Seaborn - Visualizing statistical relationships

- relplot() - scatter and line plots

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

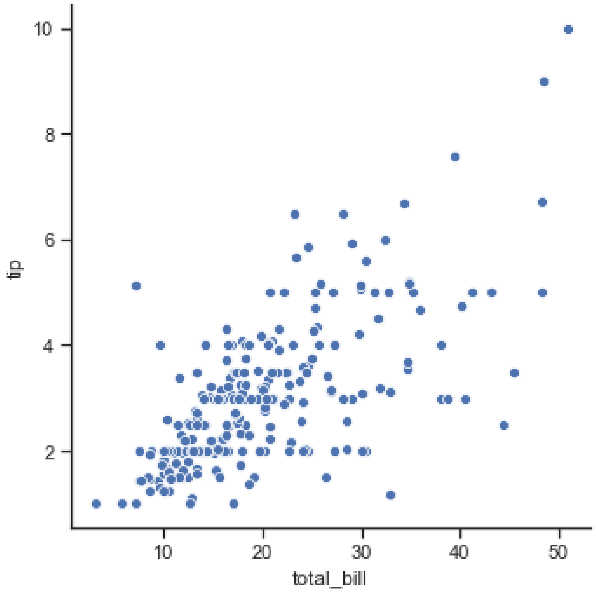
In [2]:
```
sns.set(style="ticks")
```

In [3]:
```
tips = sns.load_dataset("tips")
tips.head()
```

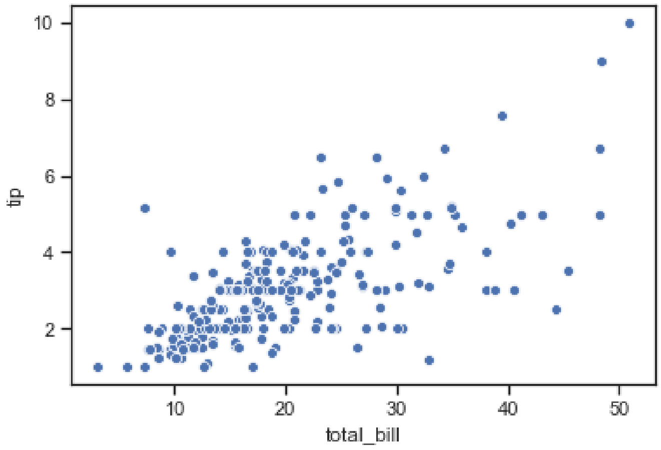
Out[3]:

	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

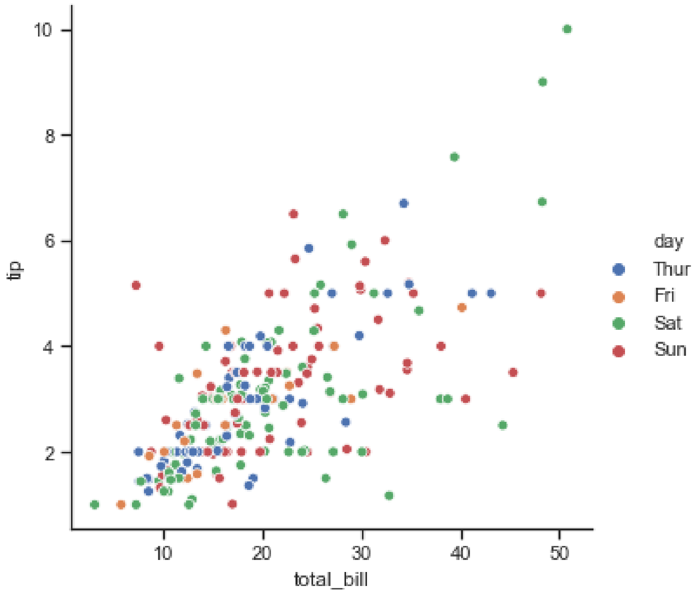
In [4]:
```
# default kind="scatter"

sns.relplot(x="total_bill", y="tip", data=tips);
```



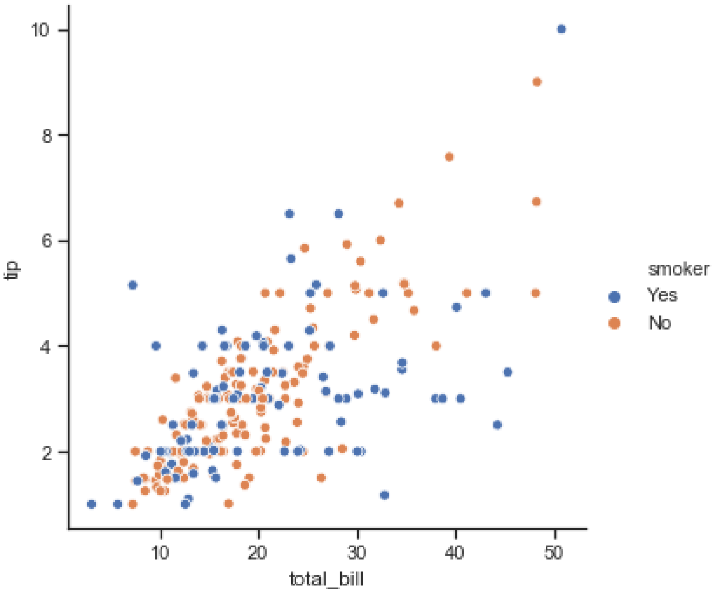
In [5]:
```
sns.scatterplot(x="total_bill", y="tip", data=tips);
```



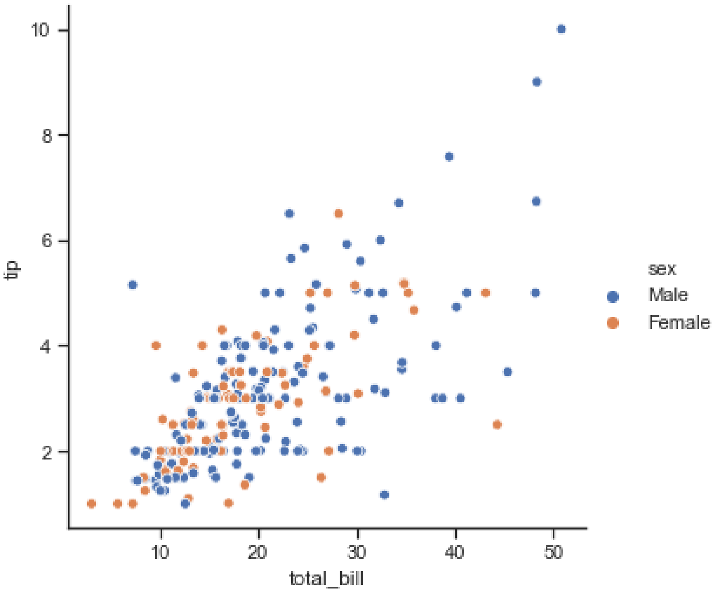
In [6]:
```
# Add a third dimension - color by a third variable (Group by)
# hue semantic - color of a point now has a meaning

sns.relplot(x="total_bill", y="tip", hue="day", data=tips);
```



In [7]:
```
sns.relplot(x="total_bill", y="tip", hue="smoker", data=tips);
```

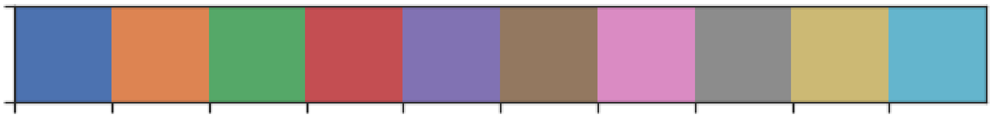


In [8]:
```
sns.relplot(x="total_bill", y="tip", hue="sex", data=tips);
```



In [9]:
```
# hue semantic categorical in above examples (qualitative palette)
# If hue semantic is numeric, sequential palette is used for color
```

In [10]:
```
current_palette = sns.color_palette()
sns.palplot(current_palette)
```

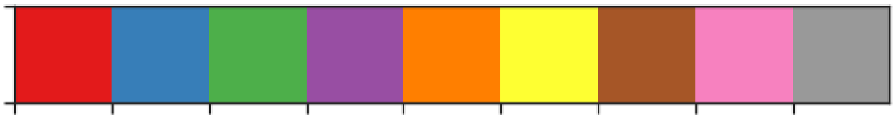


In [11]:
```
sns.choose_colorbrewer_palette(data_type="qualitative")
```

name Set1

n 9

desat 1.00



Out[11]:
```
[(0.8941176470588235, 0.10196078431372557, 0.10980392156862737),
 (0.21568627450980393, 0.4941176470588236, 0.7215686274509804),
 (0.3019607843137256, 0.6862745098039216, 0.29019607843137263),
 (0.5960784313725492, 0.3058823529411765, 0.6392156862745098),
 (1.0, 0.4980392156862745, 0.0),
 (0.9999999999999998, 1.0, 0.19999999999999996),
 (0.6509803921568629, 0.33725490196078434, 0.1568627450980391),
 (0.9686274509803922, 0.5058823529411766, 0.7490196078431374),
 (0.6, 0.6, 0.6)]
```

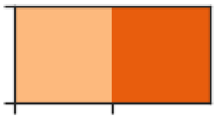
In [12]:
```
colors = sns.choose_colorbrewer_palette(data_type="sequential")
```

name Oranges

n 2

desat 1.00

variant regular

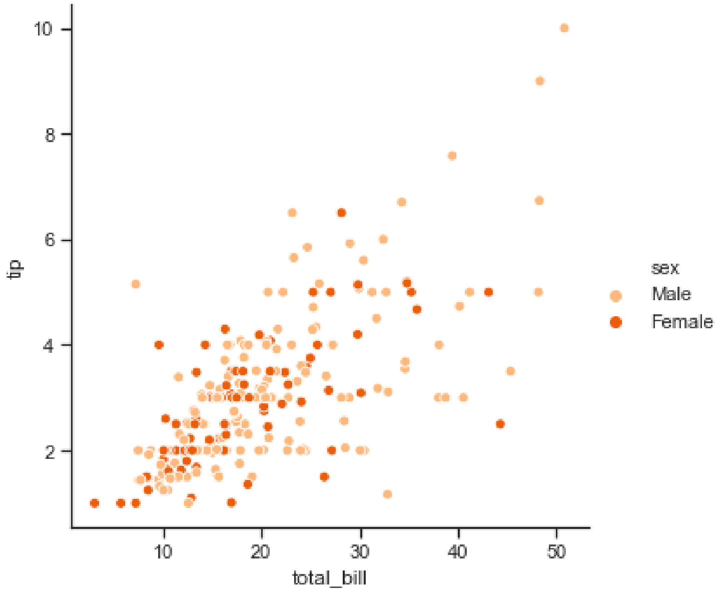


In [15]:
```
colors
```

Out[15]:
```
[(0.9921568627450981, 0.726797385620915, 0.49150326797385624),
 (0.9137254901960783, 0.36862745098039207, 0.0509803921568629)]
```

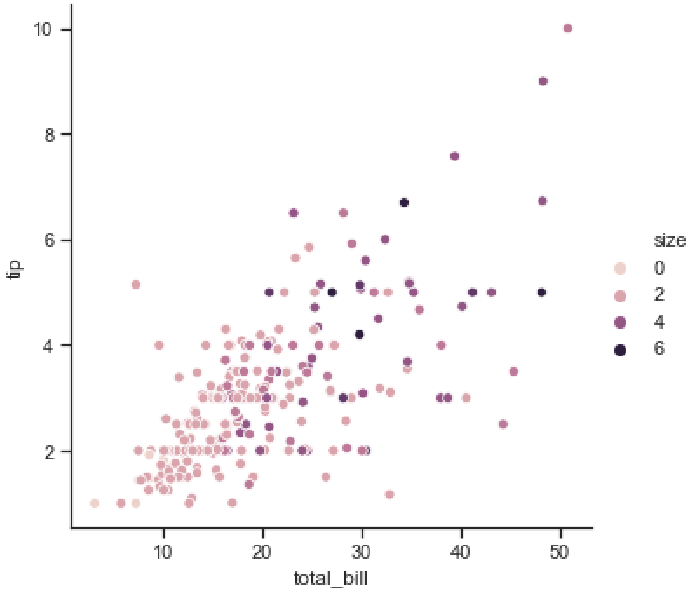
In [16]:

```
sns.relplot(x="total_bill", y="tip", hue="sex",
            palette=colors,
            data=tips);
```



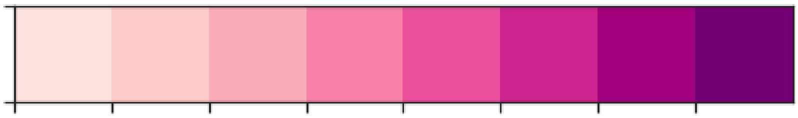
In [17]:

```
sns.relplot(x="total_bill", y="tip", hue="size", data=tips);
```



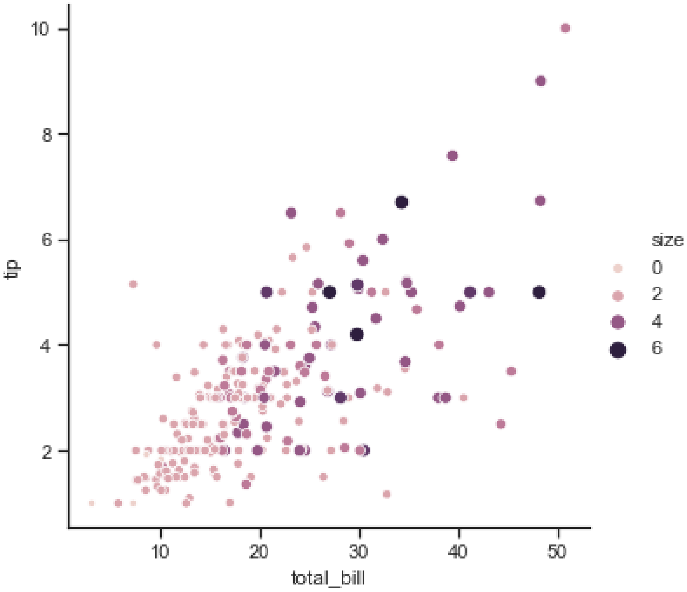
In [18]:

```
sequential_colors = sns.color_palette("RdPu", 8)
sns.palplot(sequential_colors)
```

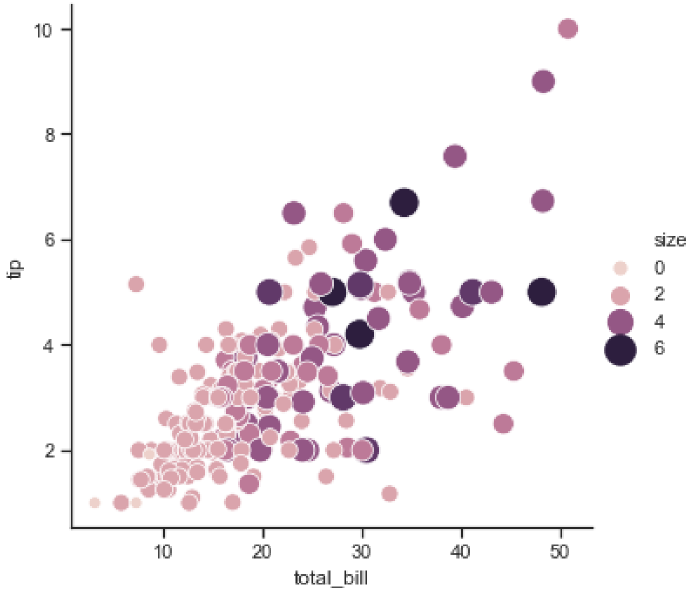


In [19]:

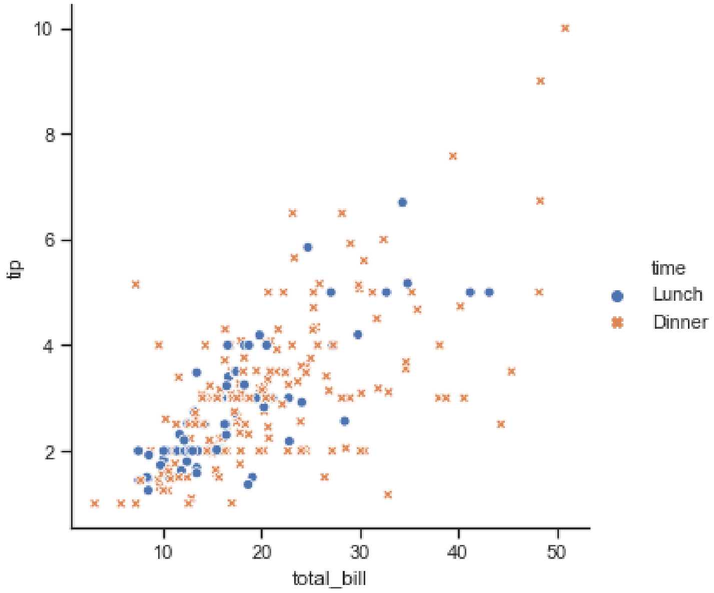
```
# semantic variable for the size of each point
# literal value not used for the area
# range of values in data units normalized into a range in area units

sns.relplot(x="total_bill", y="tip",
            size="size", hue="size", data=tips);
```



In [20]:
```
# customize area sizes range

sns.relplot(x="total_bill", y="tip",
            size="size", sizes=(50, 300),
            hue="size", data=tips);
```

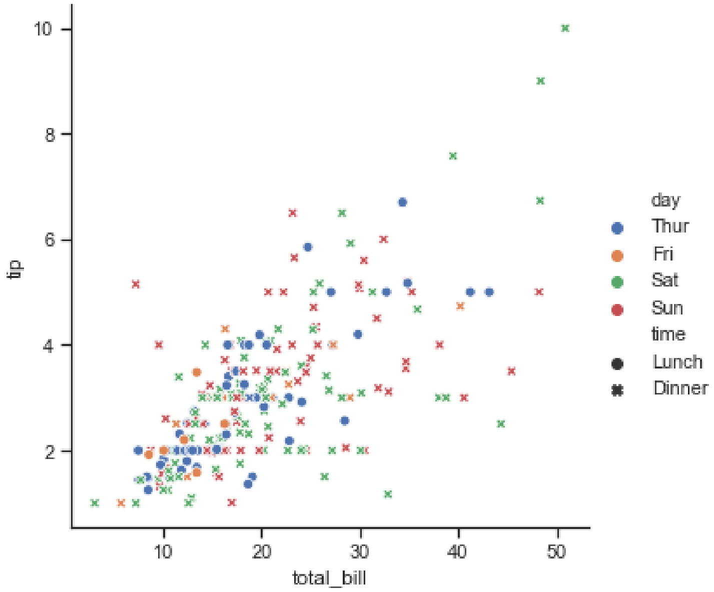


In [21]:
```
# style - vary the marker (same grouping variable)

sns.relplot(x="total_bill", y="tip", hue="time",
            style="time", data=tips);
```



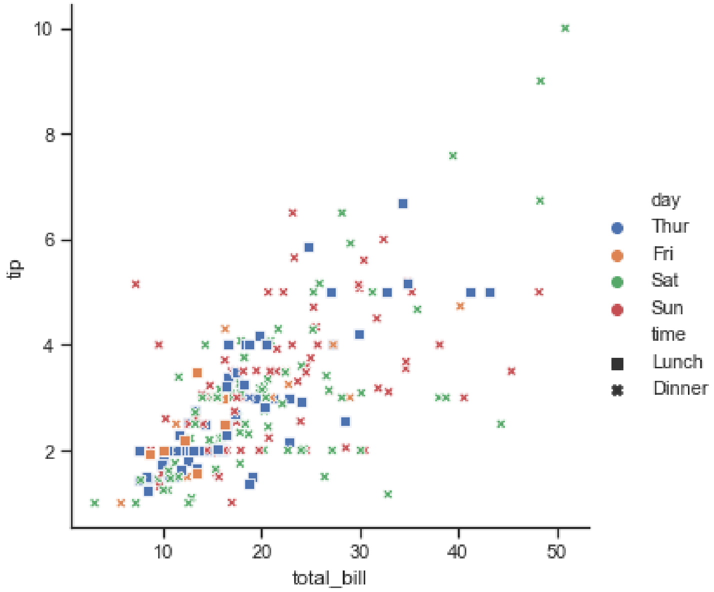
In [22]:

```python
# style - vary the marker (different grouping variables)

sns.relplot(x="total_bill", y="tip", hue="day",
            style="time", data=tips);
```



In [23]:

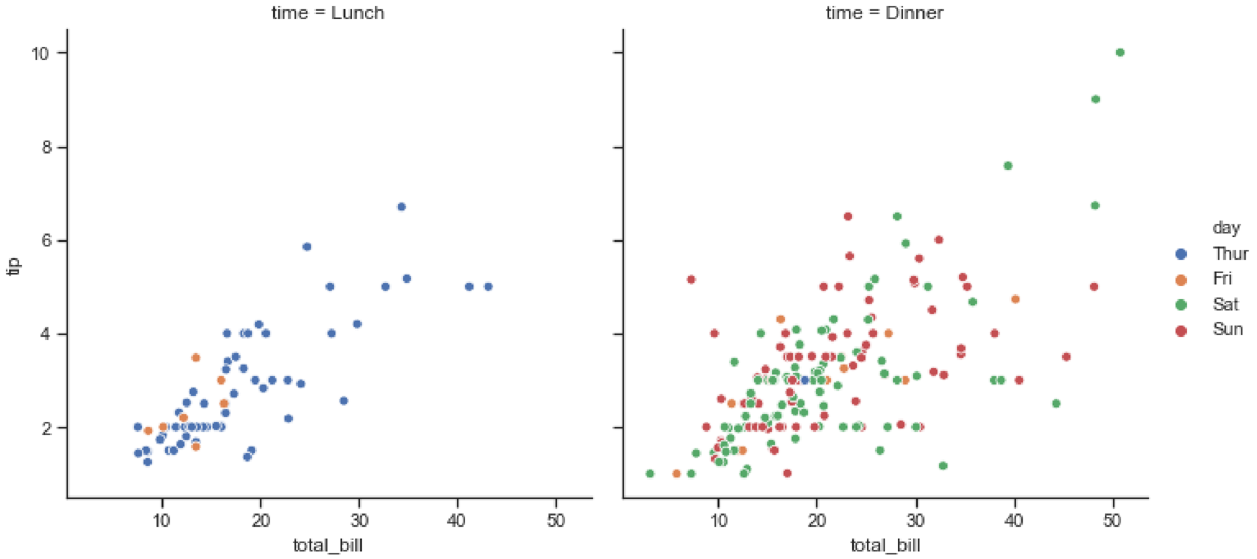
```python
sns.relplot(x="total_bill", y="tip", hue="day",
            style="time",
            markers = {"Lunch": "s", "Dinner": "X"},
            data=tips);
```



Showing multiple relationships

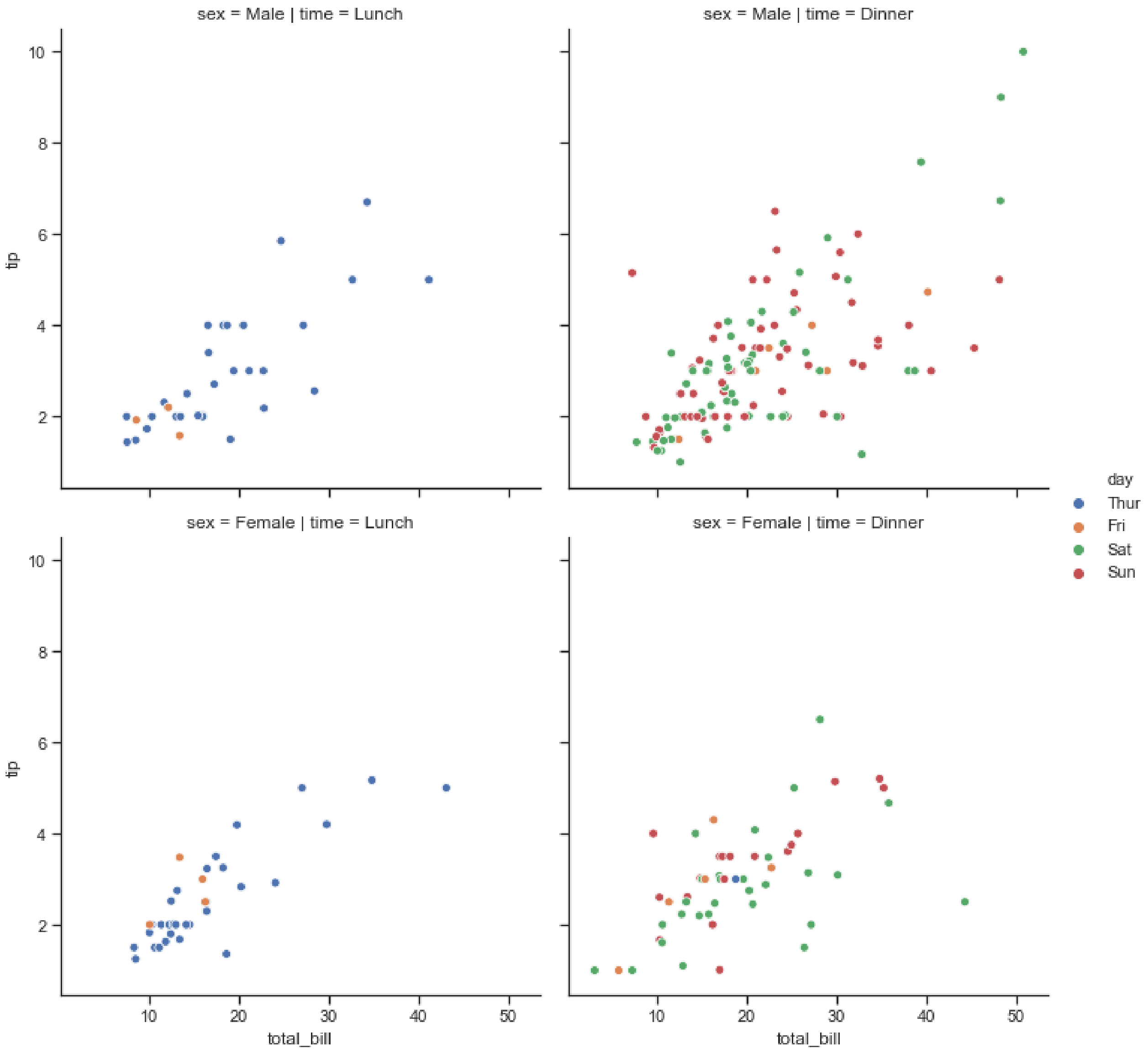
- when a relationship between two variables depends on more than one variable?

In [24]:
```
sns.relplot(x="total_bill", y="tip", hue="day",
            col="time", data=tips);
```

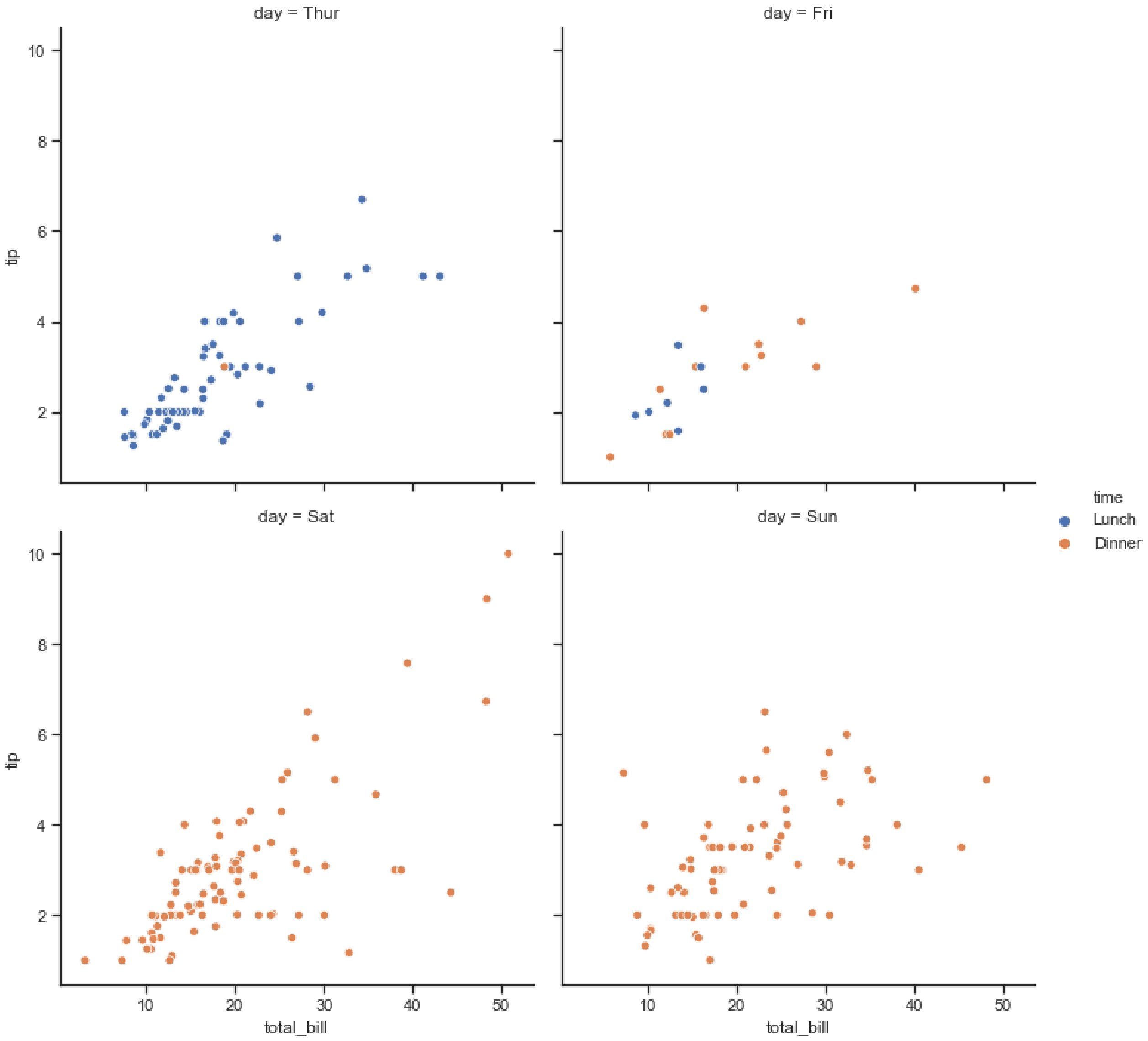


In [25]:

```
sns.relplot(x="total_bill", y="tip", hue="day",
            col="time", row="sex", data=tips);
```

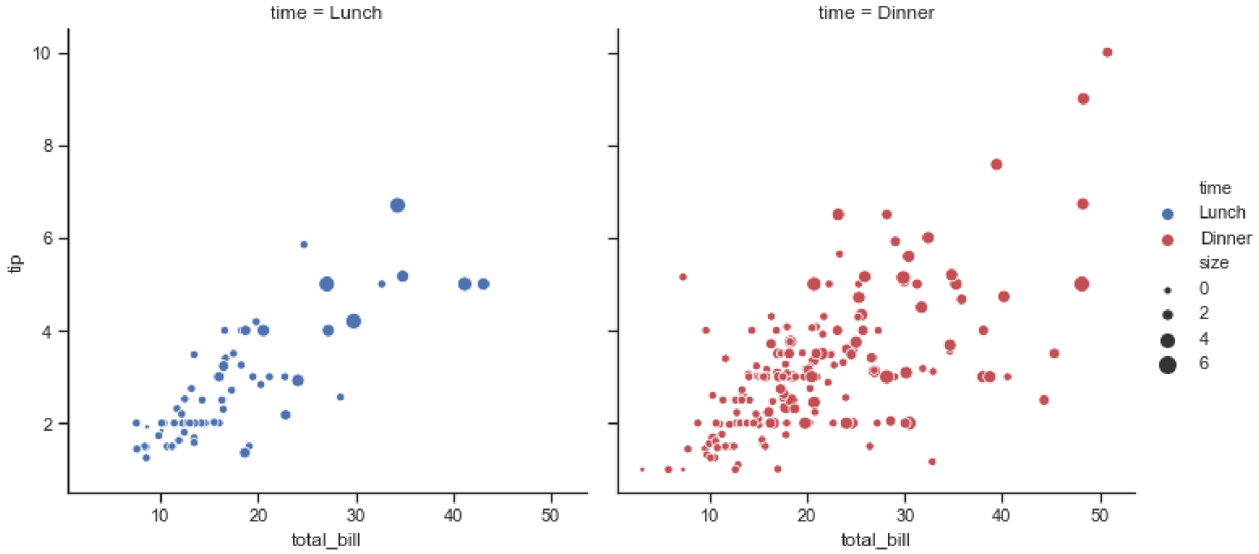


```
In [26]: sns.relplot(x="total_bill", y="tip", hue="time",
                     col="day",  col_wrap=2, data=tips);
```



In [27]:

```
g = sns.relplot(x="total_bill", y="tip", hue="time", size="size",
                palette=["b", "r"], sizes=(10, 100),
                col="time", data=tips);
```



Line Plots

In [28]:

```
sns.set(style="whitegrid")
np.random.seed(54321)
```

In [29]:

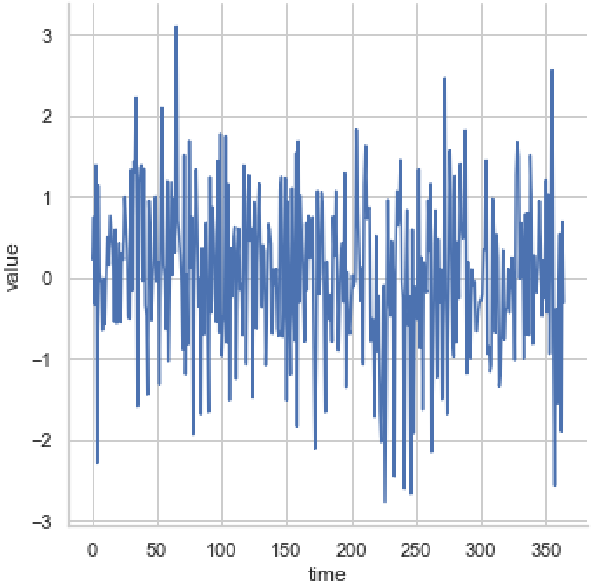
```
df = pd.DataFrame(dict(time=np.arange(365),
                       value=np.random.randn(365)))
df.head()
```

Out[29]:

	time	value
0	0	0.223979
1	1	0.744591
2	2	-0.334269
3	3	1.389172
4	4	-2.296095

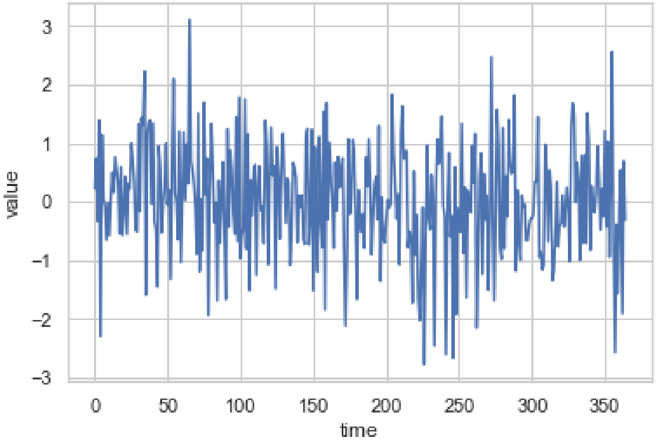
In [30]:

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



In [31]:

```
sns.lineplot(x="time", y="value", data=df);
```



In [32]:

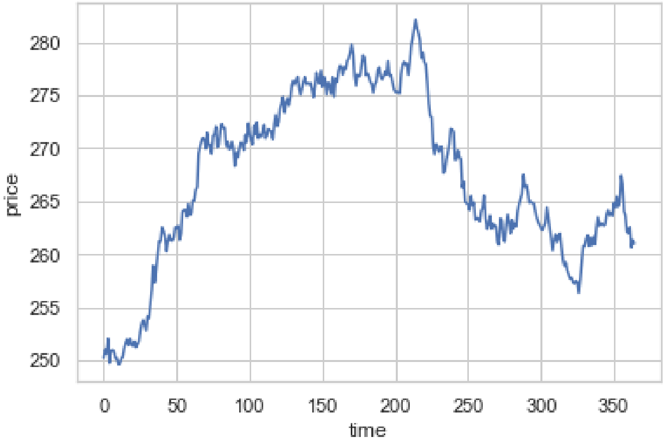
```
df["price"] = 250 + df["value"].cumsum()
df.head()
```

Out[32]:

	time	value	price
0	0	0.223979	250.223979
1	1	0.744591	250.968570
2	2	-0.334269	250.634301
3	3	1.389172	252.023472
4	4	-2.296095	249.727378

In [33]:

```
sns.lineplot(x="time", y="price", data=df);
```



In [34]:

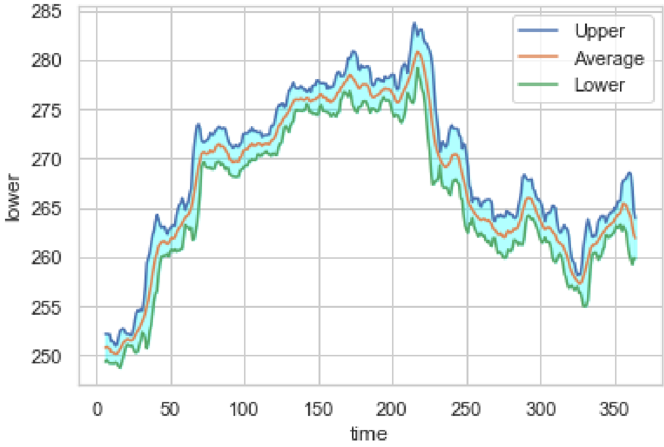
```
df["mean"] = df['price'].rolling(7).mean()
df["sd"]  = df["price"].rolling(7).std()
df["upper"] = df["mean"] + (2 * df["sd"])
df["lower"] = df["mean"] - (2 * df["sd"])
df.head(12)
```

Out[34]:

	time	value	price	mean	sd	upper	lower
0	0	0.223979	250.223979	NaN	NaN	NaN	NaN
1	1	0.744591	250.968570	NaN	NaN	NaN	NaN
2	2	-0.334269	250.634301	NaN	NaN	NaN	NaN
3	3	1.389172	252.023472	NaN	NaN	NaN	NaN
4	4	-2.296095	249.727378	NaN	NaN	NaN	NaN
5	5	1.141205	250.868583	NaN	NaN	NaN	NaN
6	6	0.055448	250.924030	250.767188	0.712697	252.192582	249.341793
7	7	-0.082760	250.841270	250.855372	0.671268	252.197908	249.512836
8	8	-0.651688	250.189583	250.744088	0.712668	252.169425	249.318752
9	9	-0.016022	250.173561	250.678268	0.745039	252.168346	249.188190
10	10	-0.574465	249.599096	250.331929	0.554659	251.441246	249.222611
11	11	0.048143	249.647239	250.320480	0.569836	251.460153	249.180808

In [35]:

```
ax = sns.lineplot(x="time", y="upper", data=df, label="Upper")
ax = sns.lineplot(x="time", y="mean", data=df, label = "Average")
ax = sns.lineplot(x="time", y="lower", data=df, label = "Lower")
ax.fill_between(df["time"], df['upper'], df['lower'],
                color='cyan', alpha=0.3);
```



In [ ]:

In [ ]: