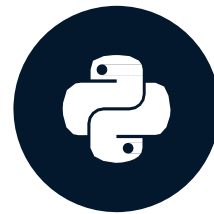


Changing plot style and color

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Changing the figure style

- Figure "style" includes background and axes
- Preset options: "white", "dark", "whitegrid", "darkgrid", "ticks"
- `sns.set_style()`

Seaborn은 5개의 preset figure style을 가짐 :

1. "white"
2. "dark"
3. "whitegrid"
4. "darkgrid"
5. "ticks"

-> 이것들은 plot의 background와 axes를 바꿈

set_style() function을 이용하면 이 중 하나를 모든 plot에 대한 global style로 지정 가능 : `sns.set_style()`

1. Default figure style ("white")

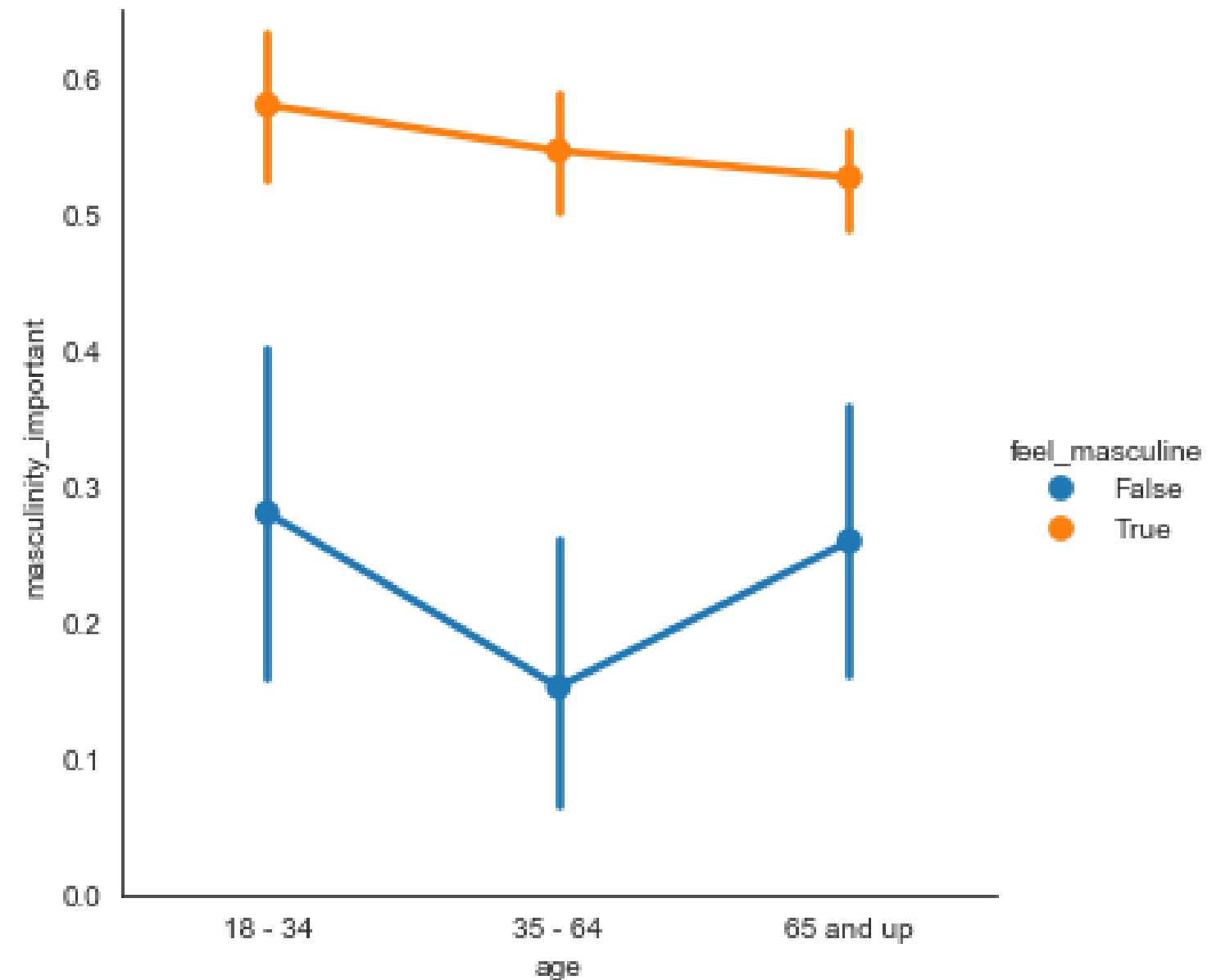
```
sns.catplot(x="age",  
            y="masculinity_important",  
            data=masculinity_data, hue="feel_masculine", kind="point")  
  
plt.show()
```

그래프 : Showing the percentage of men who reports masculinity was important to them

디폴트 style : "white"

=> provide clean axes with white solid background

Specific value가 아니라 그룹 간의 general trend에만 관심이 있거나
두 그룹 간 비교에만 중점을 둘 경우 "white" style은 good choice



2. Figure style: "whitegrid"

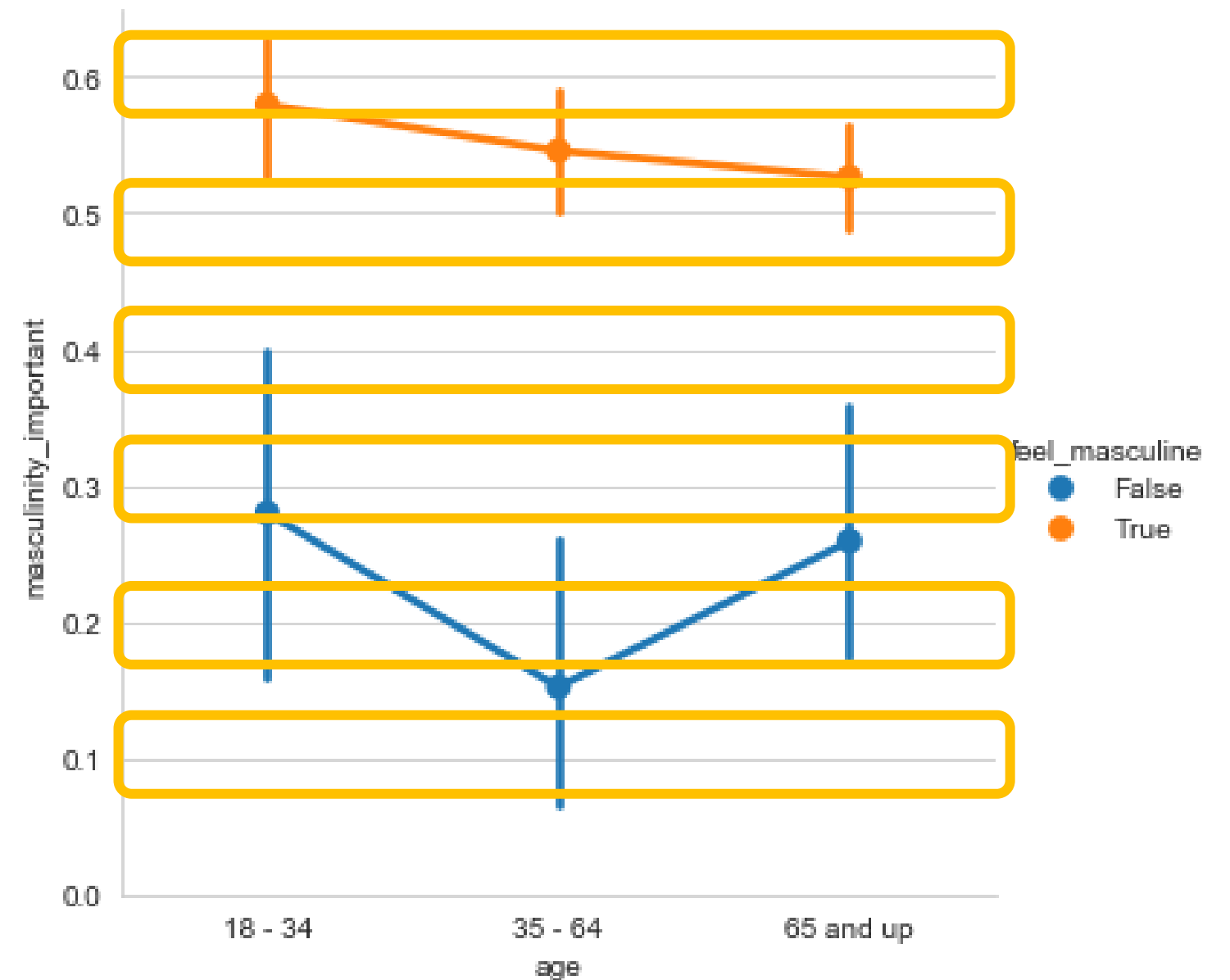
```
sns.set_style("whitegrid")

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data, hue=
            "feel_masculine", kind=
            "point")

plt.show()
```

`sns.set_style("whitegrid")` : style을 "whitegrid"로 변경
⇒ Background에 grey grid 추가

This is useful if you want your audience to be able to determine "specific values" of the plot



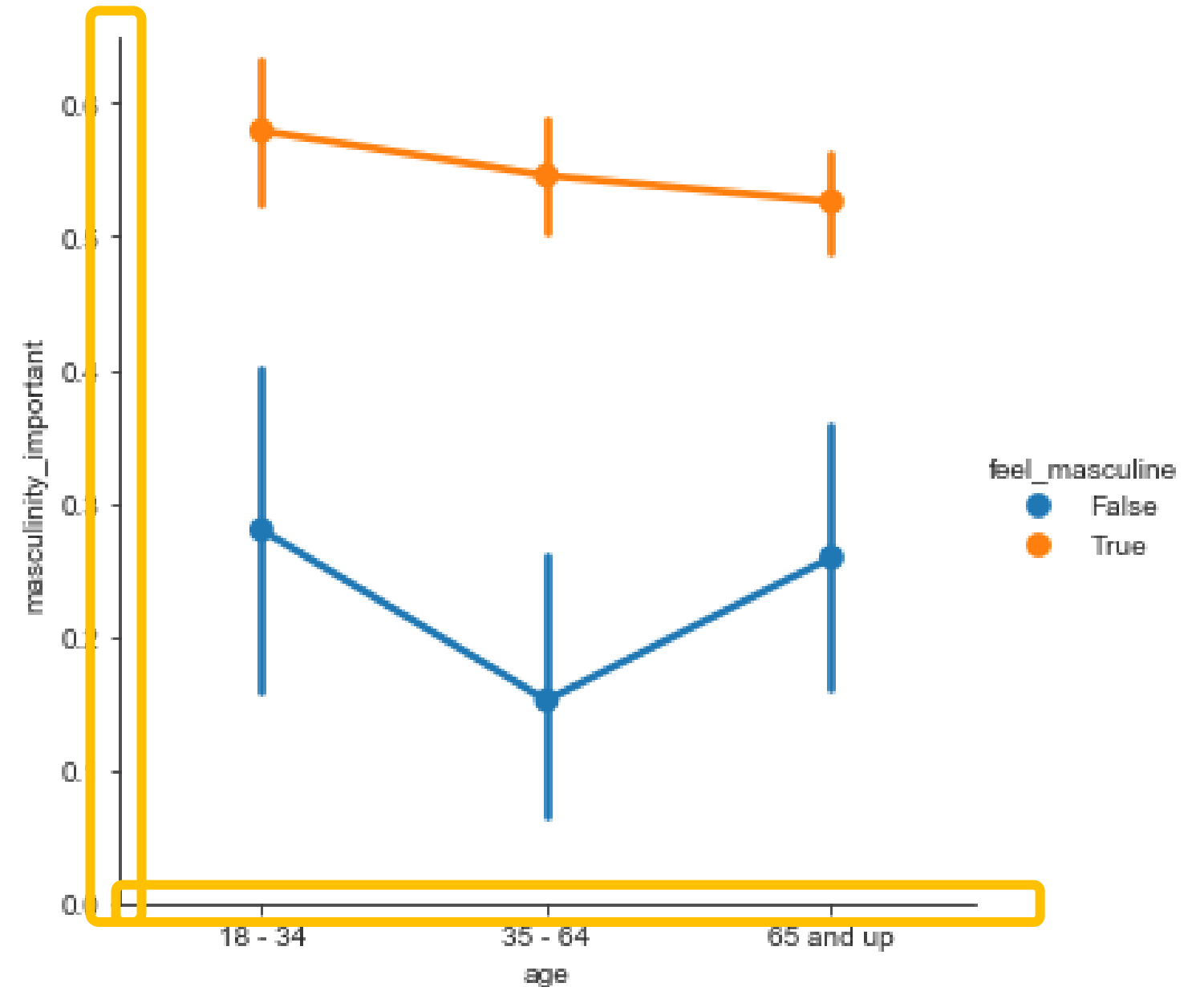
3. Other styles : “ticks”

```
sns.set_style("ticks")

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data, hue=
            "feel_masculine", kind=
            "point")

plt.show()
```

`sns.set_style("ticks")` : style을 “ticks”로 변경
⇒ “white” style과 비슷하나, 차이점은 x와 y축에 작은 tick mark 추가



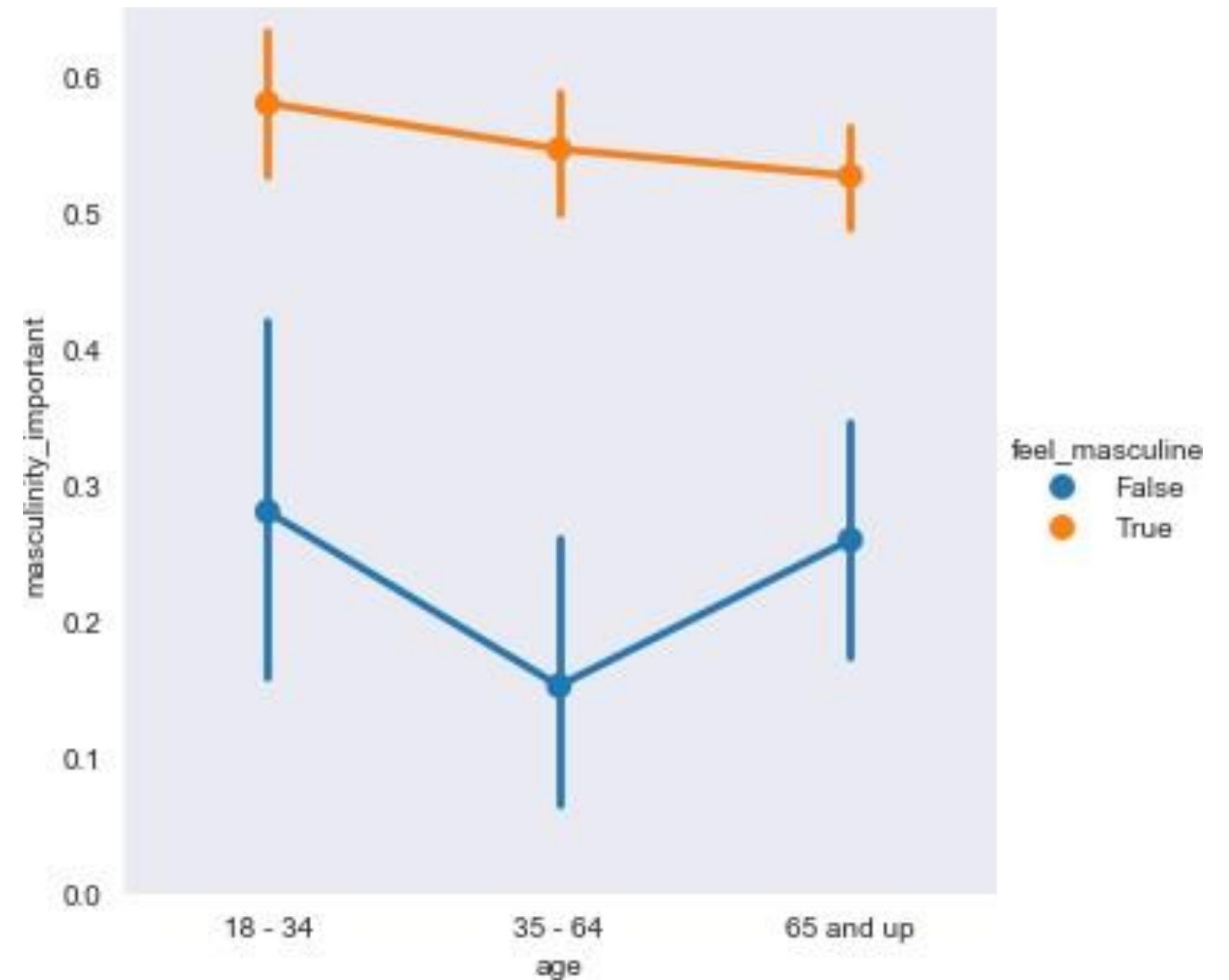
4. Other styles : “dark”

```
sns.set_style("dark")

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data, hue=
            "feel_masculine", kind=
            "point")

plt.show()
```

`sns.set_style("dark")` : style을 “dark”로 변경
⇒ 회색 background



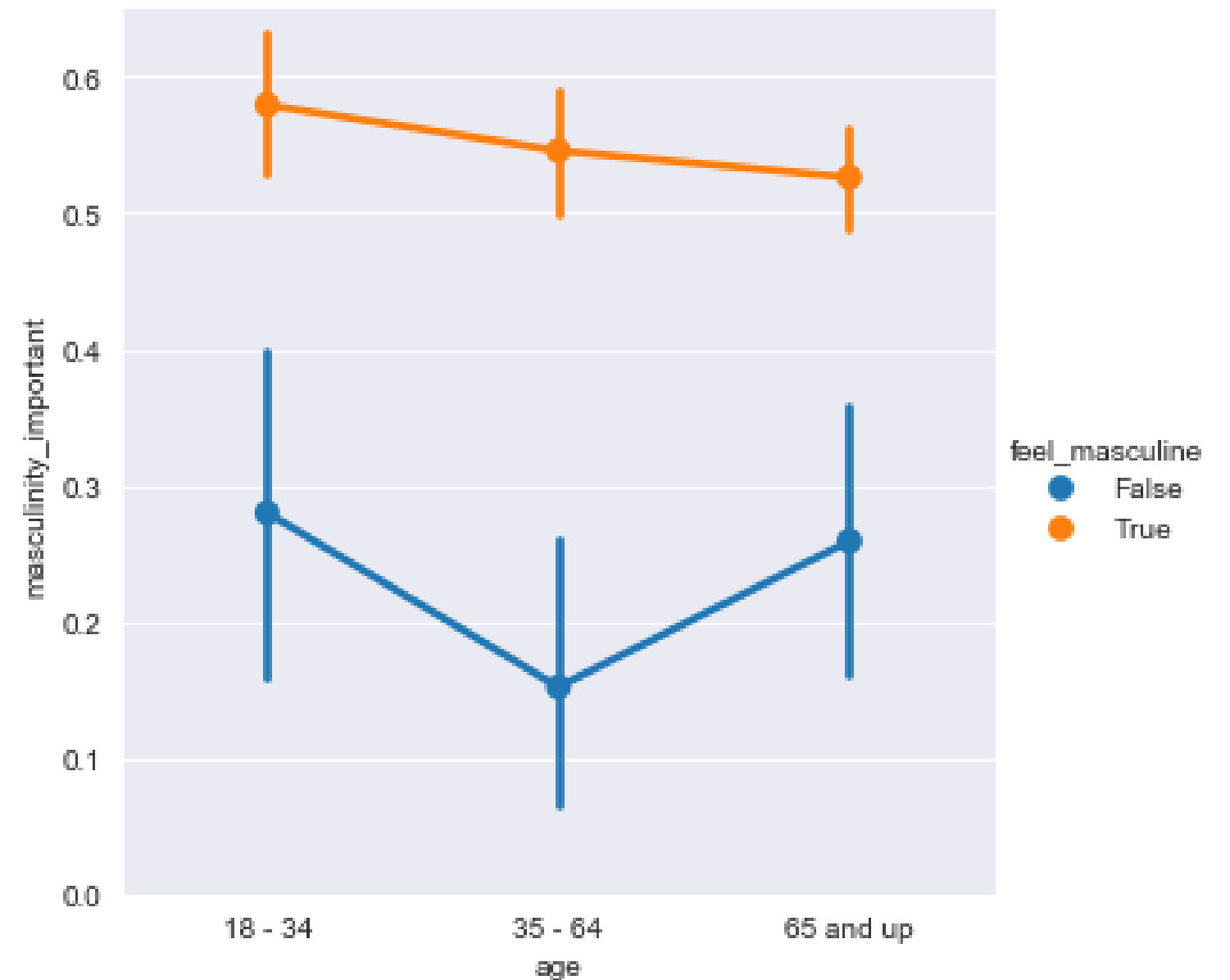
5. Other styles : “darkgrid”

```
sns.set_style("darkgrid")

sns.catplot(x="age",
            y="masculinity_important",
            data=masculinity_data, hue=
            "feel_masculine", kind=
            "point")

plt.show()
```

`sns.set_style("darkgrid")` : style을 “darkgrid”로 변경
⇒ 회색 background + 하얀색 grid



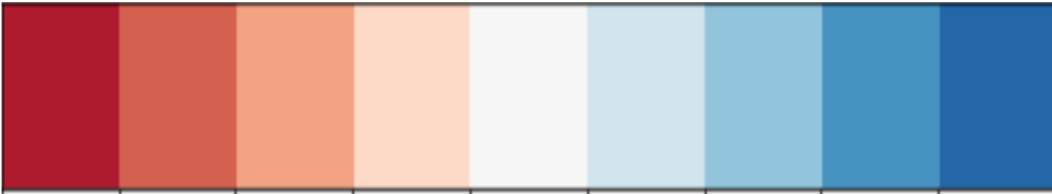
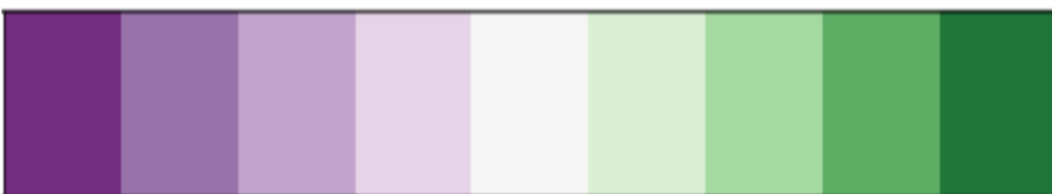

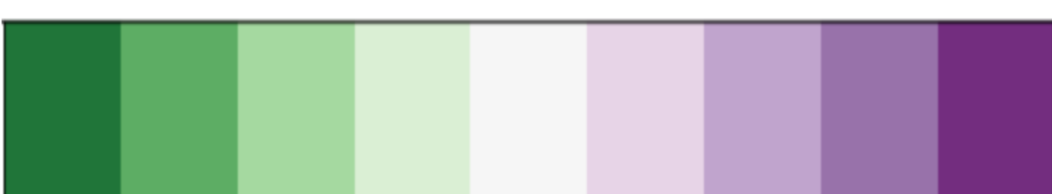
Changing the palette

- Figure "palette" changes the color of the main elements of the plot
- `sns.set_palette()`
- Use preset palettes or create a custom palette

seaborn의 `set_palette()` function 이용 => `sns.set_palette()` :
Plot의 주요 element들의 색깔 변경 가능

1. Seaborn은 많은 preset color palette를 가짐
2. 혹은 사용자가 직접 custom palette을 생성할 수도 있음

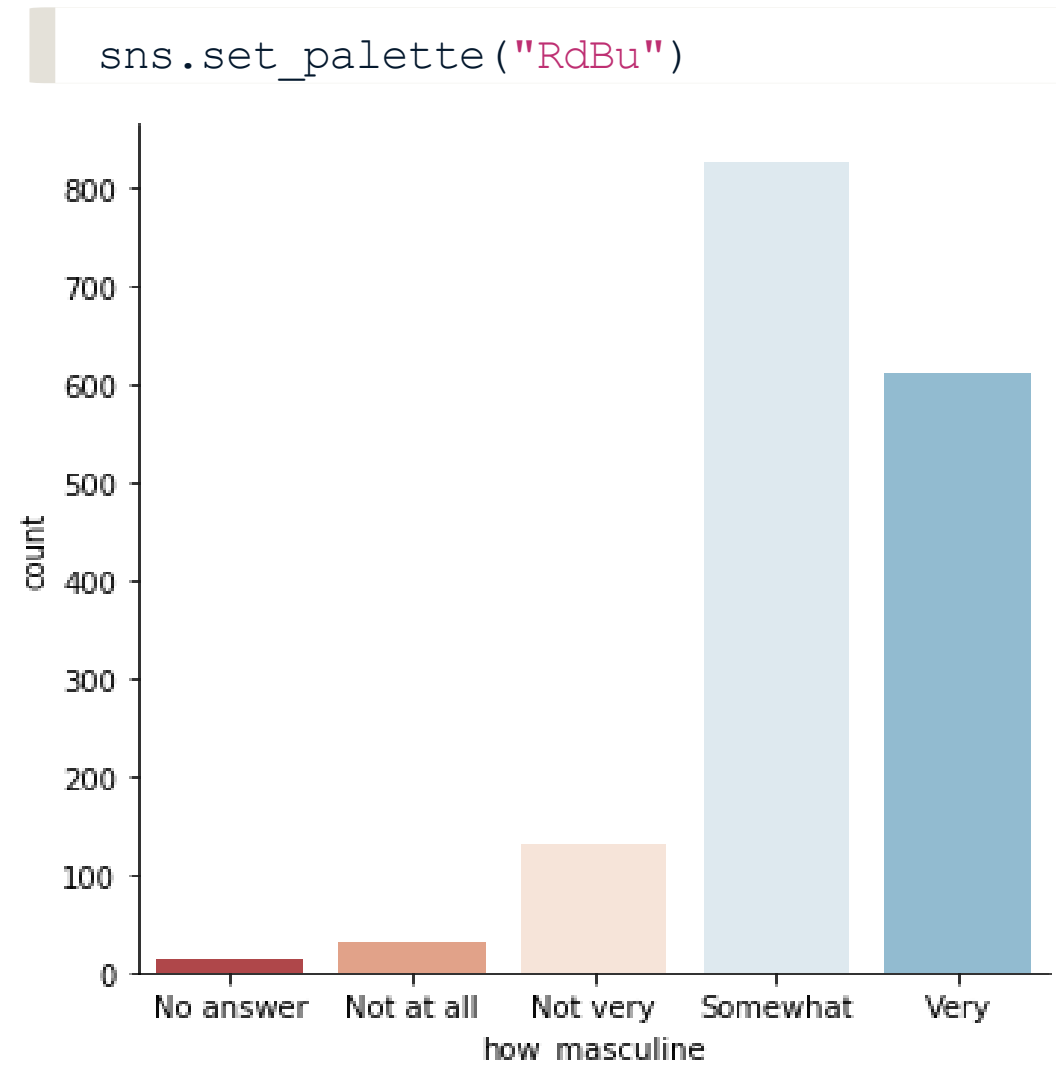
1. Diverging palettes

"RdBu"	
"PRGn"	
"RdBu_r"	
"PRGn_r"	

Seaborn의 preset color palette 예 1) Diverging palettes

Scale의 두 개의 끝 값이 반대되고 중간 지점은 neutral한 경우 diverging palette가 유용

Ex) Red-Blue, Purple-Green
-> palette 이름 끝에 '_r'을 추가하면 palette color가 역순으로 나타남







Ex) 그래프 : Count plot of responses of men reporting how masculine they feel

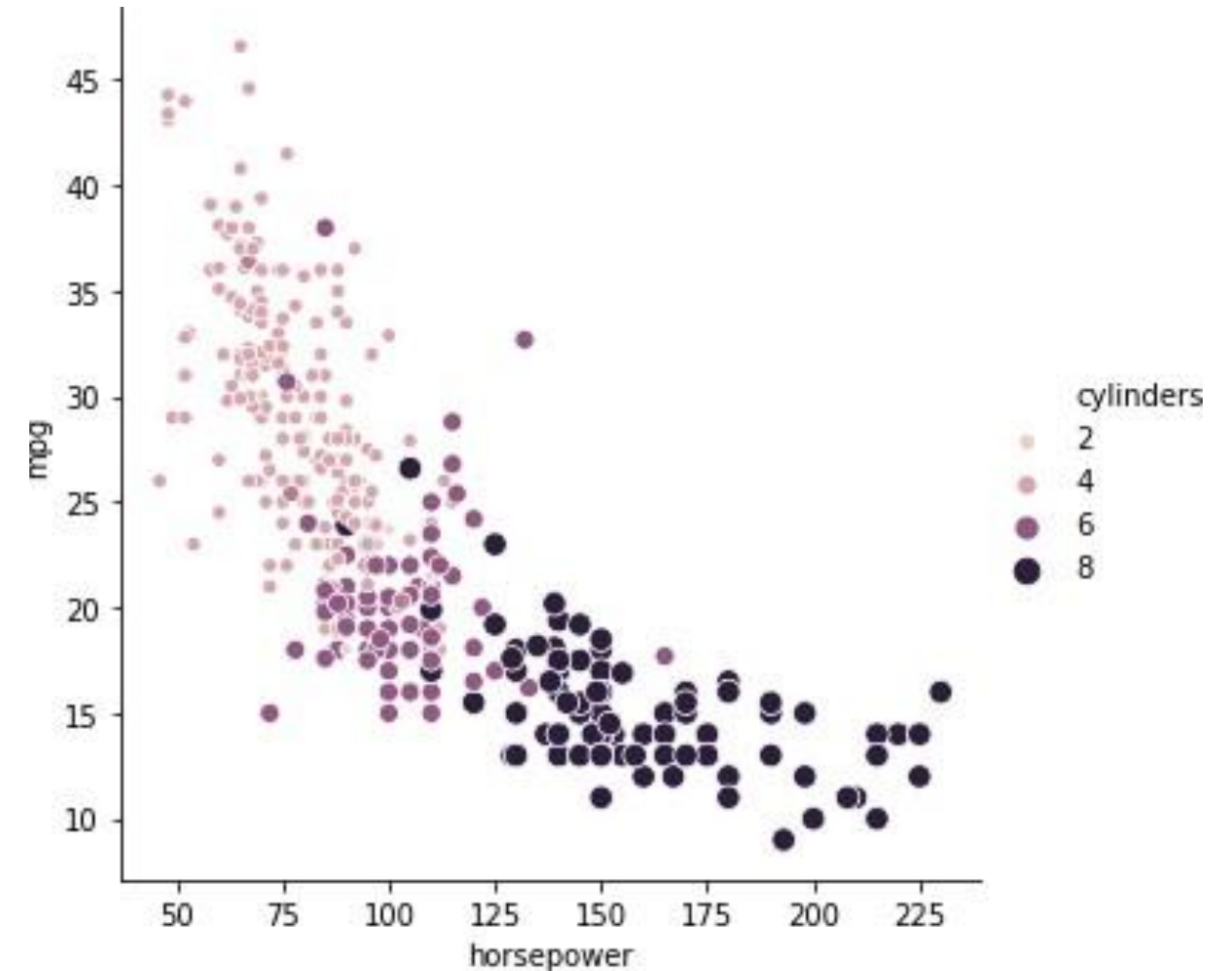
`sns.set_palette("RdBu")` : plot palette를 red-blue diverging으로 지정

⇒ Provide clear contrast between the men who do not feel masculine and the men who do

2. Sequential palettes

"Greys"	
"Blues"	
"PuRd"	
"GnBu"	

Seaborn의 preset color palette 예 2) Sequential palettes
: These are single color or two color moving from white to dark value
=> Great for emphasizing a variable on a continuous scale



Ex) 그래프 : Plot depicting the relationship between a car's horsepower and its mpg
=> 차의 cylinder가 많을 수록 점들의 사이즈가 커지고 어두워짐

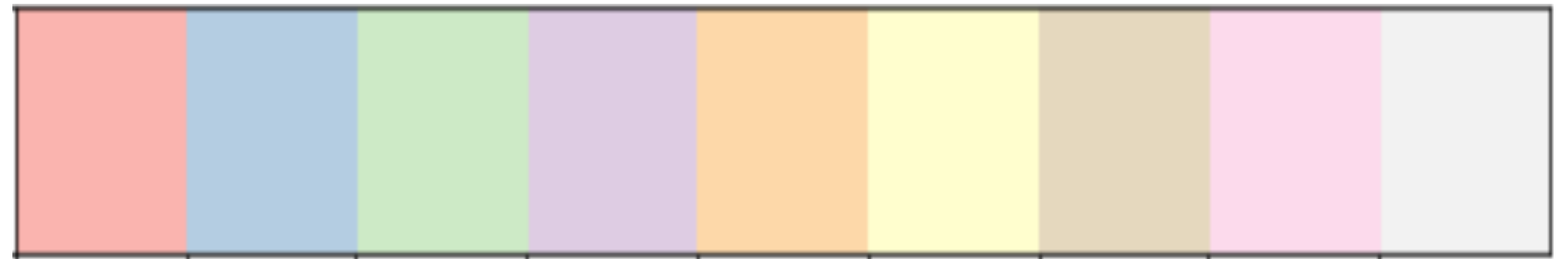
3. Custom palettes

```
custom_palette = ["red", "green", "orange", "blue",  
                  "yellow", "purple"]  
  
sns.set_palette(custom_palette)
```



사용자 정의 palette도 생성 가능
⇒ 1. 리스트 안에 색상명을 써서 사용

```
custom_palette = ['#FBB4AE', '#B3CDE3', '#CCEBC5',  
                  '#DECBE4', '#FED9A6', '#FFFFCC',  
                  '#E5D8BD', '#FDDAEC', '#F2F2F2']  
  
sns.set_palette(custom_palette)
```



사용자 정의 palette도 생성 가능
⇒ 2. 리스트 안에 hex color code를 써서 사용

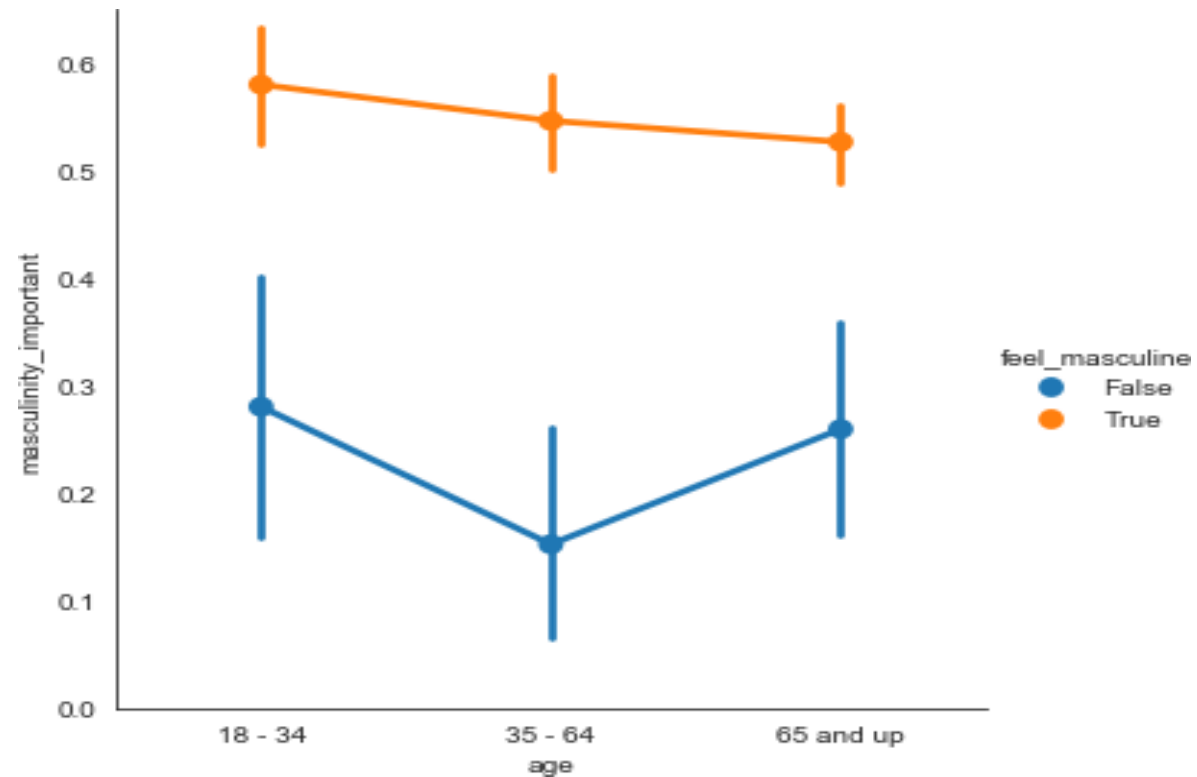
Changing the scale

- Figure "context" changes the scale of the plot elements and labels
- `sns.set_context()`
- Smallest to largest: "paper", "notebook", "talk", "poster"

set_context() function : plot의 scale 변경 가능
Scale option (작은 것부터 큰 것 순으로):
1. "paper" -> 디폴트
2. "notebook"
3. "talk"
4. "poster"

"paper"

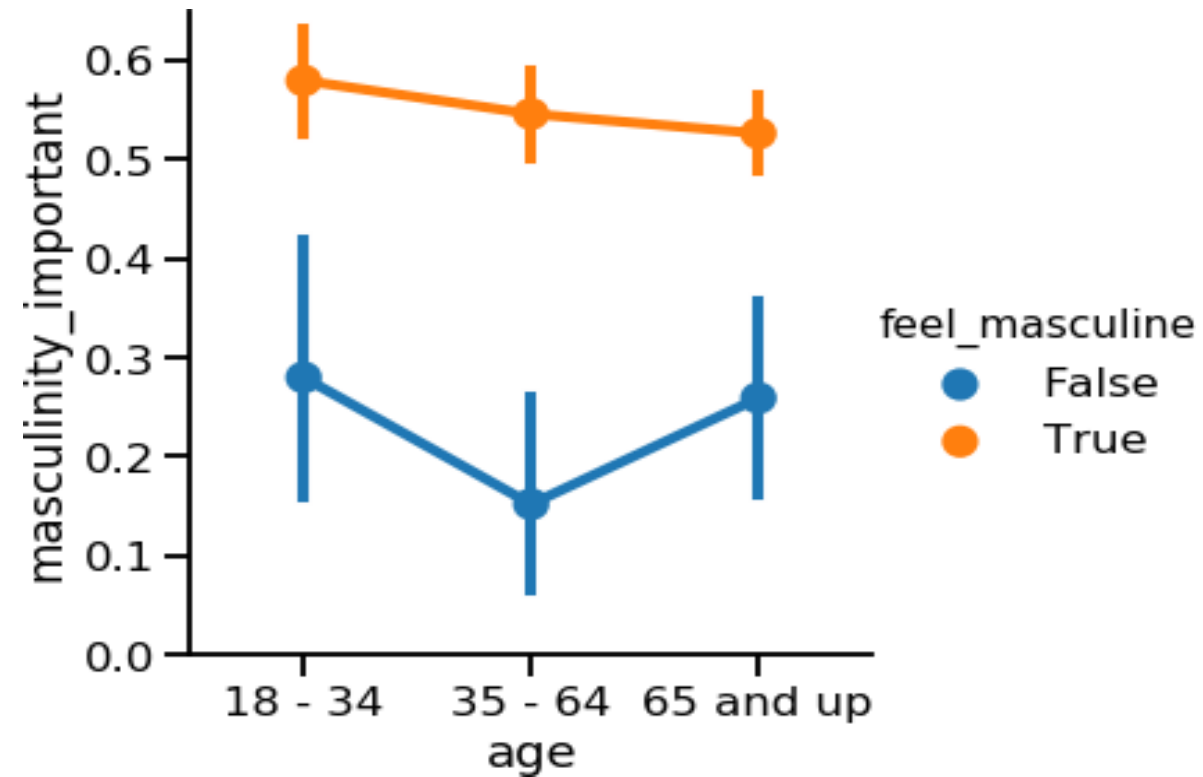
```
sns.catplot(x="age",  
            y="masculinity_important",  
            data=masculinity_data, hue=  
            "feel_masculine", kind=  
            "point")  
  
plt.show()
```



"talk"

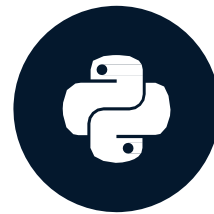
```
sns.set_context("talk")  
  
sns.catplot(x="age",  
            y="masculinity_important",  
            data=masculinity_data, hue=  
            "feel_masculine", kind=  
            "point")  
  
plt.show()
```

"talk"이 "paper"보다 더 scale이 큼을 알 수 있음



Adding titles and labels

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

FacetGrid vs. AxesSubplot objects

Seaborn plots create two different types of objects: `FacetGrid` and `AxesSubplot`

```
g = sns.scatterplot(x="height", y="weight", data=df) type(g)
```

```
> matplotlib.axes._subplots.AxesSubplot
```

Seaborn plot function 은 2개의 object 타입 형성 : 1. FacetGrid 2. AxesSubplot

To figure out which type of object you're working with:

1. Plot output 을 변수에 할당 : `g = sns.scatterplot()`
2. `type(g)` : object type을 반환 : 이 `scatterplot`이 `AxesSubplot` 타입임을 알 수 있음

FacetGrid vs. AxesSubplot objects

Object Type	Plot Types	Characteristics
FacetGrid	<code>relplot()</code> , <code>catplot()</code>	Can create subplots
AxesSubplot	<code>scatterplot()</code> , <code>countplot()</code> ,etc.	Only creates a single plot

Object Type

1. FacetGrid :
 - `relplot()`, `catplot()` : subplot 생성
2. AxesSubplot:
 - `scatterplot()`, `countplot()` : single plot 생성

1. Adding title & Adjusting height of title in FacetGrid

```
g = sns.catplot(x="Region",
                y="Birthrate", data=gdp_data,
                kind="box")

g.fig.suptitle("New Title",
              y=1.03)

plt.show()
```

앞에서 밝혔듯, `catplot()` : FacetGrid object

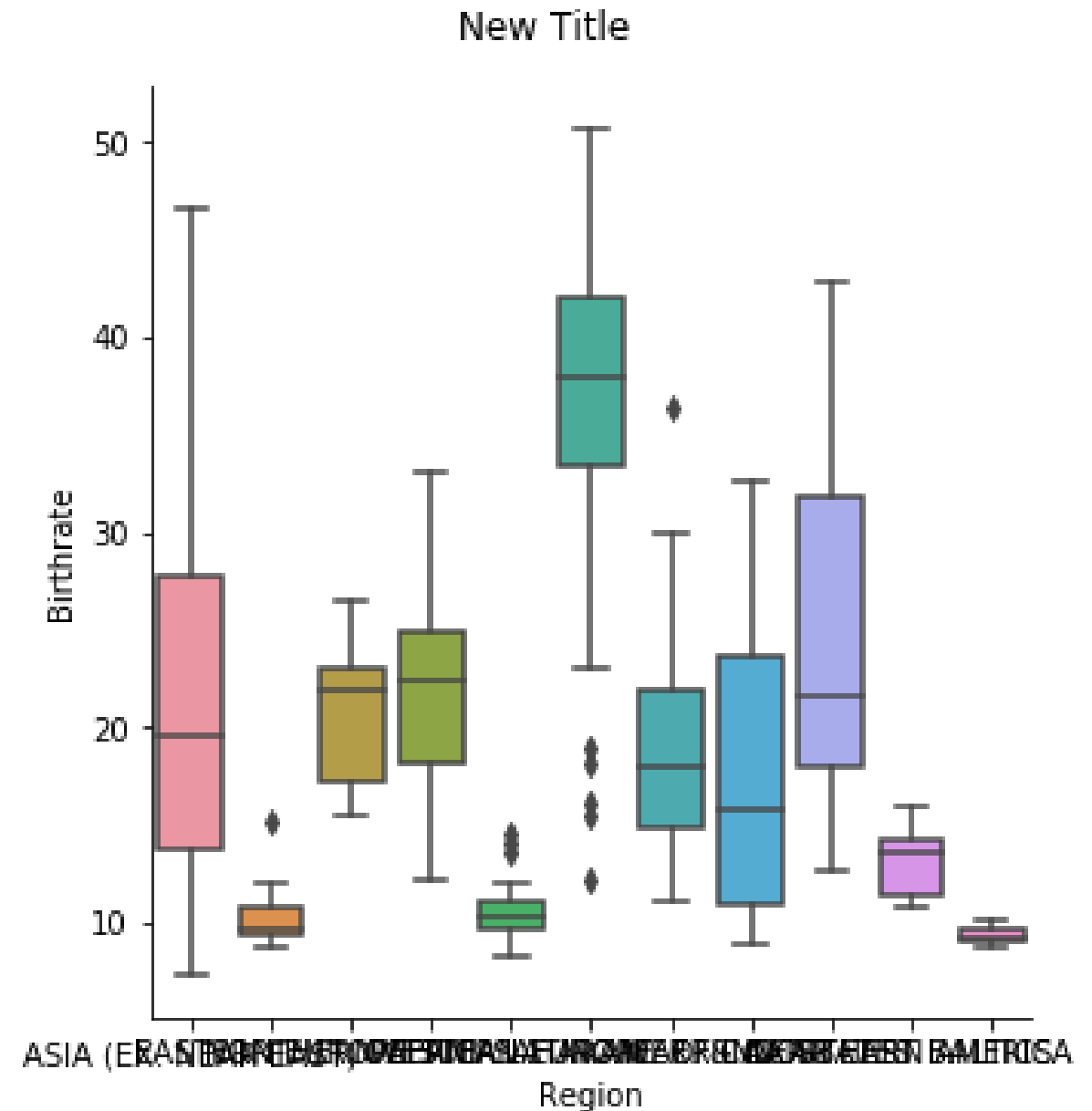
FacetGrid object에 타이틀 추가하는 법:

1. 생성한 plot을 변수에 할당 (g)
 2. `g.fig.suptitle()` 함수를 통해 타이틀 설정 가능
- => 전체 figure에 대한 타이틀 추가

타이틀의 높이를 'y' parameter를 통해 조절 가능

디폴트 : `y=1`

디폴트보다 조금 더 높게 설정하려면, `y=1.03`



2. Adding a title to AxesSubplot

FacetGrid

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")  
  
g.fig.suptitle("New Title",  
               y=1.03)
```

FacetGrid object에 타이틀 추가하는 법:
g.fig.suptitle()

AxesSubplot

```
g = sns.boxplot(x="Region",  
                y="Birthrate",  
                data=gdp_data)  
  
g.set_title("New Title",  
            y=1.03)
```

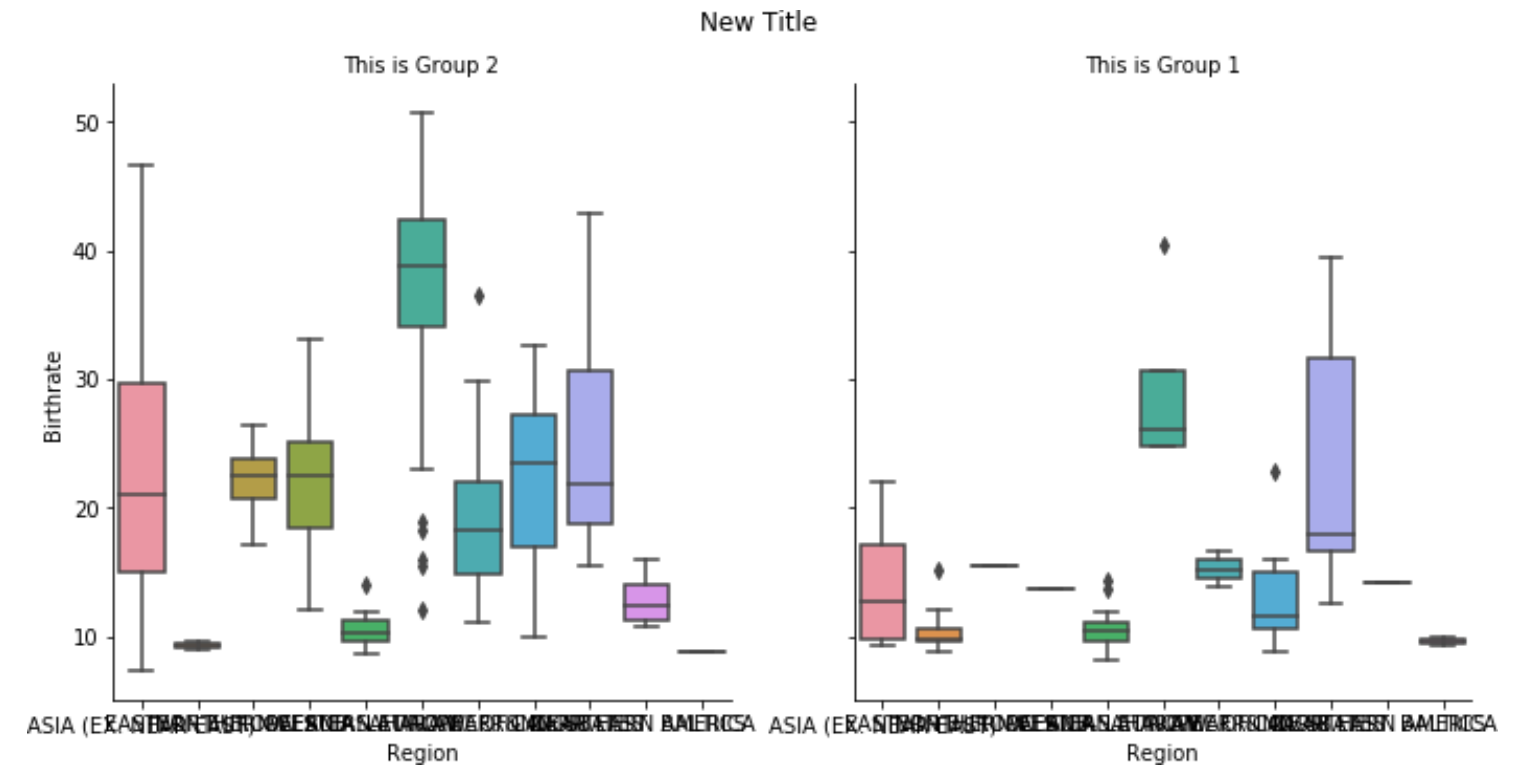
AxesSubplot object 에 타이틀 추가하는 법:
1. 생성한 plot을 변수에 할당 (g)
2. g.set_title() function 이용

Titles for subplots

```
g = sns.catplot(x="Region",
                y="Birthrate",
                data=gdp_data,
                kind="box",
                col="Group")

g.fig.suptitle("New Title",
               y=1.03)

g.set_titles("This is {col_name}")
```



각 그룹에 대해 subplot 형성 -> subplot별로 타이틀 넣는 법:

1. FacetGrid objec인 plot을 변수에 할당 (g)
2. g.set_titles() : 각 subplot별로 타이틀 설정

(<-> g.fig_titles() : 전체 plot에 대한 타이틀 설정)

각 subplot의 title에 variable name을 넣고 싶다면,
{col_name} 사용

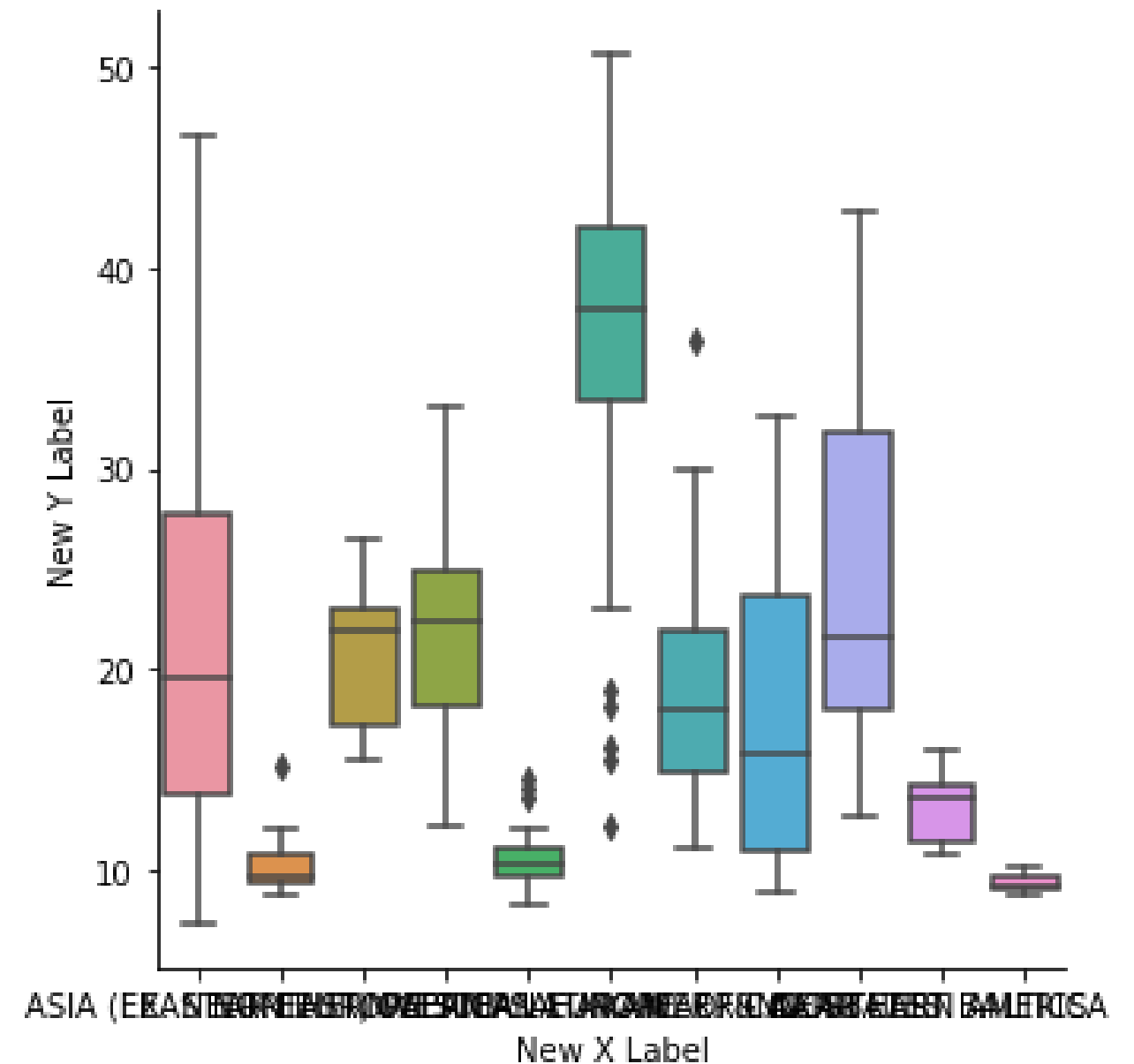
Adding axis labels

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")
```

```
g.set(xlabel="New X Label",  
      ylabel="New Y Label")
```

```
plt.show()
```

Axis label 추가하는 법:
set() function 이용
-> set() 함수의 xlabel, ylabel 파라미터 이용



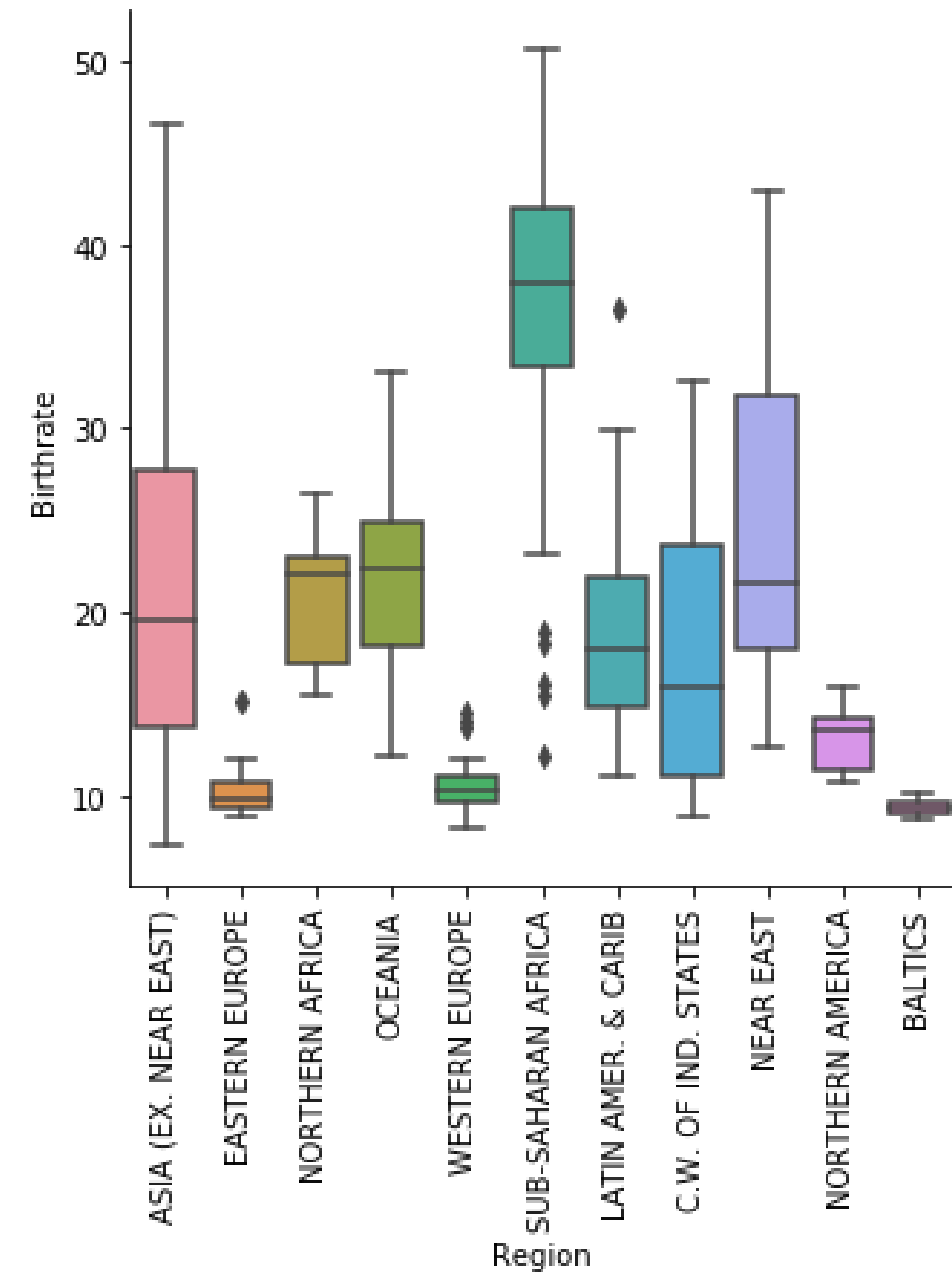
Rotating x-axis tick labels

```
g = sns.catplot(x="Region",  
                y="Birthrate",  
                data=gdp_data,  
                kind="box")  
plt.xticks(rotation=90)  
plt.show()
```

x축의 tick label 회전시키는 법:

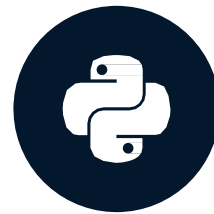
plot object 에 대한 함수 사용하지 않음! -> 대신, matplotlib의 함수 사용

plt.xticks() function 이용 -> rotation 파라미터 이용
=> rotation = 90 : 90도 회전



Putting it all together

INTRODUCTION TO DATA VISUALIZATION WITH SEABORN



Erin Case
Data Scientist

Relational plots

- Show the relationship between two quantitative variables
- Examples: scatter plots, line plots

```
sns.relplot(x="x_variable_name",  
            y="y_variable_name", data=pandas_df,  
            kind="scatter")
```

Categorical plots

- Show the distribution of a quantitative variable within categories defined by a categorical variable
- Examples: bar plots, count plots, box plots, point plots

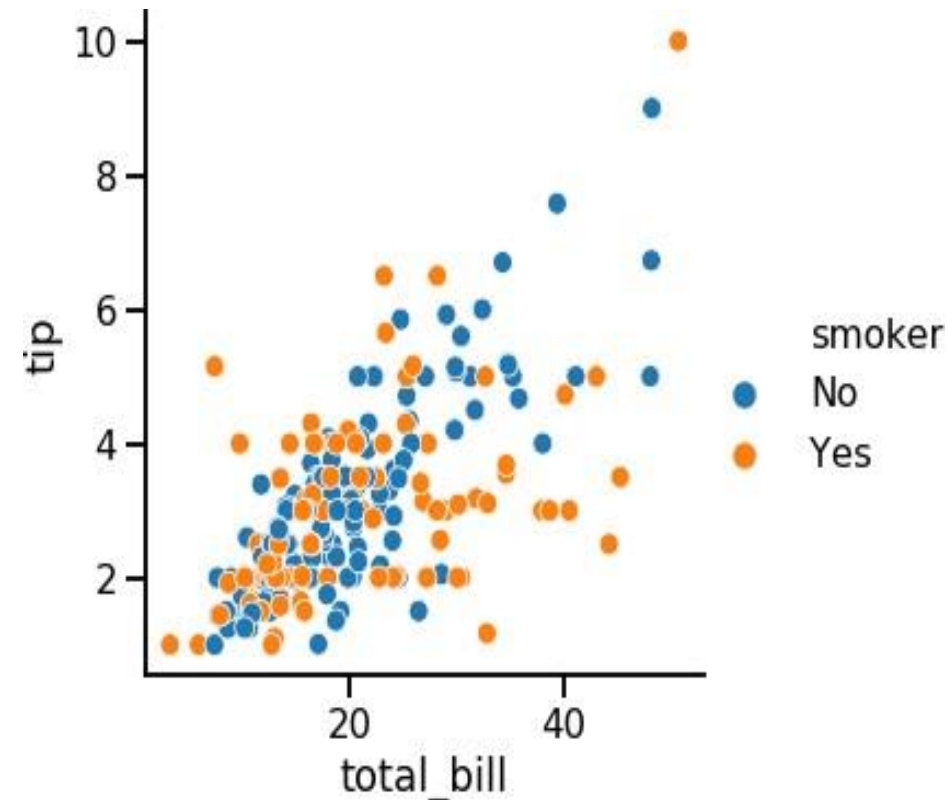
```
sns.catplot(x="x_variable_name",  
            y="y_variable_name", data=pandas_df,  
            kind="bar")
```

어떤 타입의 plot을 생성하고 싶은 지 고르기

1. Relational plot => relplot() 으로 생성 가능 : plot의 타입은 kind 매개변수로 설정 가능 kind = "scatter" 또는 "line"
2. Categorical plot => catplot() 으로 생성 가능 : plot의 타입은 kind 매개변수로 설정 가능 kind = "bar" 또는 "count" 또는 "box" 또는 "point"

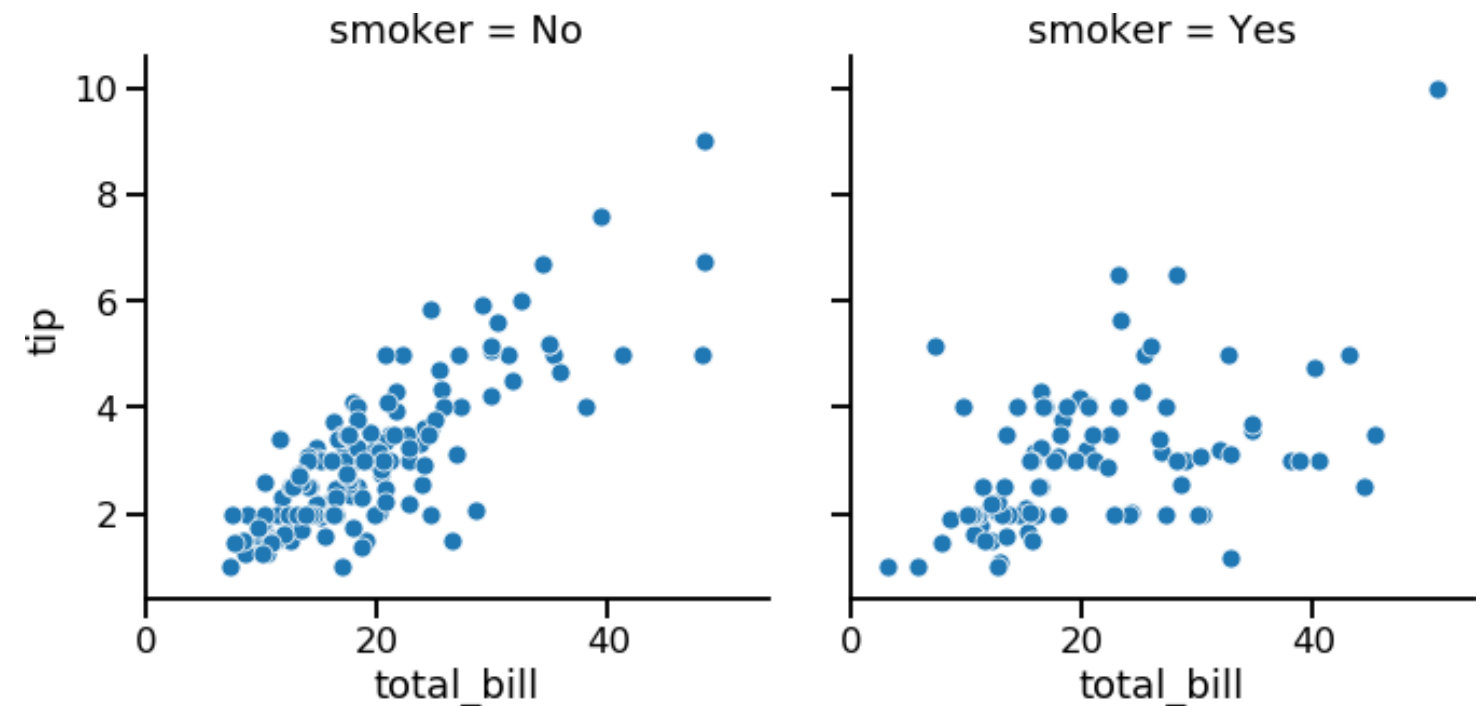
1. Adding a third variable (hue)

Setting `hue` will create subgroups that are displayed as different colors on a single plot.



2. Adding a third variable (row/col)

Setting `row` and/or `col` in `relplot()` or `catplot()` will create subgroups that are displayed on separate subplots.



Add a third dimension to our plot:

1. 'hue' parameter 이용 :
=> single plot 생성. 단, variable values에 기초하여 그룹별로 다른 색으로 표시
2. `relplot()` 또는 `catplot()`의 'row' or 'col' parameter 이용:
=> 각각의 subgroup별로 subplot 생성 (즉, 여러 개의 plot)

Customization

- Change the background: `sns.set_style()`
- Change the main element colors: `sns.set_palette()`
- Change the scale: `sns.set_context()`

Adding a title

Object Type	Plot Types	How to Add Title
FacetGrid	<code>relplot()</code> , <code>catplot()</code>	<code>g.fig.suptitle()</code>
AxesSubplot	<code>scatterplot()</code> , <code>countplot()</code> , etc.	<code>g.set_title()</code>

Final touches

Add x- and y-axis labels:

```
g.set(xlabel="new x-axis label", ylabel  
      ="new y-axis label")
```

Rotate x-tick labels:

```
plt.xticks(rotation=90)
```