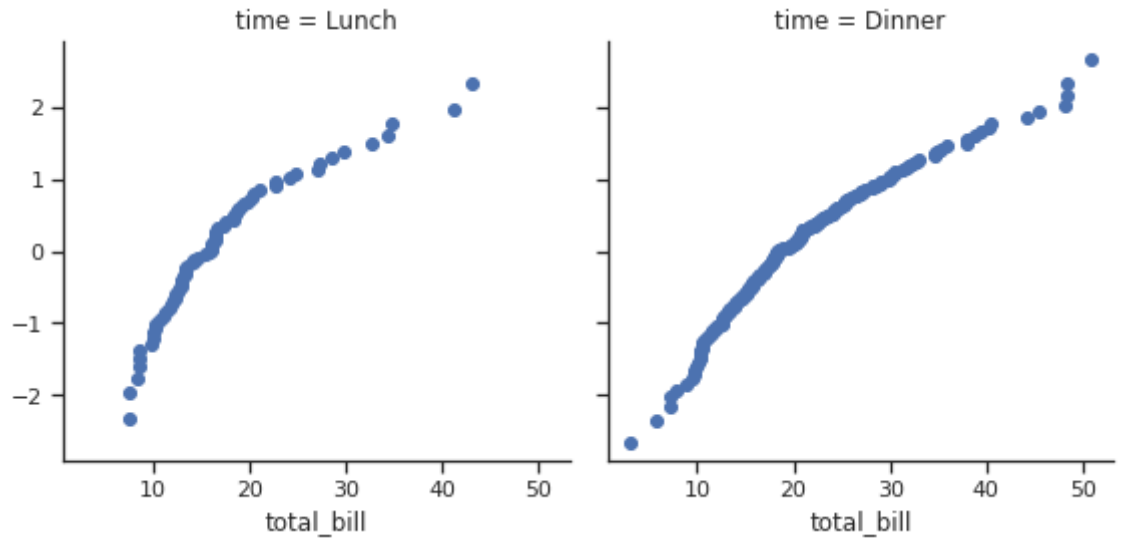
time = Lunch time = Dinner

```
r = np.linspace(0, 10, num=100)
df = pd.DataFrame({'r': r, 'slow': r, 'medium': 2 * r, 'fast': 4 * r})
df = pd.melt(df, id_vars=['r'], var_name='speed', value_name='theta')

g = sns.FacetGrid(df, col="speed", hue="speed",
                  subplot_kws=dict(projection='polar'), height=5,
                  sharex=False, sharey=False, despine=False)
g.map(sns.scatterplot, "theta", "r");
```

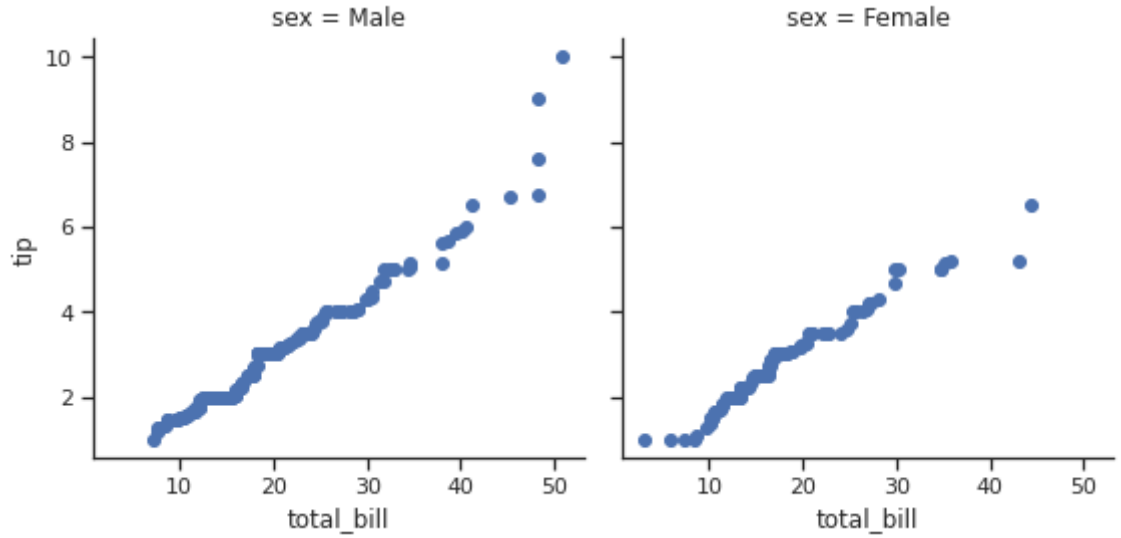


커스텀 함수(Custom functions)

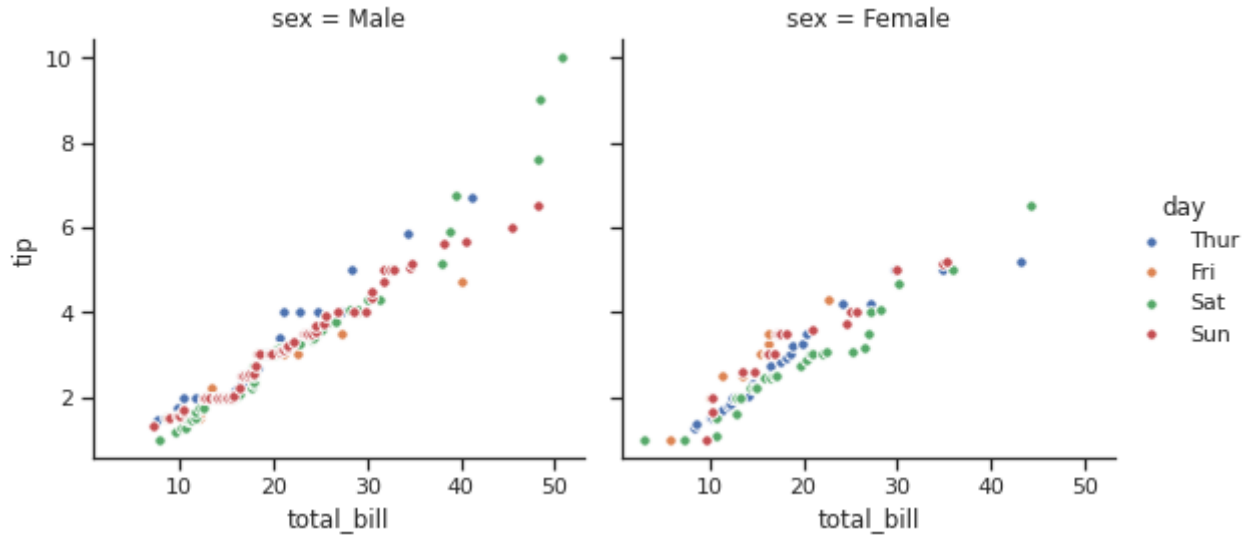
```
def quantile_plot(x, **kwargs):
  qntls, xr = stats.probplot(x, fit=False)
  plt.scatter(xr, qntls, **kwargs)

g = sns.FacetGrid(tips, col="time", height=4)
g.map(quantile_plot, "total_bill");
```



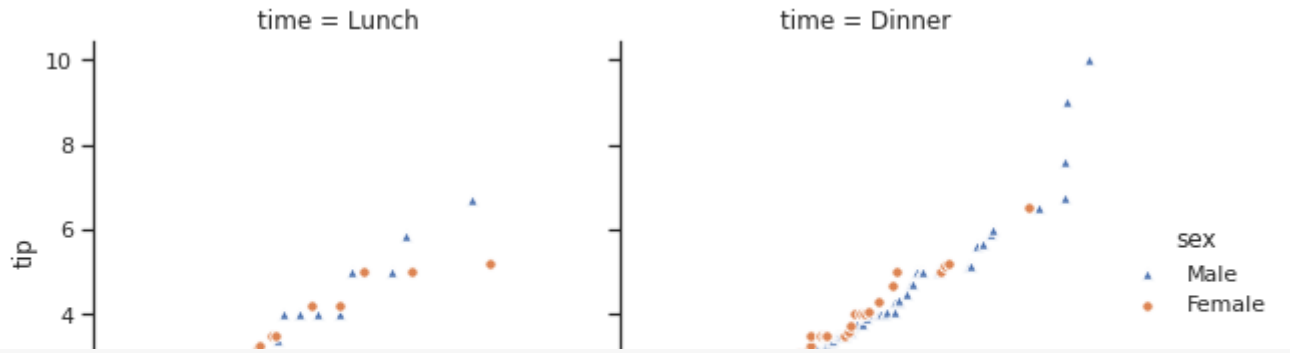
```
def qqplot(x, y, **kwargs):
  _, xr = stats.probplot(x, fit=False)
  _, yr = stats.probplot(y, fit=False)
  plt.scatter(xr, yr, **kwargs)

g = sns.FacetGrid(tips, col="sex", height=4)
g.map(qqplot, "total_bill", "tip");
```



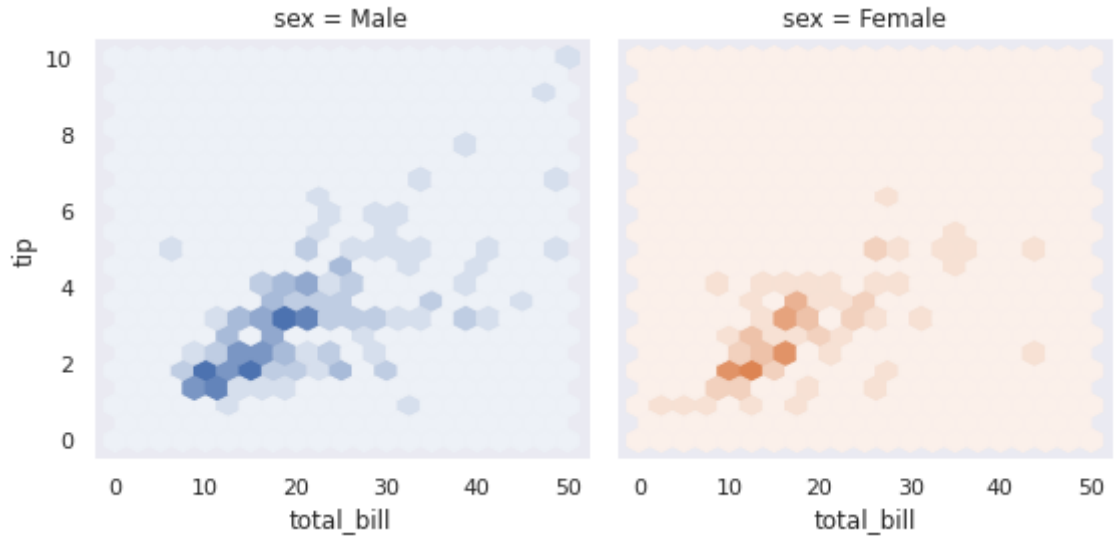

```
g = sns.FacetGrid(tips, col="sex", hue="day", height=4)
g.map(qqplot, "total_bill", "tip", s=30, edgecolor="w")
g.add_legend();
```




```
g = sns.FacetGrid(tips, col="time", hue="sex", height=4,
                  hue_kws={"marker": ["^", "o"]})
g.map(qqplot, "total_bill", "tip", s=30, edgecolor="w")
g.add_legend();
```

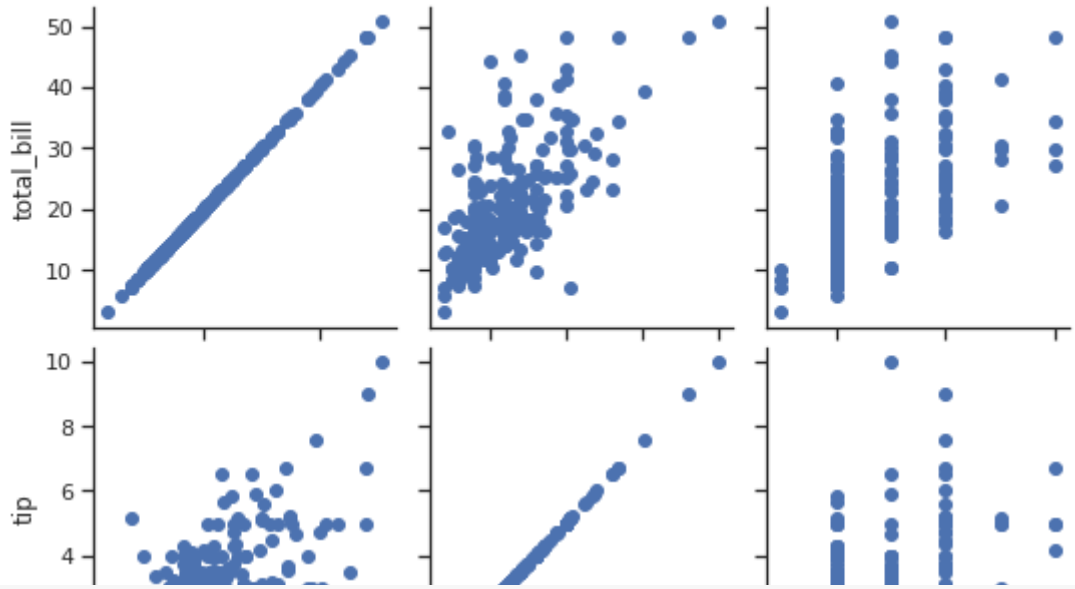



```
def hexbin(x, y, color, **kwargs):
  cmap = sns.light_palette(color, as_cmap=True)
  plt.hexbin(x, y, gridsize=20, cmap=cmap, **kwargs)

with sns.axes_style("dark"):
  g = sns.FacetGrid(tips, hue="sex", col="sex", height=4)
g.map(hexbin, "total_bill", "tip", extent=[0, 50, 0, 10]);
```

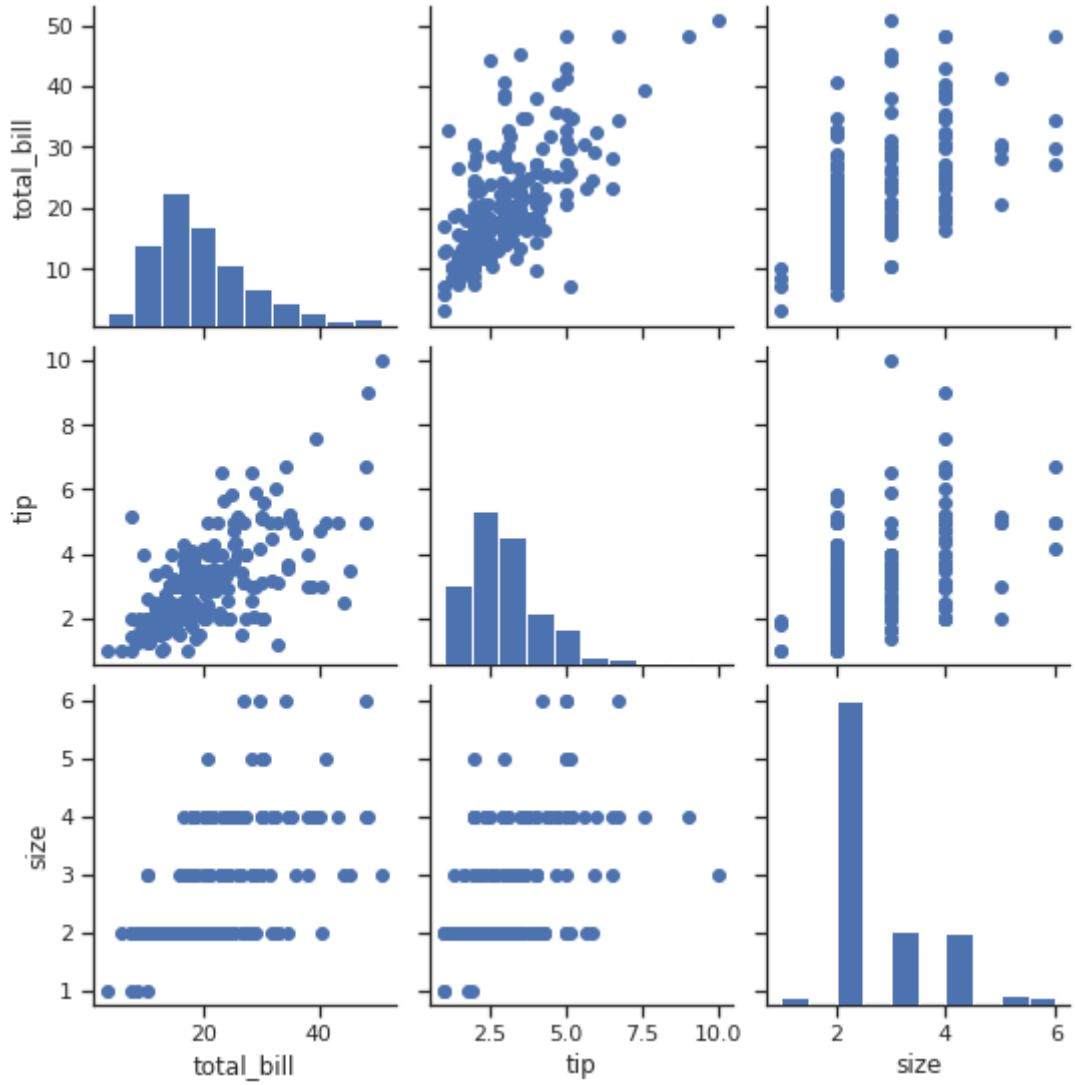


페어와이즈 데이터 관계(pairwise data relationships)

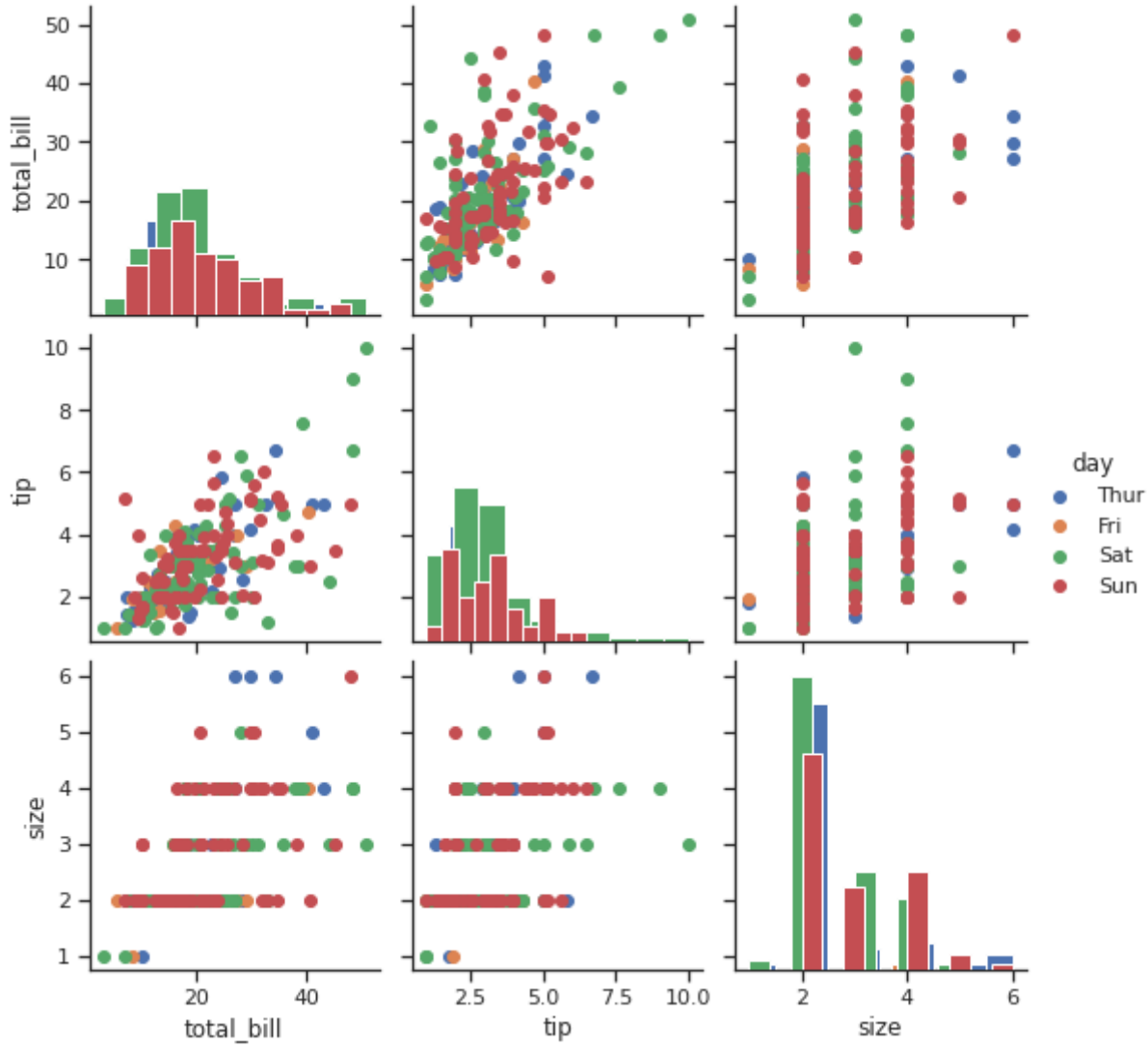
```
g = sns.PairGrid(tips)
g.map(plt.scatter);
```



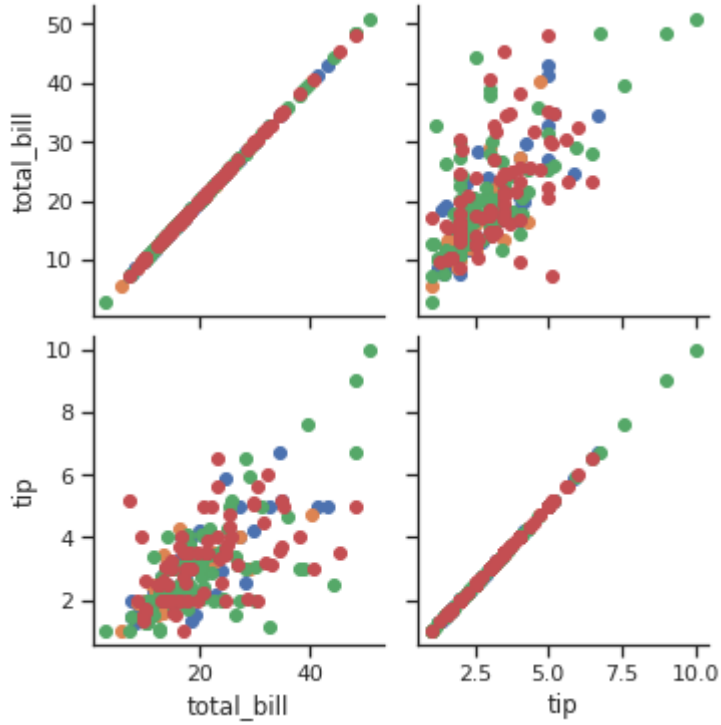

```
g = sns.PairGrid(tips)
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter);
```



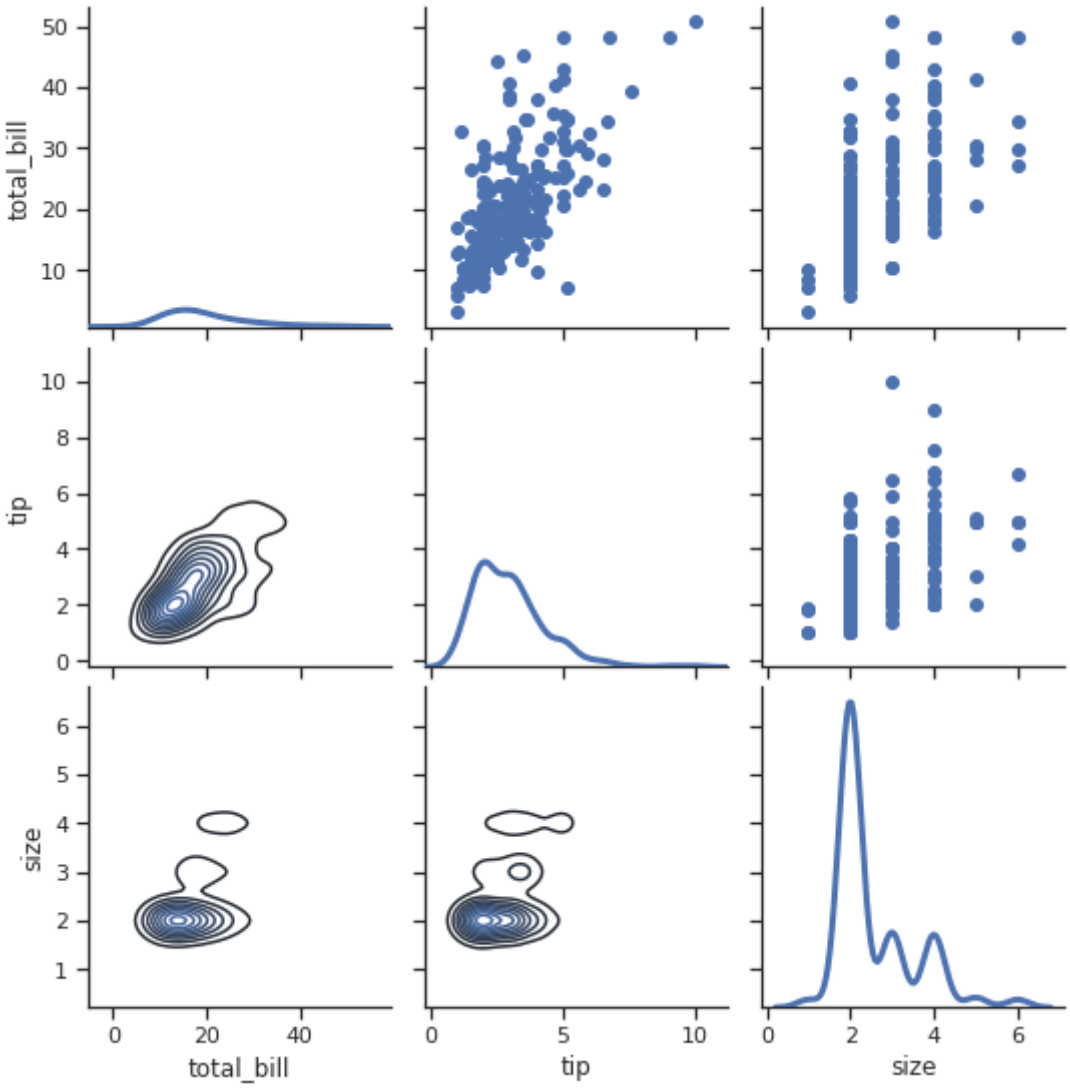

```
g = sns.PairGrid(tips, hue="day")
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter)
g.add_legend();
```




```
g = sns.PairGrid(tips, vars=["total_bill", "tip"], hue="day")
g.map(plt.scatter);
```

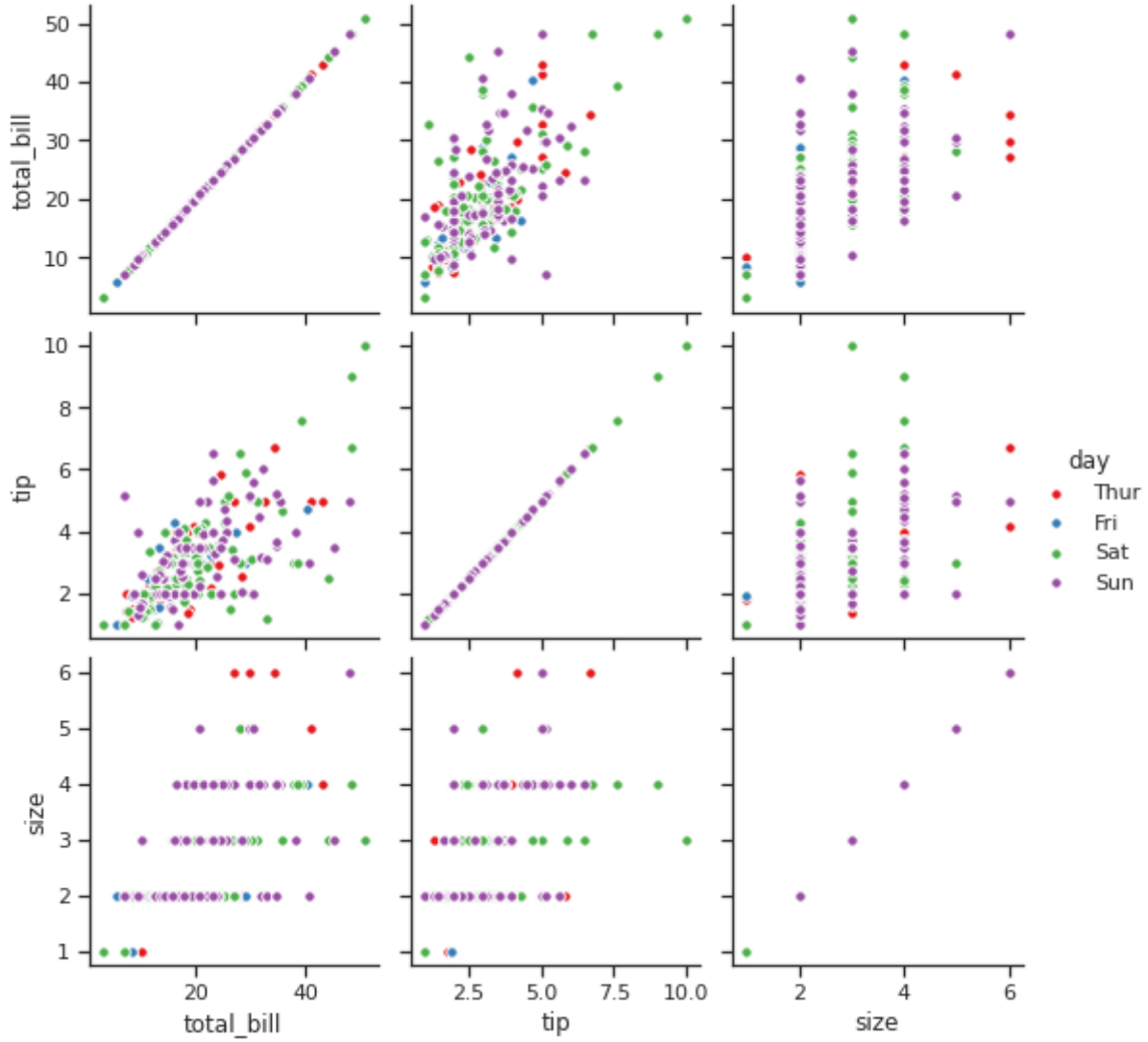


```
g = sns.PairGrid(tips)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
g.map_diag(sns.kdeplot, lw=3, legend=False);
```

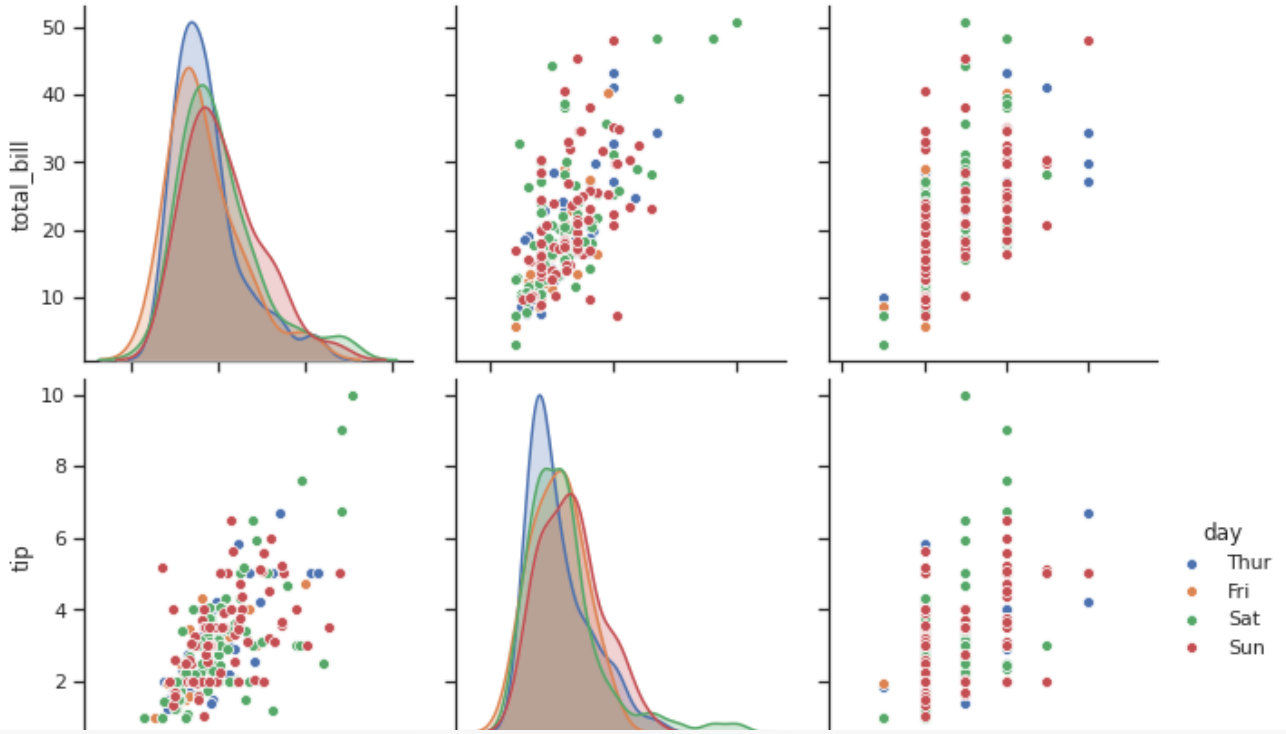



```
g = sns.PairGrid(tips, y_vars=["tip"],
                 x_vars=["size", "total_bill"], height=4)
g.map(sns.regplot);
```


```
g = sns.PairGrid(tips, hue="day", palette="Set1")
g.map(plt.scatter, s=30, edgecolor="white")
g.add_legend();
```




```
sns.pairplot(tips, hue="day", height=3);
```

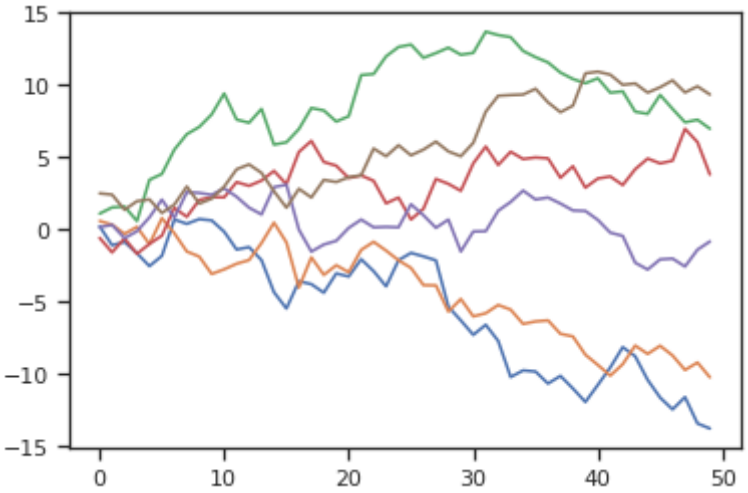



```
sns.pairplot(tips, hue="day", palette="Set1", diag_kind="hist", height=3);
```

그림 미학 제어

```
def randplot(flip=1):
  for i in range(1, 7):
    plt.plot(np.random.randn(50).cumsum());
```

```
randplot();
```

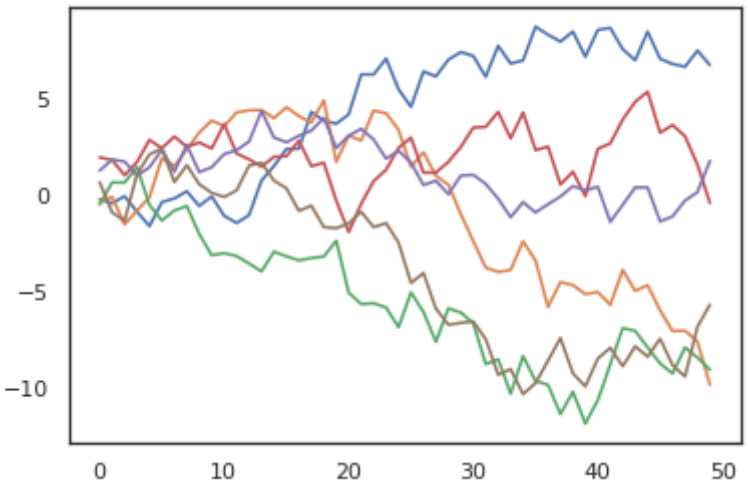



```
sns.set()
randplot();
```

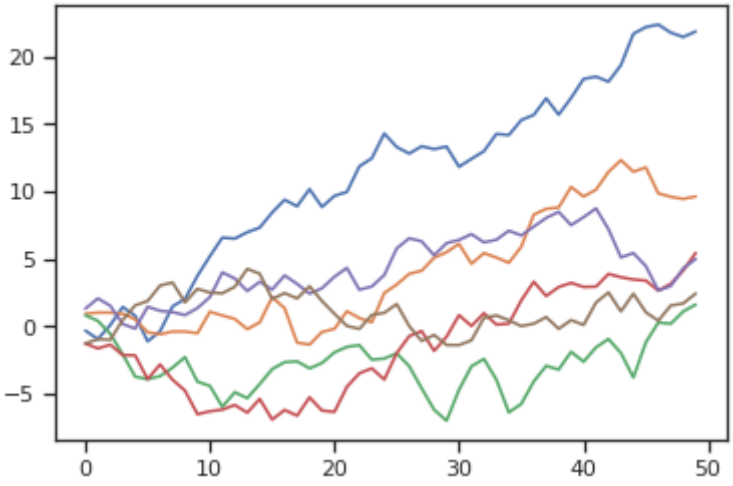
Seaborn 스타일

```
sns.set_style("dark")
randplot();
```

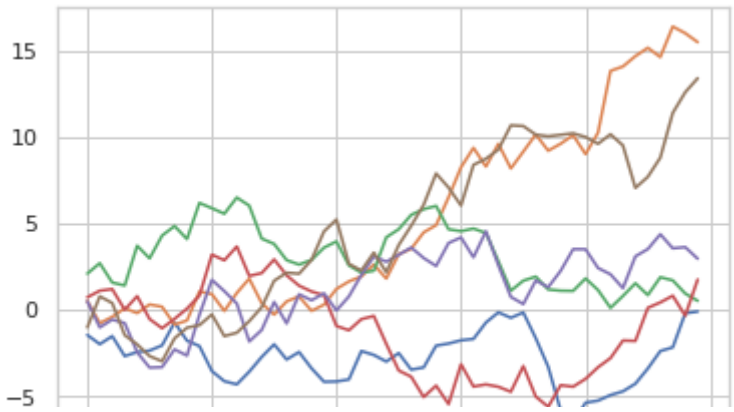

```
sns.set_style("white")
randplot();
```




```
sns.set_style("ticks")
randplot();
```




```
sns.set_style("whitegrid")
randplot();
```

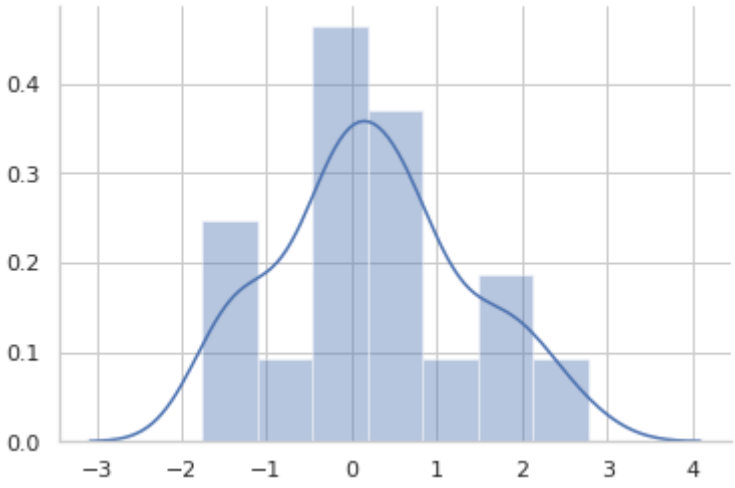


축 스핀 제거

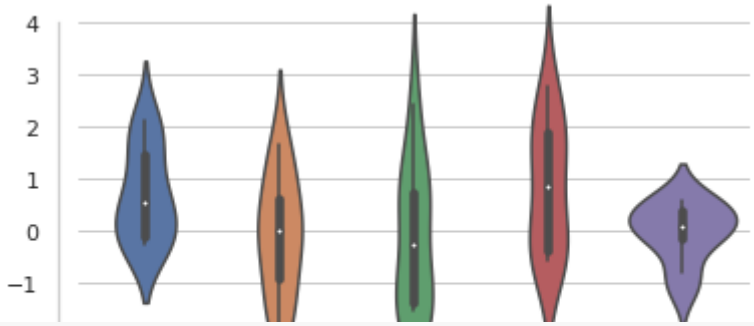
```
d = np.random.randn(50).reshape(10, 5)
d
```

```
array([[-0.14345284,  0.14281093,  0.51012472,  1.48795513, -0.81150338],
       [ 1.80688002, -1.11073811, -1.40715657,  0.44564092, -0.14544976],
       [ 0.81881835, -0.12550074, -1.32940081,  2.77042053,  0.29349789],
       [ 1.66855198, -1.45894745,  2.42163417, -0.44654466,  0.02844722],
       [ 2.13349114,  1.67342372, -1.53281237,  2.02813167,  0.42107755],
       [ 0.00616135,  0.11844605,  0.96362046,  2.0121311 ,  0.13101397],
       [ 0.66028377, -0.35024241, -1.47337614, -0.58273093, -1.11202186],
       [ 0.42323624,  0.80297211,  0.64742277, -0.43426212,  0.58504976],
       [-0.20493977, -1.75356947,  0.76583953, -0.20189372,  0.54851309],
       [-0.24068777,  1.20682629, -1.00517743,  1.24909899, -0.08734775]])
```

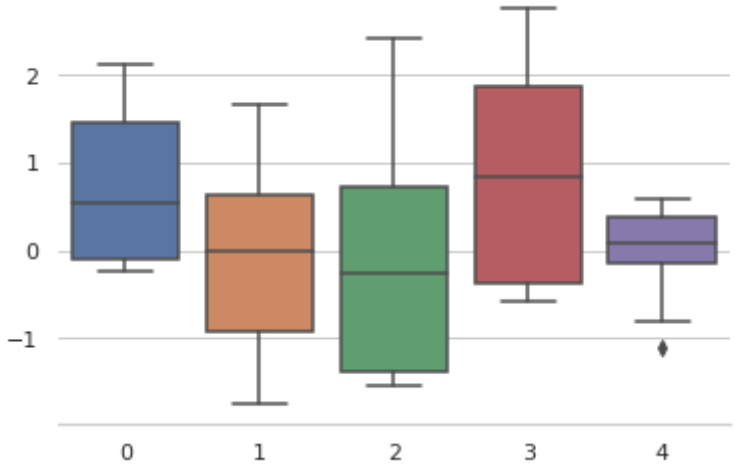
```
sns.distplot(d)
sns.despine()
```




```
sns.violinplot(data=d)
sns.despine(offset=10, trim=True);
```

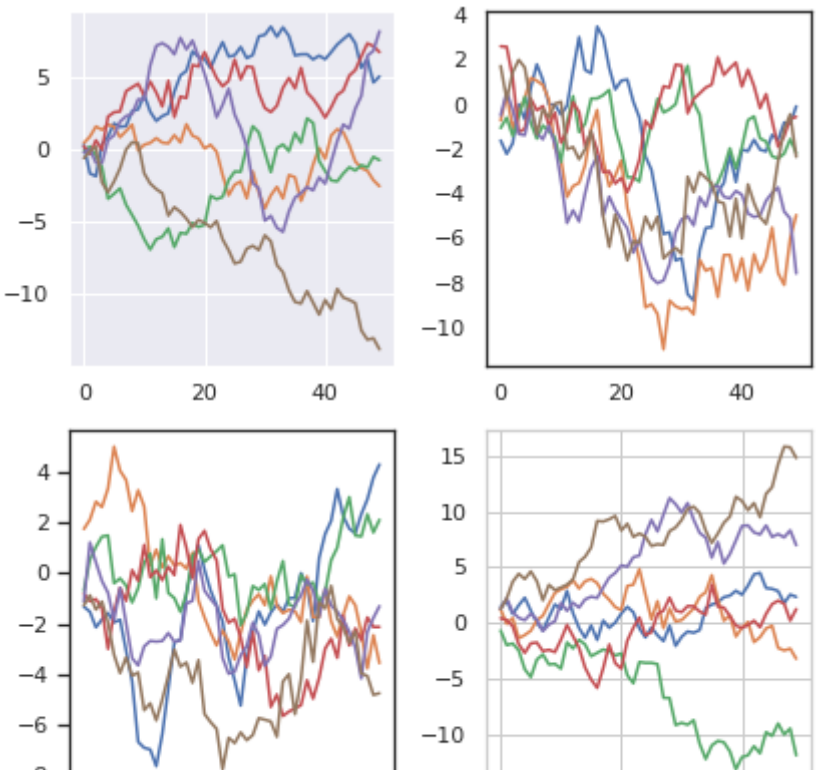



```
sns.boxplot(data=d, palette="deep")
sns.despine(left=True)
```



스타일 임시 설정

```
f = plt.figure(figsize=(6, 6))
gs = f.add_gridspec(2, 2)

with sns.axes_style("darkgrid"):
  ax = f.add_subplot(gs[0, 0])
  randplot()

with sns.axes_style("white"):
  ax = f.add_subplot(gs[0, 1])
  randplot()

with sns.axes_style("ticks"):
  ax = f.add_subplot(gs[1, 0])
  randplot()

with sns.axes_style("whitegrid"):
  ax = f.add_subplot(gs[1, 1])
  randplot()

f.tight_layout()
```



스타일 요소 재정의

```
sns.axes_style()
```

```
{'axes.axisbelow': True,
 'axes.edgecolor': '.8',
 'axes.facecolor': 'white',
 'axes.grid': True,
 'axes.labelcolor': '.15',
 'axes.spines.bottom': True,
 'axes.spines.left': True,
 'axes.spines.right': True,
 'axes.spines.top': True,
 'figure.facecolor': 'white',
 'font.family': ['sans-serif'],
 'font.sans-serif': ['Arial',
  'DejaVu Sans',
  'Liberation Sans',
  'Bitstream Vera Sans',
  'sans-serif'],
 'grid.color': '.8',
 'grid.linestyle': '-',
 'image.cmap': 'rocket',
 'lines.solid_capstyle': 'round',
 'patch.edgecolor': 'w',
 'patch.force_edgecolor': True,
 'text.color': '.15',
 'xtick.bottom': False,
 'xtick.color': '.15',
 'xtick.direction': 'out',
 'xtick.top': False,
 'ytick.color': '.15',
 'ytick.direction': 'out',
 'ytick.left': False,
 'ytick.right': False}
```

```
{'axes.axisbelow': True,
 'axes.edgecolor': '.8',
 'axes.facecolor': 'white',
 'axes.grid': True,
 'axes.labelcolor': '.15',
 'axes.spines.bottom': True,
 'axes.spines.left': True,
 'axes.spines.right': True,
 'axes.spines.top': True,
 'figure.facecolor': 'white',
 'font.family': ['sans-serif'],
 'font.sans-serif': ['Arial',
  'DejaVu Sans',
  'Liberation Sans',
  'Bitstream Vera Sans',
  'sans-serif'],
 'grid.color': '.8',
 'grid.linestyle': '-',
 'image.cmap': 'rocket',
 'lines.solid_capstyle': 'round',
 'patch.edgecolor': 'w',
 'patch.force_edgecolor': True,
 'text.color': '.15',
 'xtick.bottom': False,
 'xtick.color': '.15',
 'xtick.direction': 'out',
 'xtick.top': False,
 'ytick.color': '.15',
 'ytick.direction': 'out',
 'ytick.left': False,
 'ytick.right': False}
```

```
    {'axes.axisbelow': True,
     'axes.edgecolor': '.8',
     'axes.facecolor': 'white',
     'axes.grid': True,
     'axes.labelcolor': '.15',
     'axes.spines.bottom': True,
     'axes.spines.left': True,
     'axes.spines.right': True,
     'axes.spines.top': True,
     'figure.facecolor': 'white',
     'font.family': ['sans-serif'],
     'font.sans-serif': ['Arial',
      'DejaVu Sans',
      'Liberation Sans',
      'Bitstream Vera Sans',
      'sans-serif'],
     'grid.color': '.8',
     'grid.linestyle': '-',
     'image.cmap': 'rocket',
     'lines.solid_capstyle': 'round',
     'patch.edgecolor': 'w',
     'patch.force_edgecolor': True,
     'text.color': '.15',
     'xtick.bottom': False,
```

```
 'xtick.color': '.15',
 'xtick.direction': 'out',
 'xtick.top': False,
 'ytick.color': '.15',
 'ytick.direction': 'out',
 'ytick.left': False,
 'ytick.right': False}
```

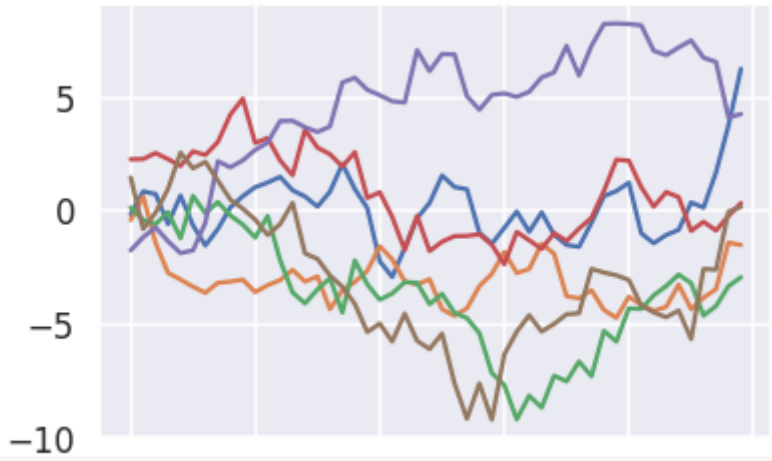
```
sns.set_style("darkgrid", {"axes.facecolor":".5", "grid.linestyle": ":"})
randplot();
```

스케일링 플롯 요소

```
sns.set()
```

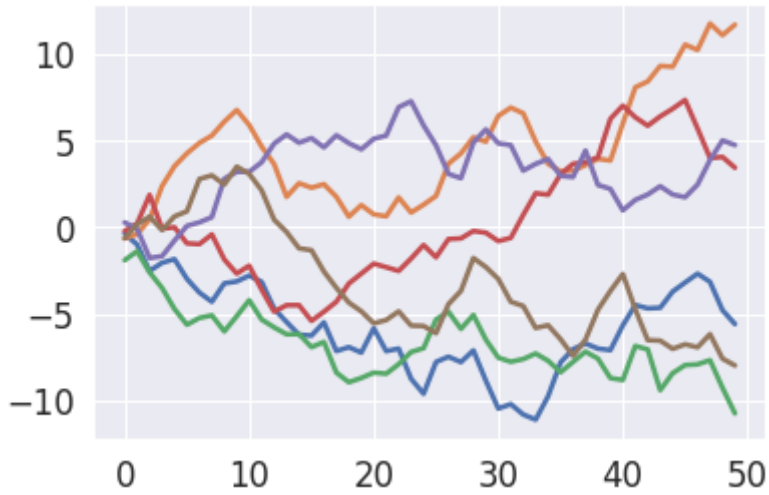
```
sns.set_context("paper")
randplot();
```


```
sns.set_context("talk")
randplot();
```




```
sns.set_context("poster")
randplot();
```


```
sns.set_context("notebook", font_scale=1.5, rc={"lines.linewidth":2.5})
randplot();
```



컬러 팔레트 선택

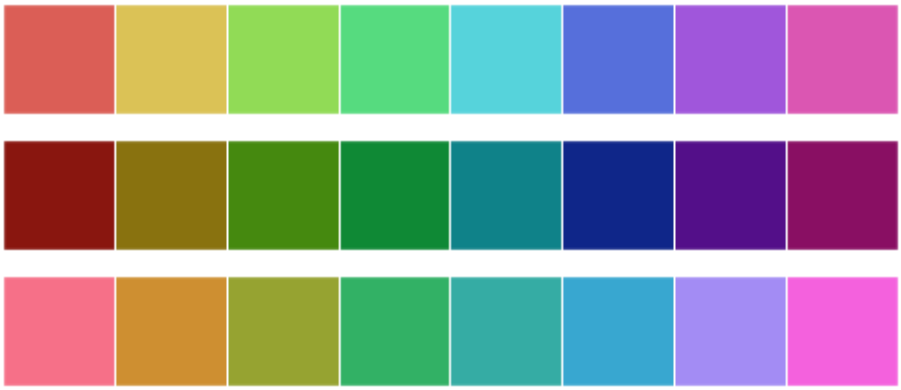
```
sns.set()
```

질적 색상 팔레트

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

원형 컬러 시스템 사용

```
sns.palplot(sns.color_palette("hls", 8))
sns.palplot(sns.hls_palette(8, l=.3, s=.8))
sns.palplot(sns.color_palette("husl", 8))
```



범주형 컬러 브루어 팔레트 사용

```
sns.palplot(sns.color_palette("Paired"))
sns.palplot(sns.color_palette("Set1"))
sns.palplot(sns.color_palette("Set2"))
sns.palplot(sns.color_palette("Set3"))
flatui = ["#99FF00", "#99CC00", "#999900", "#996600", "#993300", "#990000"]
sns.palplot(sns.color_palette(flatui))
```

xkcd 색상 측량에서 정의된 색상 사용

- xkcd 색상표: https://xkcd.com/color/rgb/

```
plt.plot(np.random.randn(50).cumsum(), sns.xkcd_rgb["pale red"], lw=3)
plt.plot(np.random.randn(50).cumsum(), sns.xkcd_rgb["medium green"], lw=3)
plt.plot(np.random.randn(50).cumsum(), sns.xkcd_rgb["denim blue"], lw=3);
```


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

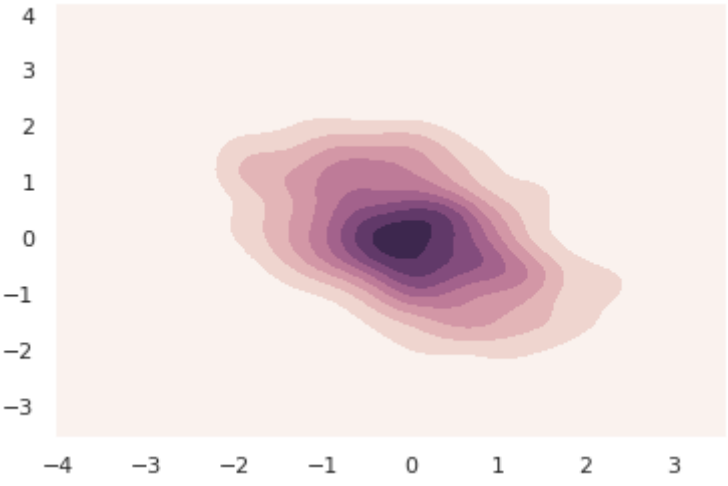
순차 색상 팔레트

```
sns.palplot(sns.color_palette("Blues"))
sns.palplot(sns.color_palette("BuGn_r"))
sns.palplot(sns.color_palette("GnBu_d"))
```

순차적 입방체 팔레트


```
sns.palplot(sns.color_palette("cubehelix", 8))
sns.palplot(sns.cubehelix_palette(8))
sns.palplot(sns.cubehelix_palette(8, start=.5, rot=-.75))
sns.palplot(sns.cubehelix_palette(8, start=2, rot=0, dark=0, light=.95, reverse=True))
```

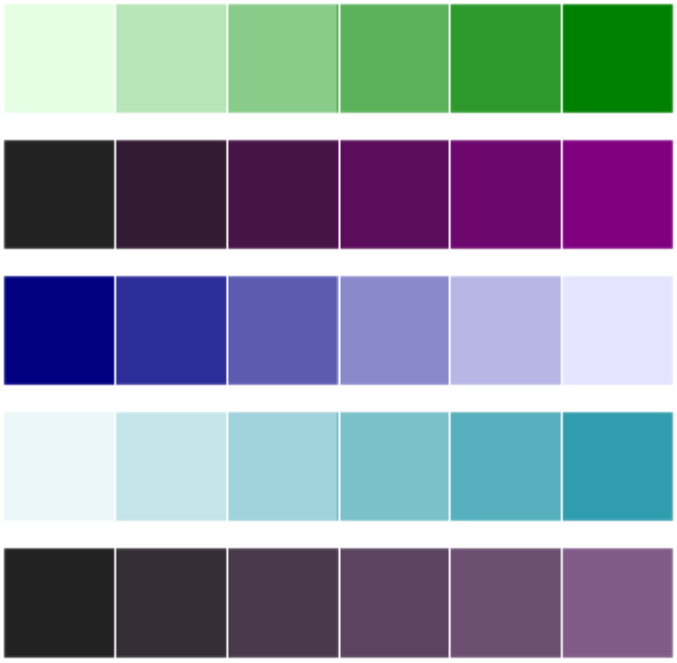
```
x, y = np.random.multivariate_normal([0, 0], [[1, -.5], [-.5, 1]], size=500).T
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
sns.kdeplot(x, y, cmap=cmap, shade=True);
```



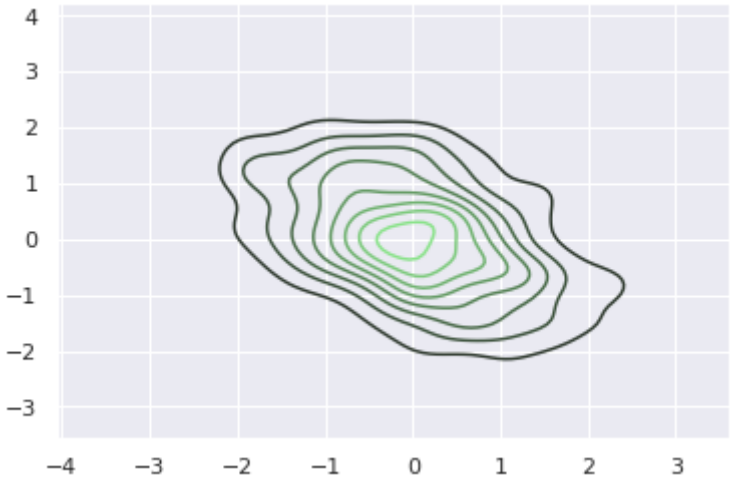
사용자 정의 순차적 팔레트

```
sns.palplot(sns.light_palette("green"))
sns.palplot(sns.dark_palette("purple"))
sns.palplot(sns.light_palette("navy", reverse=True))
sns.palplot(sns.light_palette((210, 90, 60), input="husl"))
```

```
sns.palplot(sns.dark_palette("muted purple", input="xkcd"))
```



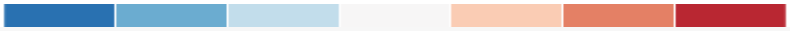
```
pal = sns.dark_palette("palegreen", as_cmap=True)
sns.kdeplot(x, y, cmap=pal);
```



색상 팔레트 나누기

```
sns.palplot(sns.color_palette("BrBG", 7))
sns.palplot(sns.color_palette("RdBu_r", 7))
sns.palplot(sns.color_palette("coolwarm", 7))
```

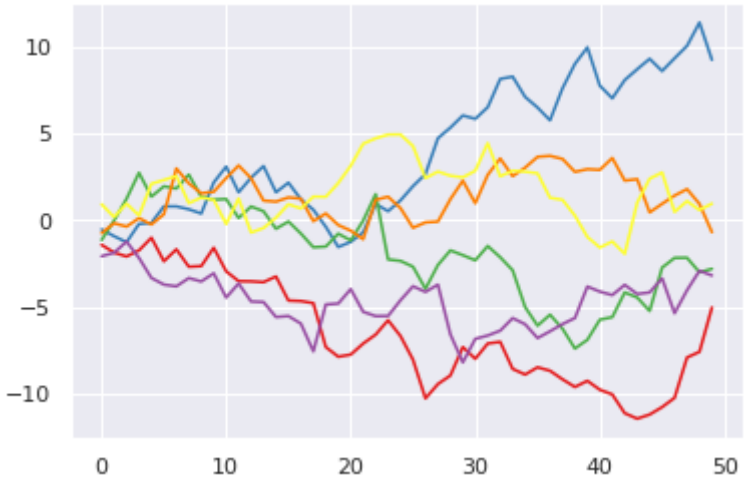
커스텀 분기 팔레트



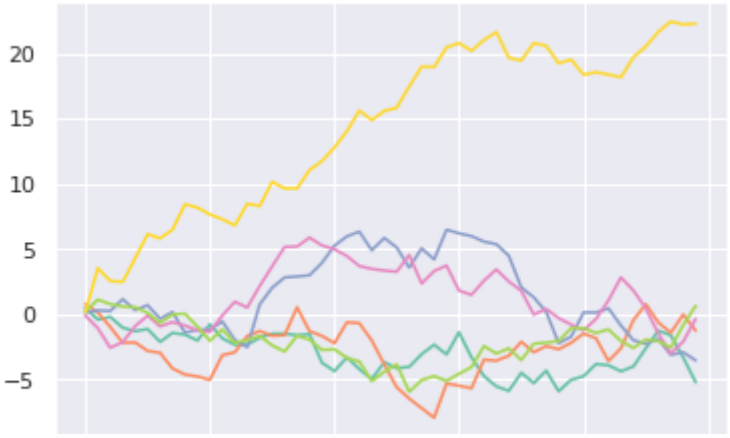
```
sns.palplot(sns.diverging_palette(220, 20, n=7))
sns.palplot(sns.diverging_palette(145, 280, s=85, l=25, n=7))
sns.palplot(sns.diverging_palette(10, 220, sep=80, n=7))
sns.palplot(sns.diverging_palette(255, 133, l=60, n=7, center="dark"))
```

기본 색상 표 설정

```
sns.set_palette("Set1")
randplot();
```




```
sns.set_palette("Set2")
randplot();
```



참고문헌

- Seaborn, https://seaborn.pydata.org/
- Igor Milovanovi, "Python Data Visualization Cookbook", Packt
- Jake VanderPlas, "Python Data Science Handbook", O'Reilly
- Wes Mckinney, "Python for Data Analysis", O'Reilly