Data Processing and Visualizations using Python

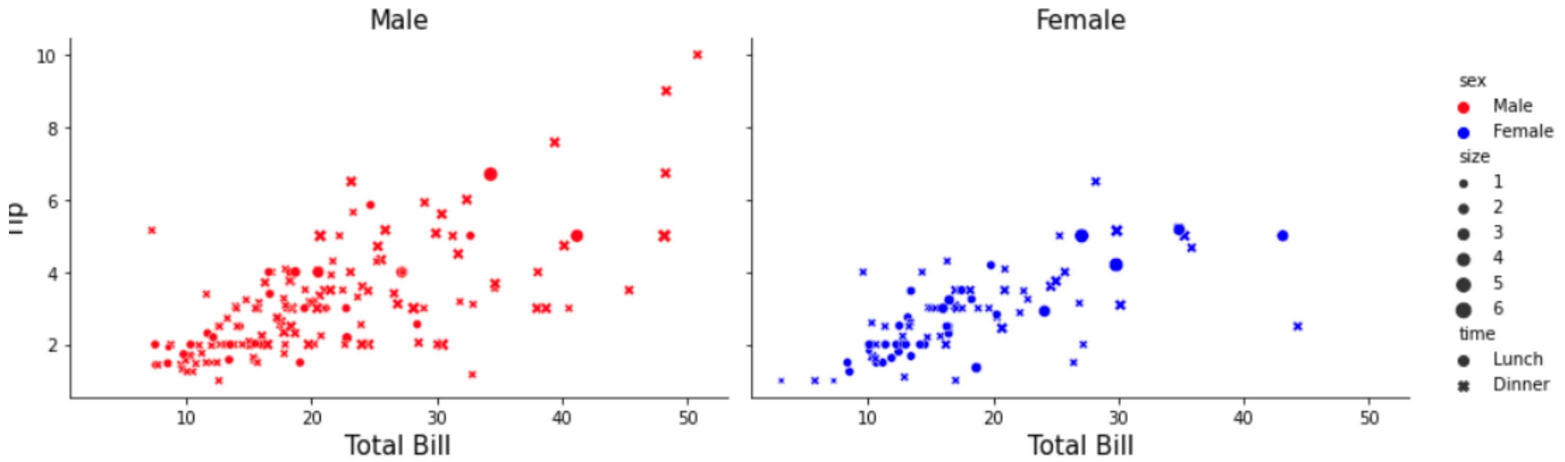
Day 4 – Advanced visualizations using seaborn

SICSS 2022 – Haifa University

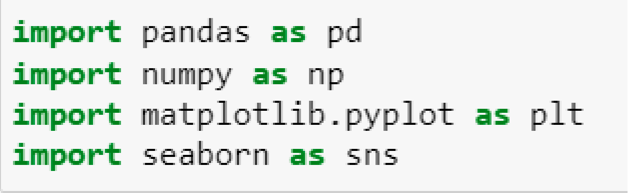
Amit Donner

Seaborn module

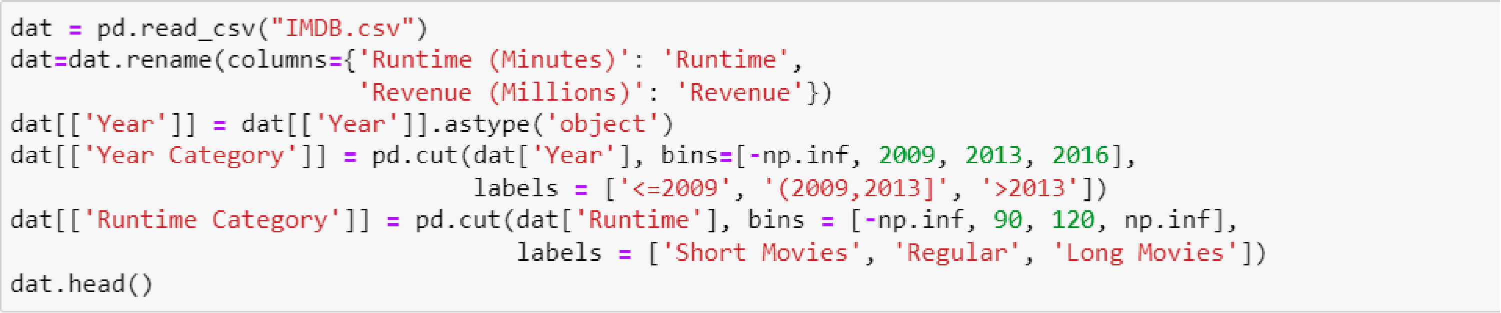
- Very strong tool for data visualization.
- Based on matplotlib and integrates perfectly with pandas.
- Makes complex plots using very short code.
- The perfect choice for statistical graphics.



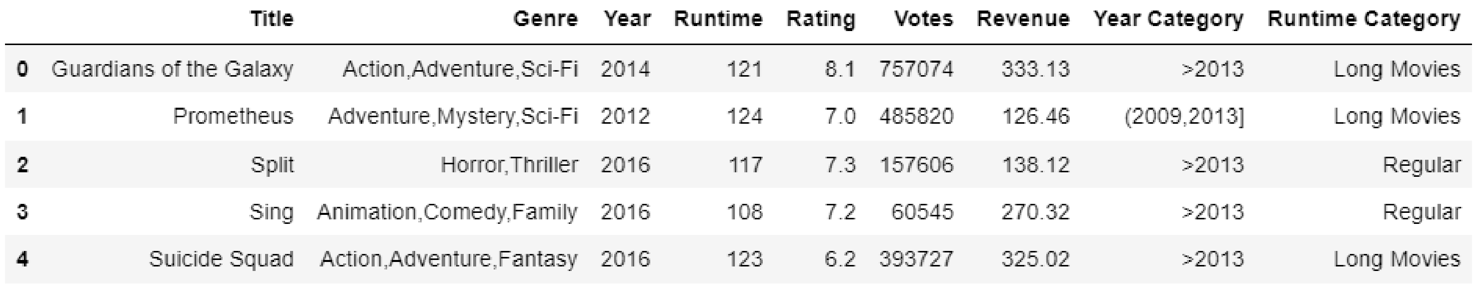
- First, recall the IMDB data from last meeting.
- We will use it once again, hence we need the pandas module.
- We will load matplotlib as well.
- And numpy (just in case….)



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```



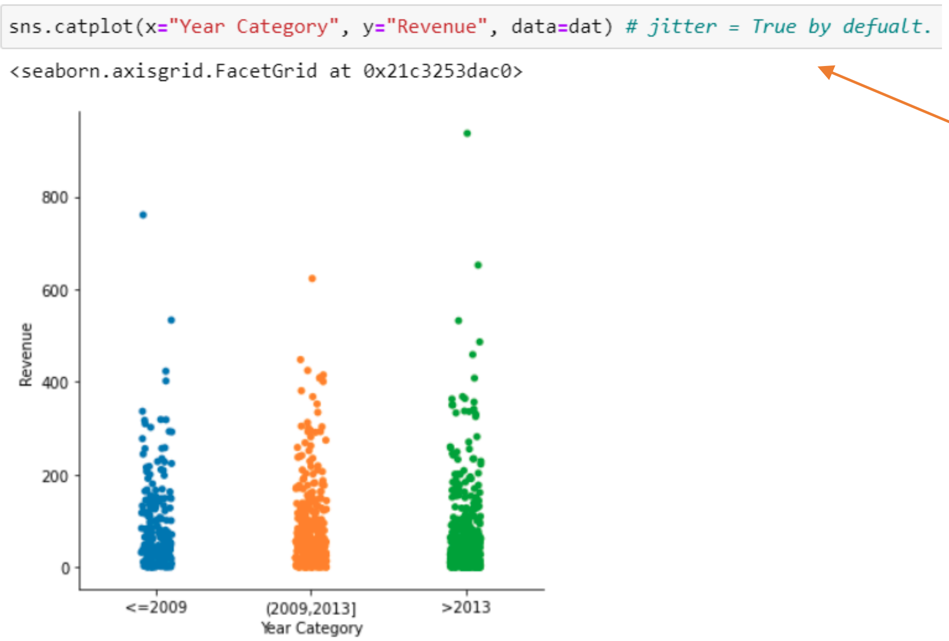
```
dat = pd.read_csv("IMDB.csv")
dat=dat.rename(columns={'Runtime (Minutes)': 'Runtime',
                        'Revenue (Millions)': 'Revenue'})
dat[['Year']] = dat[['Year']].astype('object')
dat[['Year Category']] = pd.cut(dat['Year'], bins=[-np.inf, 2009, 2013, 2016],
                                labels = ['<=2009', '(2009,2013]', '>2013'])
dat[['Runtime Category']] = pd.cut(dat['Runtime'], bins = [-np.inf, 90, 120, np.inf],
                                   labels = ['Short Movies', 'Regular', 'Long Movies'])
dat.head()
```



	Title	Genre	Year	Runtime	Rating	Votes	Revenue	Year Category	Runtime Category
0	Guardians of the Galaxy	Action,Adventure,Sci-Fi	2014	121	8.1	757074	333.13	>2013	Long Movies
1	Prometheus	Adventure,Mystery,Sci-Fi	2012	124	7.0	485820	126.46	(2009,2013]	Long Movies
2	Split	Horror,Thriller	2016	117	7.3	157606	138.12	>2013	Regular
3	Sing	Animation,Comedy,Family	2016	108	7.2	60545	270.32	>2013	Regular
4	Suicide Squad	Action,Adventure,Fantasy	2016	123	6.2	393727	325.02	>2013	Long Movies

Catplot (categorical plot)

- Let’s start with a very simple example – Revenue Vs. Year category.



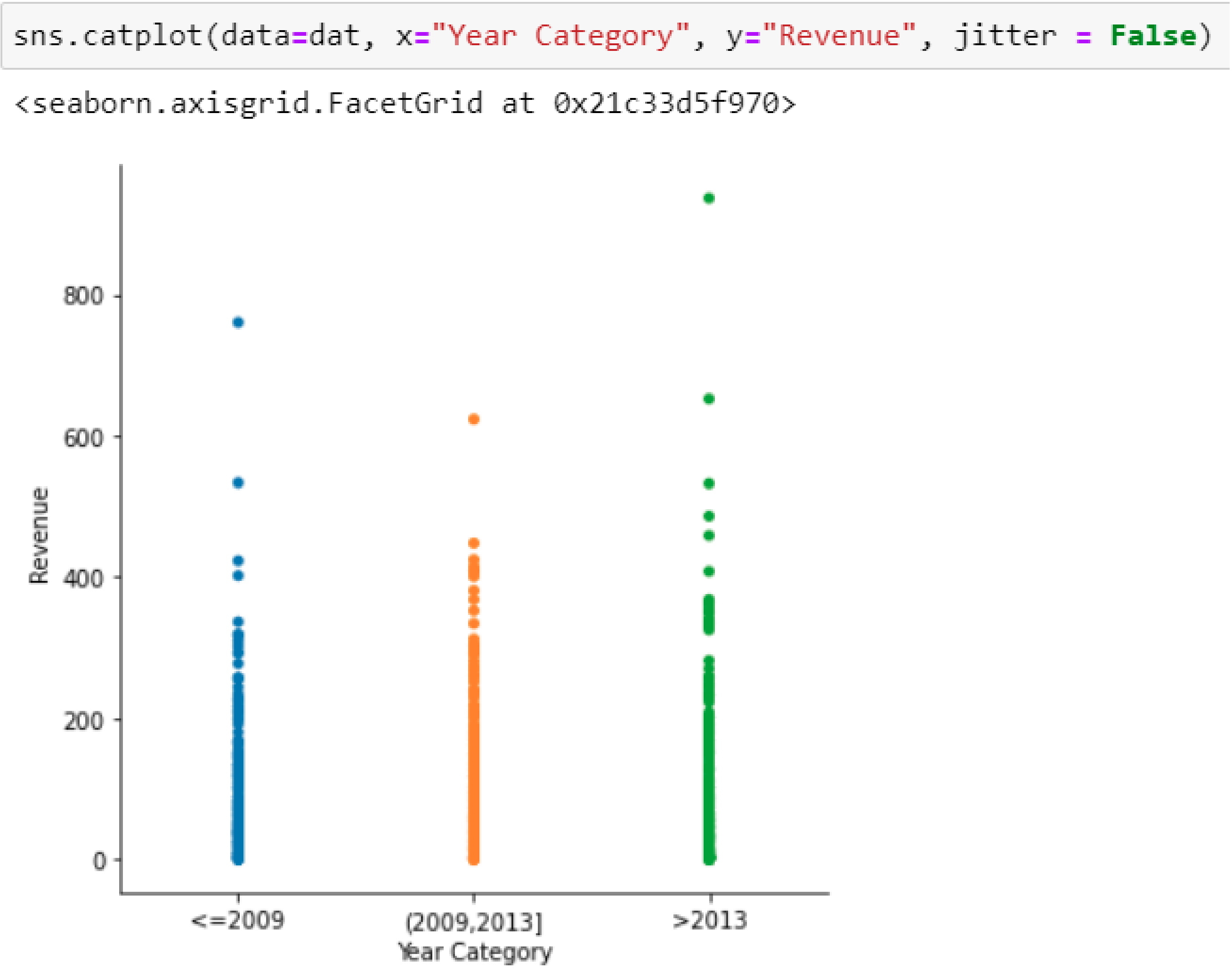
```
sns.catplot(x="Year Category", y="Revenue", data=dat) # jitter = True by defualt.
```

```
<seaborn.axisgrid.FacetGrid at 0x21c3253dac0>
```

Jitter?

- Observe that we pass the data we are using as an argument and then we specify the *x* and *y* axes just by variable names.
- By default, *catplot* creates *strip plots.*

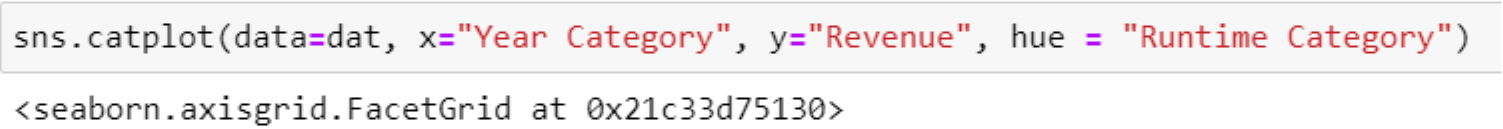
- With jitter = False we get,



```
sns.catplot(data=dat, x="Year Category", y="Revenue", jitter = False)
```

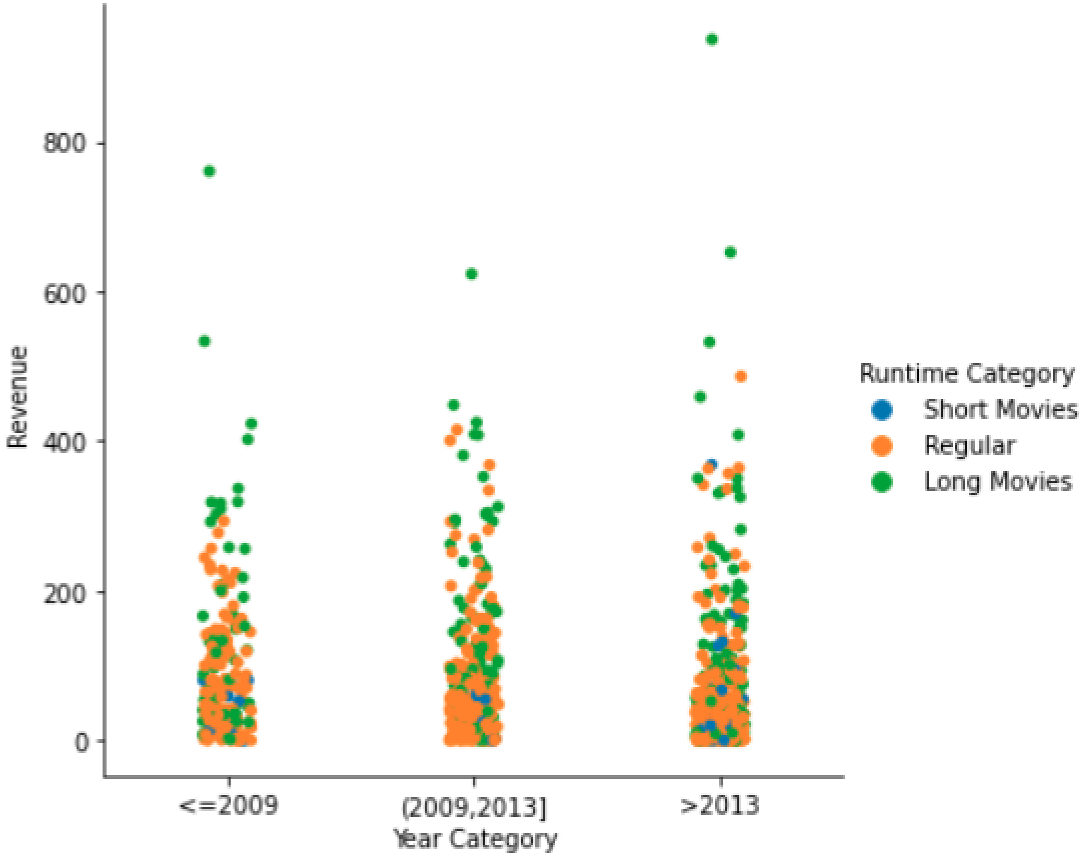
```
<seaborn.axisgrid.FacetGrid at 0x21c33d5f970>
```

- We can add a grouping variable which gives different colors to each group, using the *hue* argument:

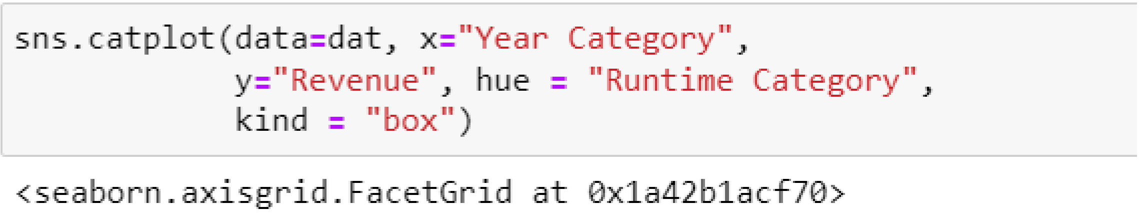


```
sns.catplot(data=dat, x="Year Category", y="Revenue", hue = "Runtime Category")
```

```
<seaborn.axisgrid.FacetGrid at 0x21c33d75130>
```

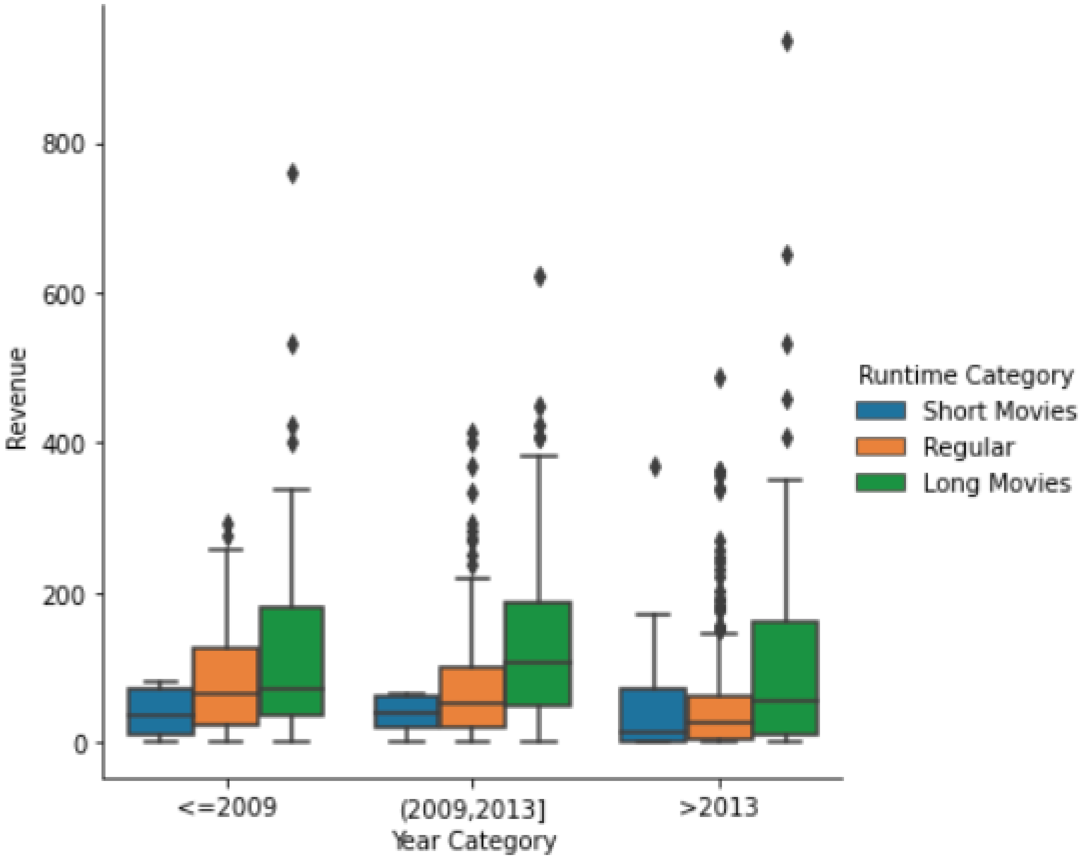


- A possibly better option is the *boxplot*, using kind = “box”.

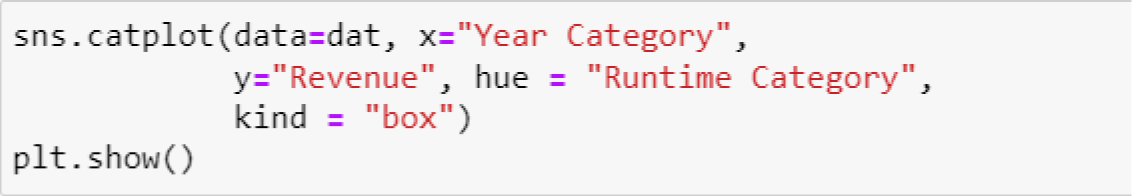


```
sns.catplot(data=dat, x="Year Category",
            y="Revenue", hue = "Runtime Category",
            kind = "box")
```

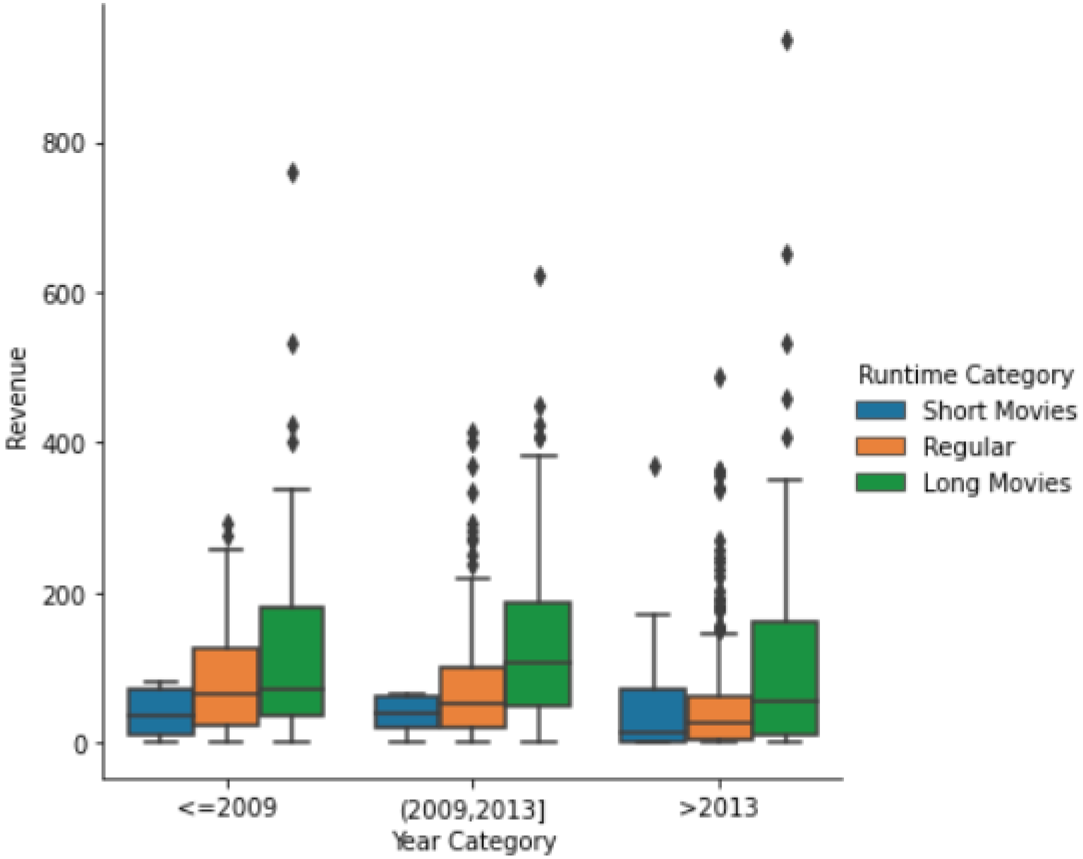
```
<seaborn.axisgrid.FacetGrid at 0x1a42b1acf70>
```



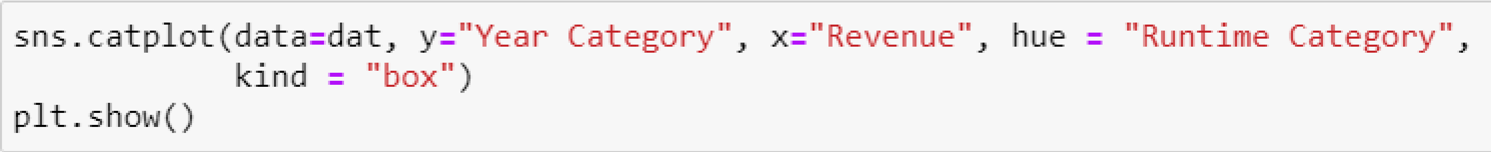
- We can suppress the message in the output by adding *plt.show()*



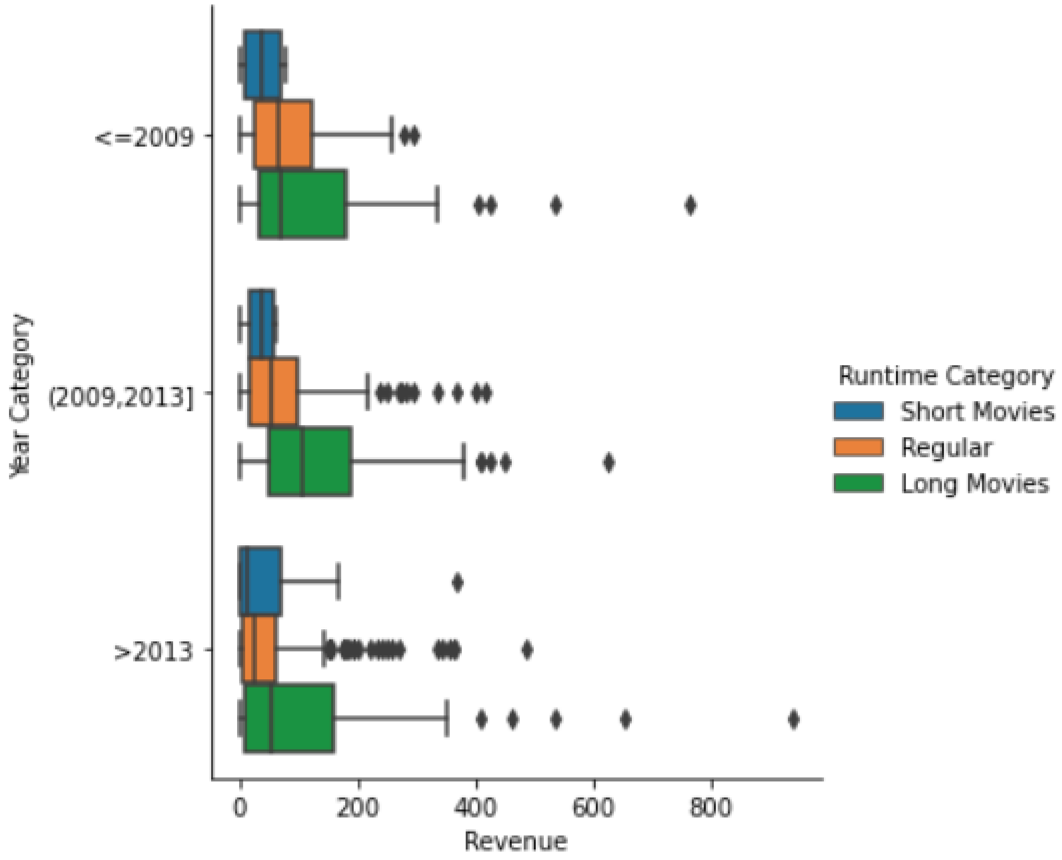
```
sns.catplot(data=dat, x="Year Category",
            y="Revenue", hue = "Runtime Category",
            kind = "box")
plt.show()
```



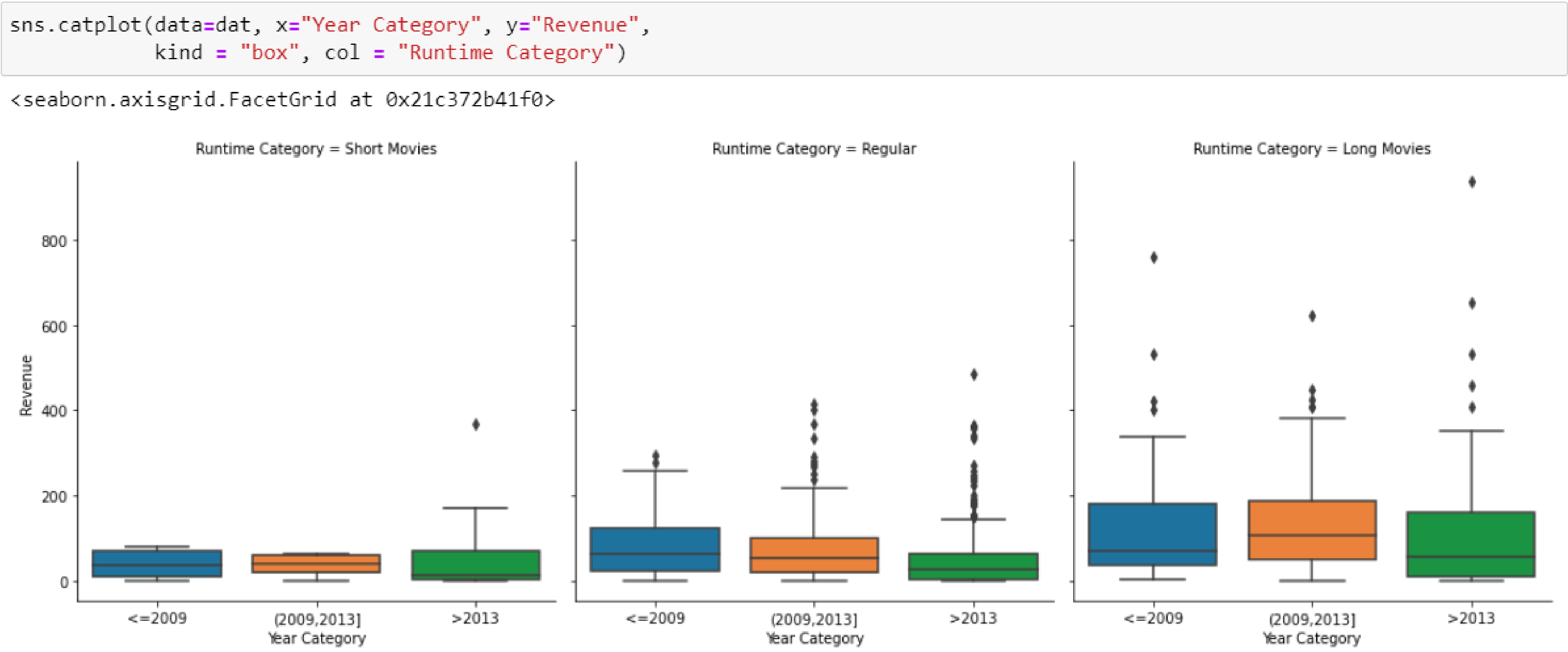
- If we switch the *x* and *y* arguments, we get



```
sns.catplot(data=dat, y="Year Category", x="Revenue", hue = "Runtime Category",
            kind = "box")
plt.show()
```



- Sometimes, it’s more useful to split the plot into categories instead of just ‘collapsing’ them into one. By passing the *col* (stands for column) argument, we can split the plot according a grouping variable.

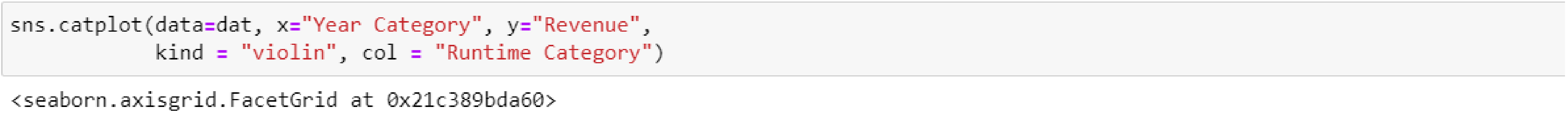


```
sns.catplot(data=dat, x="Year Category", y="Revenue",
            kind = "box", col = "Runtime Category")
```

```
<seaborn.axisgrid.FacetGrid at 0x21c372b41f0>
```

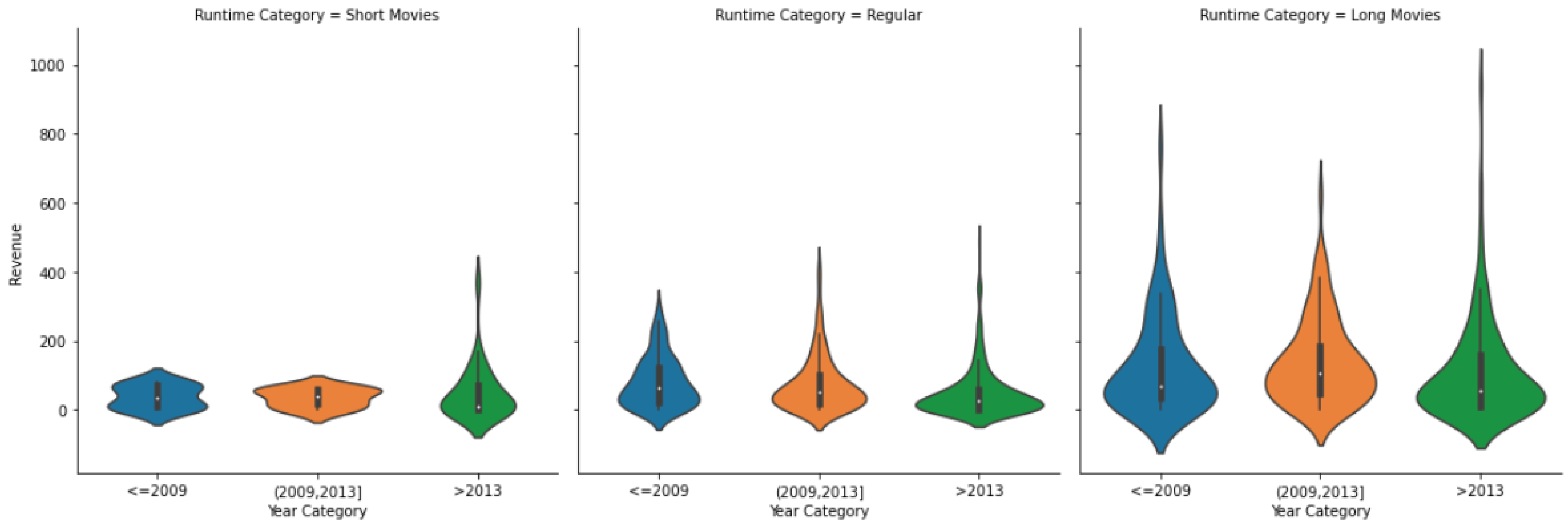
Runtime Category = Short Movies
Runtime Category = Regular
Runtime Category = Long Movies
Revenue
800
600
400
200
0
<=2009
(2009,2013]
>2013
Year Category

- Another option is the *violin plot* which also visualizes the ‘concentration’ of the data.

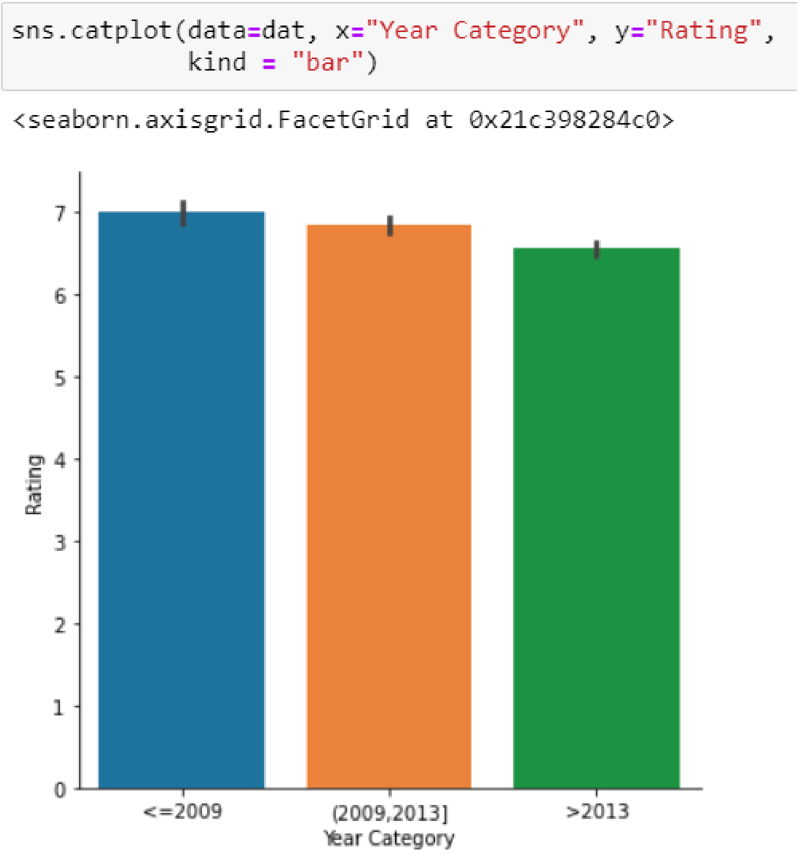


```
sns.catplot(data=dat, x="Year Category", y="Revenue",
            kind = "violin", col = "Runtime Category")
```

```
<seaborn.axisgrid.FacetGrid at 0x21c389bda60>
```

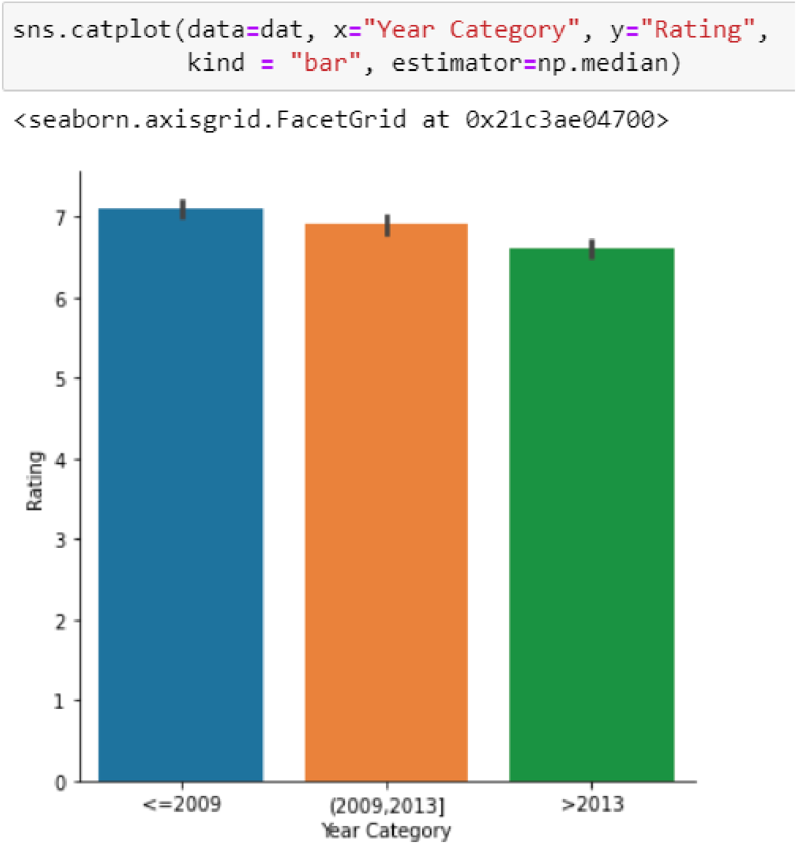


- By passing *kind = “bar”* we get as heights of the mean of *y* within each level of *x*. By default, the error bars are bootstrap CI’s.
- By passing *estimator = np.median* we can see the median within each group (we can also use np.std, np.var, np.sem and others).



```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar")
```

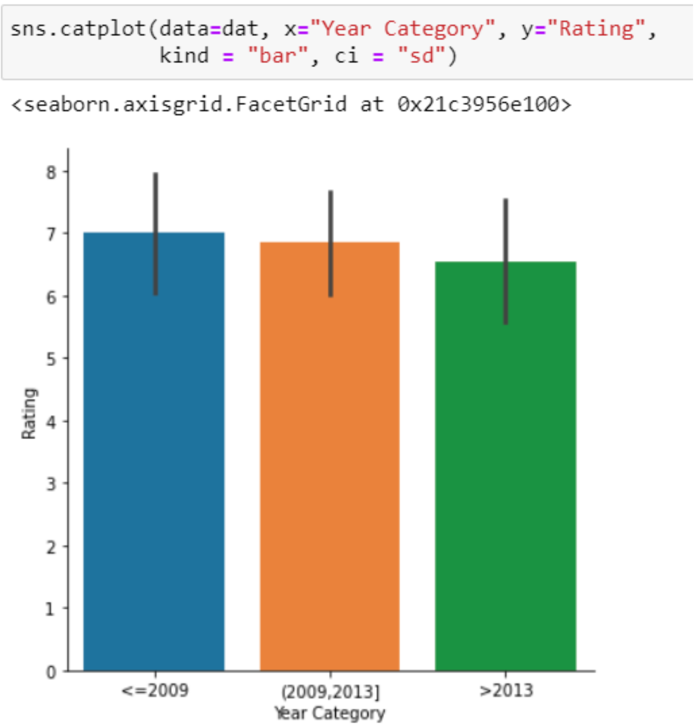
```
<seaborn.axisgrid.FacetGrid at 0x21c398284c0>
```



```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar", estimator=np.median)
```

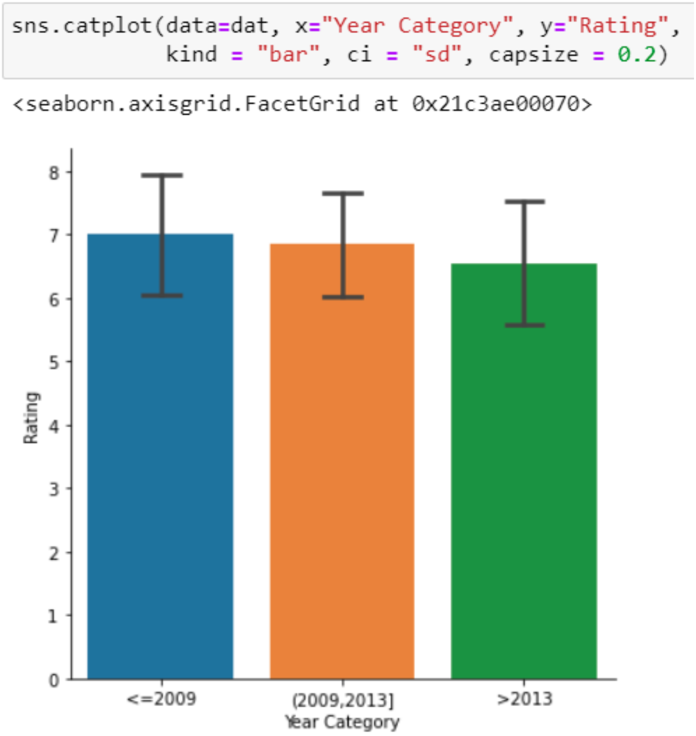
```
<seaborn.axisgrid.FacetGrid at 0x21c3ae04700>
```

- By setting *ci = “sd”* the error bars are *average ± standard deviation.*
- We can add *caps* using the argument *capsize*.
- For no error bars, pass *ci = None.*



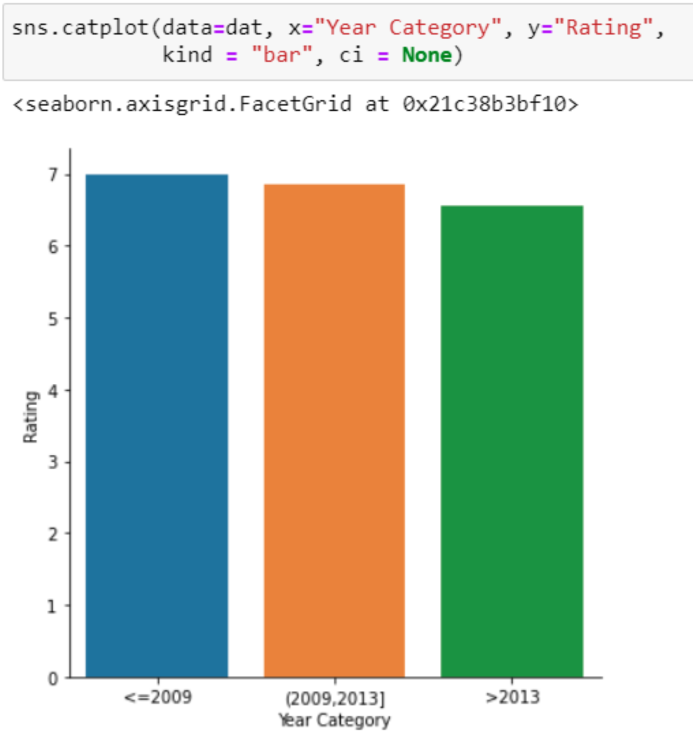
```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar", ci = "sd")
```

```
<seaborn.axisgrid.FacetGrid at 0x21c3956e100>
```



```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar", ci = "sd", capsize = 0.2)
```

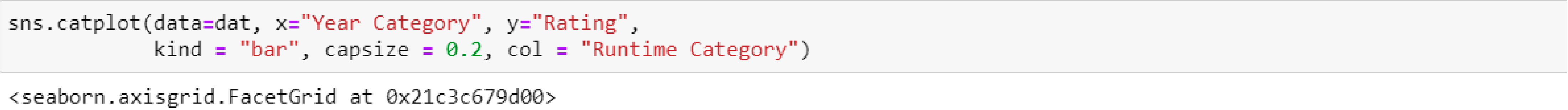
```
<seaborn.axisgrid.FacetGrid at 0x21c3ae00070>
```



```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar", ci = None)
```

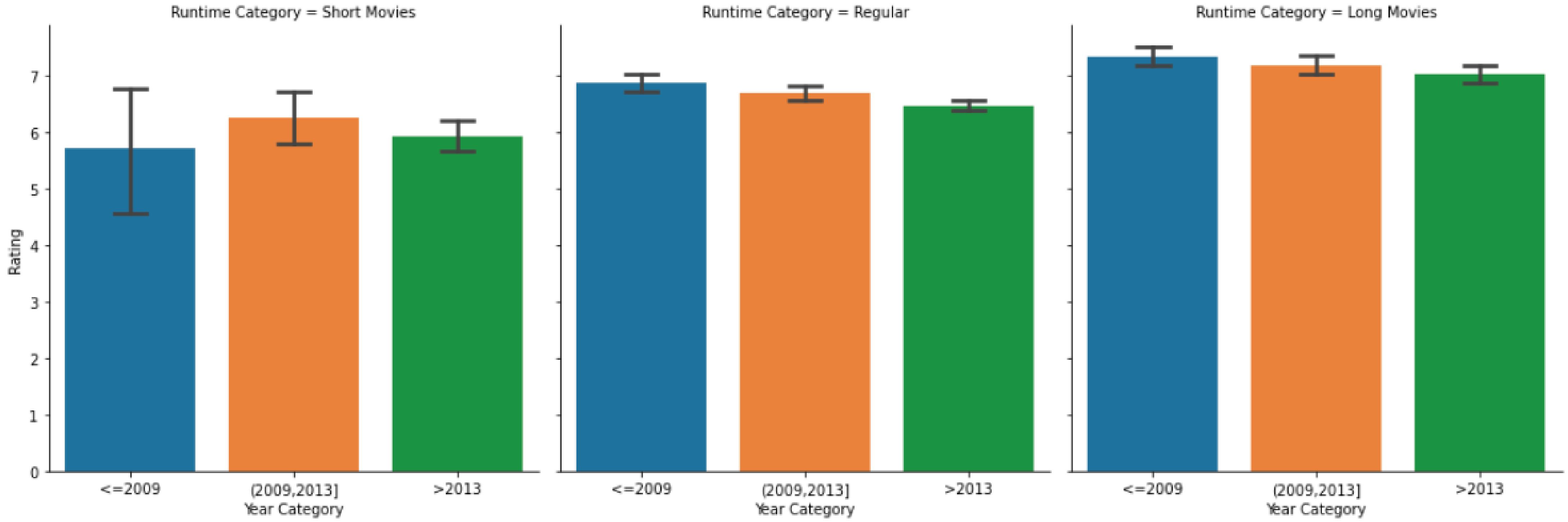
```
<seaborn.axisgrid.FacetGrid at 0x21c38b3bf10>
```

- Faceting is possible here as well. For example,

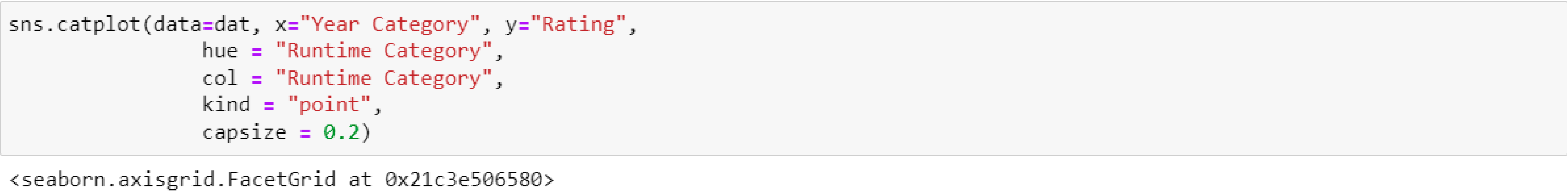


```
sns.catplot(data=dat, x="Year Category", y="Rating",
            kind = "bar", capsize = 0.2, col = "Runtime Category")
```

```
<seaborn.axisgrid.FacetGrid at 0x21c3c679d00>
```

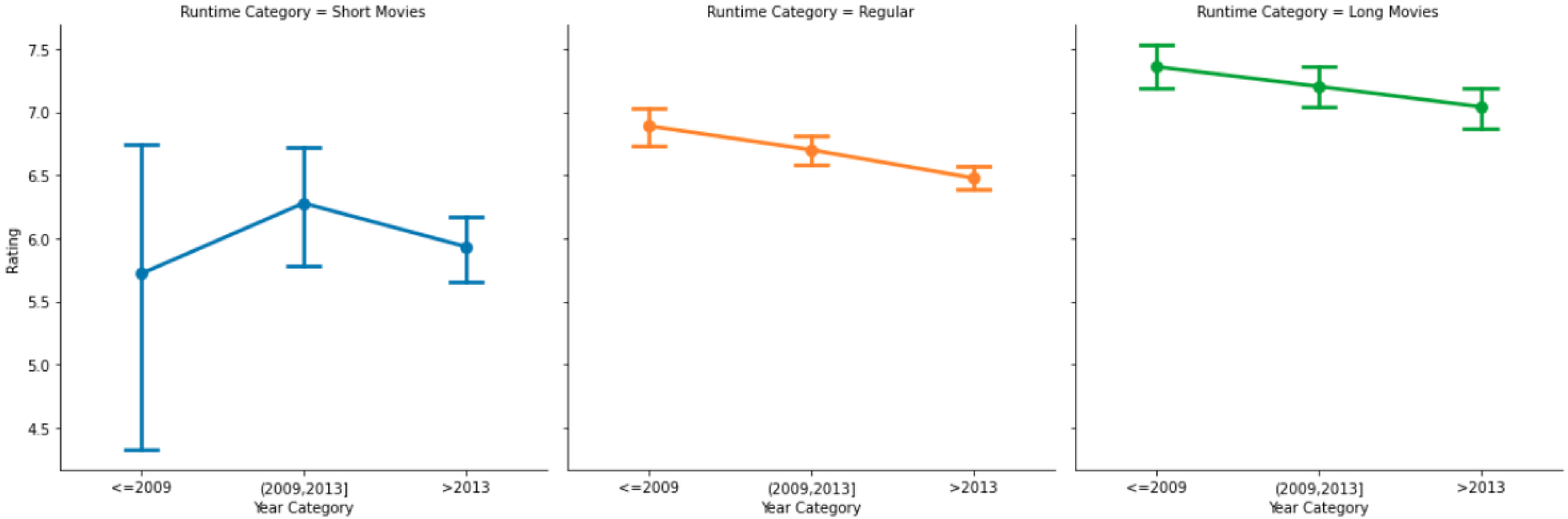


- Maybe a nicer option will be using kind = *“point”,* which yields:
- Note that we observe the bootstrap CI’s.

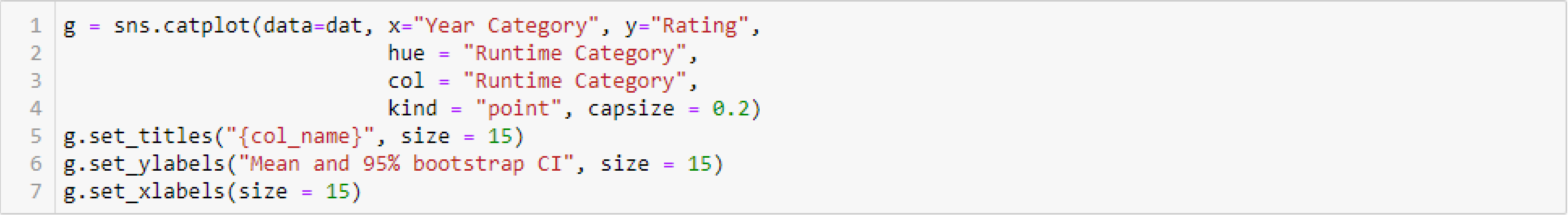


```
sns.catplot(data=dat, x="Year Category", y="Rating",
                hue = "Runtime Category",
                col = "Runtime Category",
                kind = "point",
                capsize = 0.2)
```

```
<seaborn.axisgrid.FacetGrid at 0x21c3e506580>
```

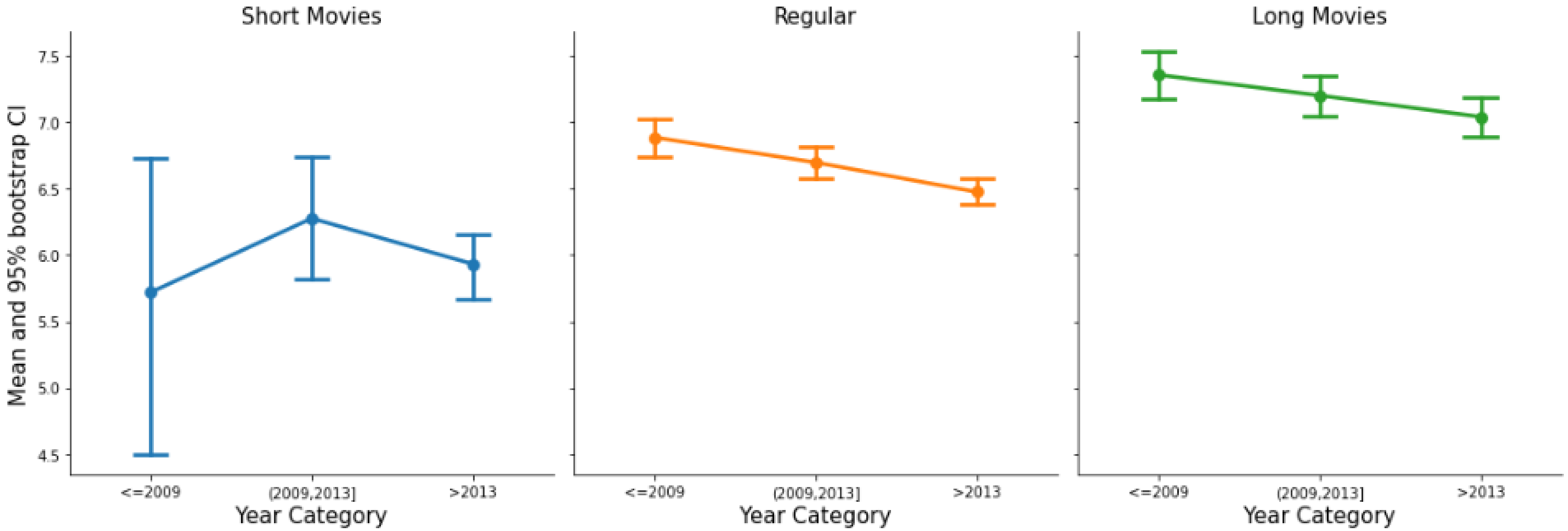


- Some improvements can be made by setting the titles to include only the levels without the name of the grouping variable.
- In addition, we can change the axis labels.

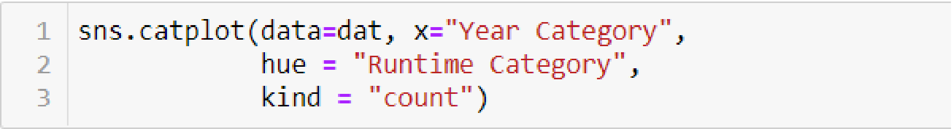


```
g = sns.catplot(data=dat, x="Year Category", y="Rating",
                          hue = "Runtime Category",
                          col = "Runtime Category",
                          kind = "point", capsize = 0.2)
g.set_titles("{col_name}", size = 15)
g.set_ylabels("Mean and 95% bootstrap CI", size = 15)
g.set_xlabels(size = 15)
```

```
<seaborn.axisgrid.FacetGrid at 0x1db904ef6d0>
```

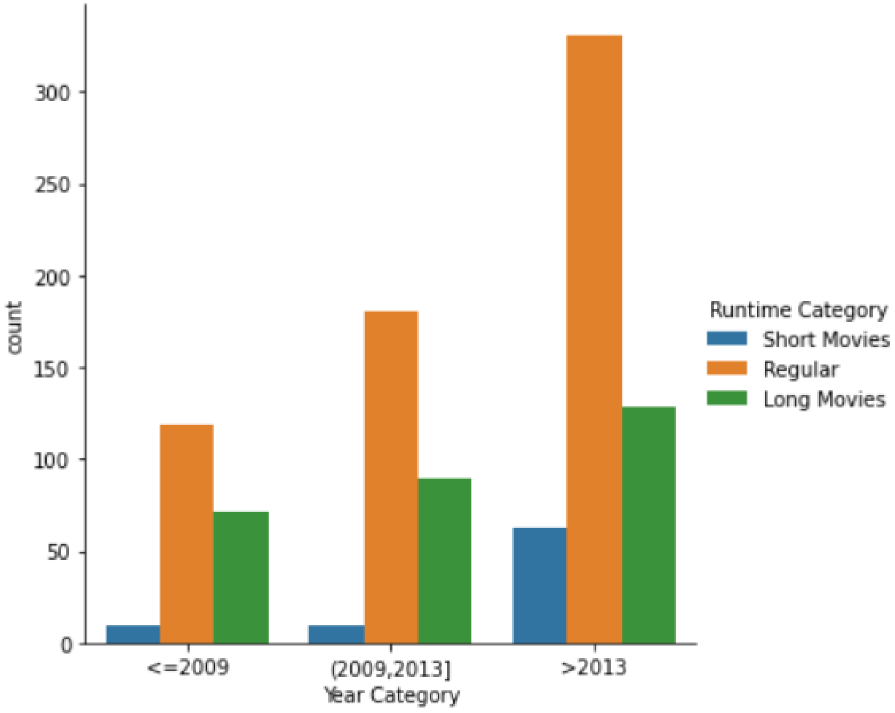


- When we have only categorical variables, we can use the argument *kind = “count”* to get a bar plot of frequencies.



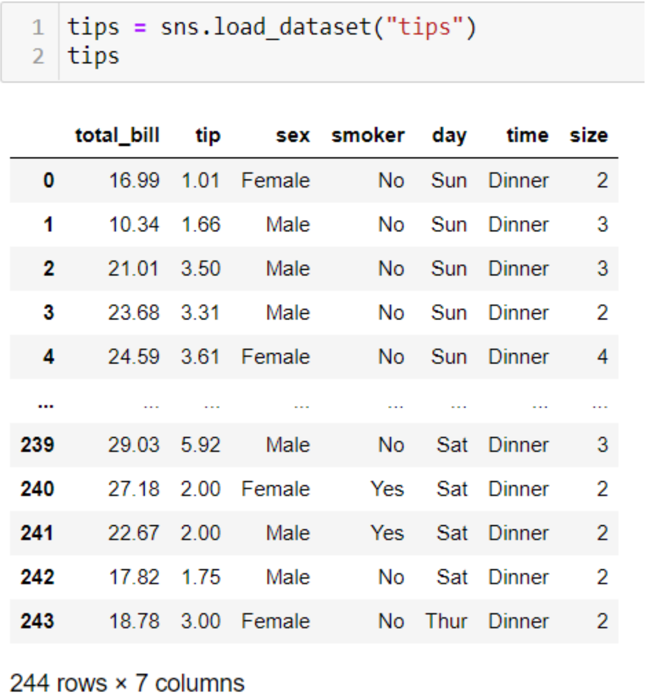
```python
sns.catplot(data=dat, x="Year Category",
            hue = "Runtime Category",
            kind = "count")
```

```
<seaborn.axisgrid.FacetGrid at 0x1db90a2fb50>
```



Relplot – relational plot

- The next ‘family’ of graphics, is the *relplot* which is mostly useful for *scatter* and *line* plots.
- For the following example I’ll use the *tips* data that comes with the *seaborn* module.

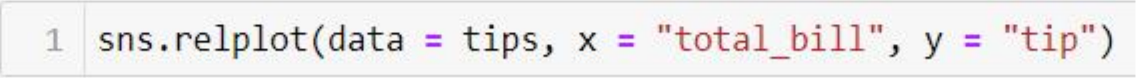


```
tips = sns.load_dataset("tips")
tips
```

	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
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

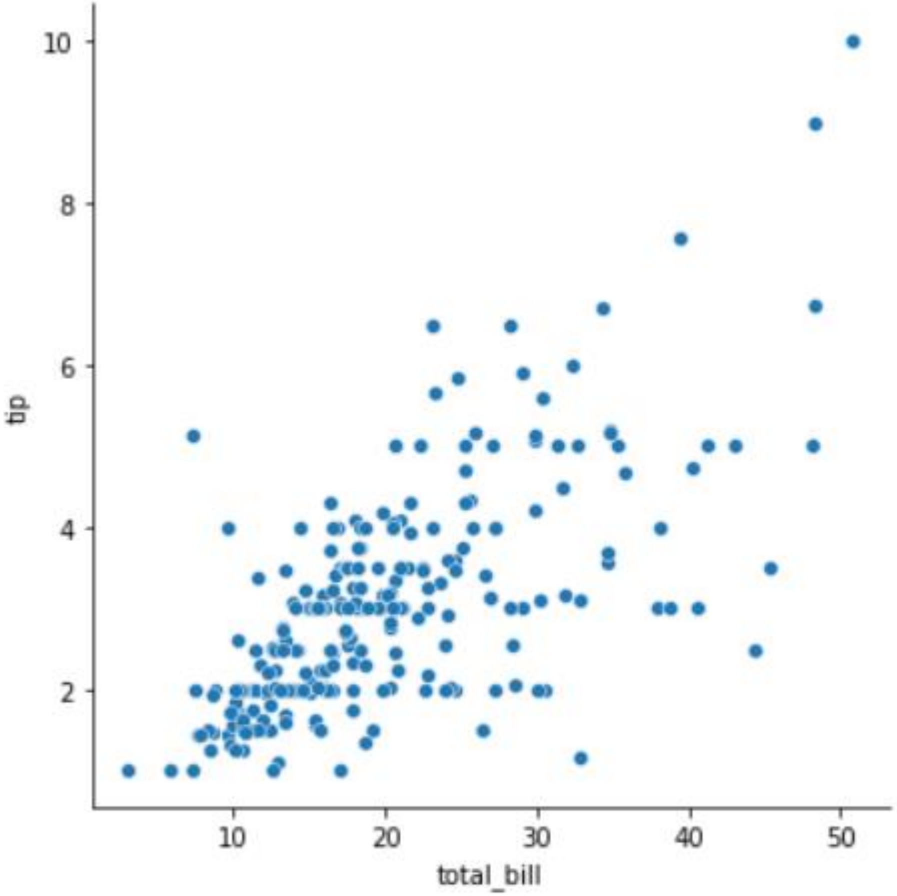
244 rows × 7 columns

- The most basic option is the *scatterplot* of *x* vs. *y.*

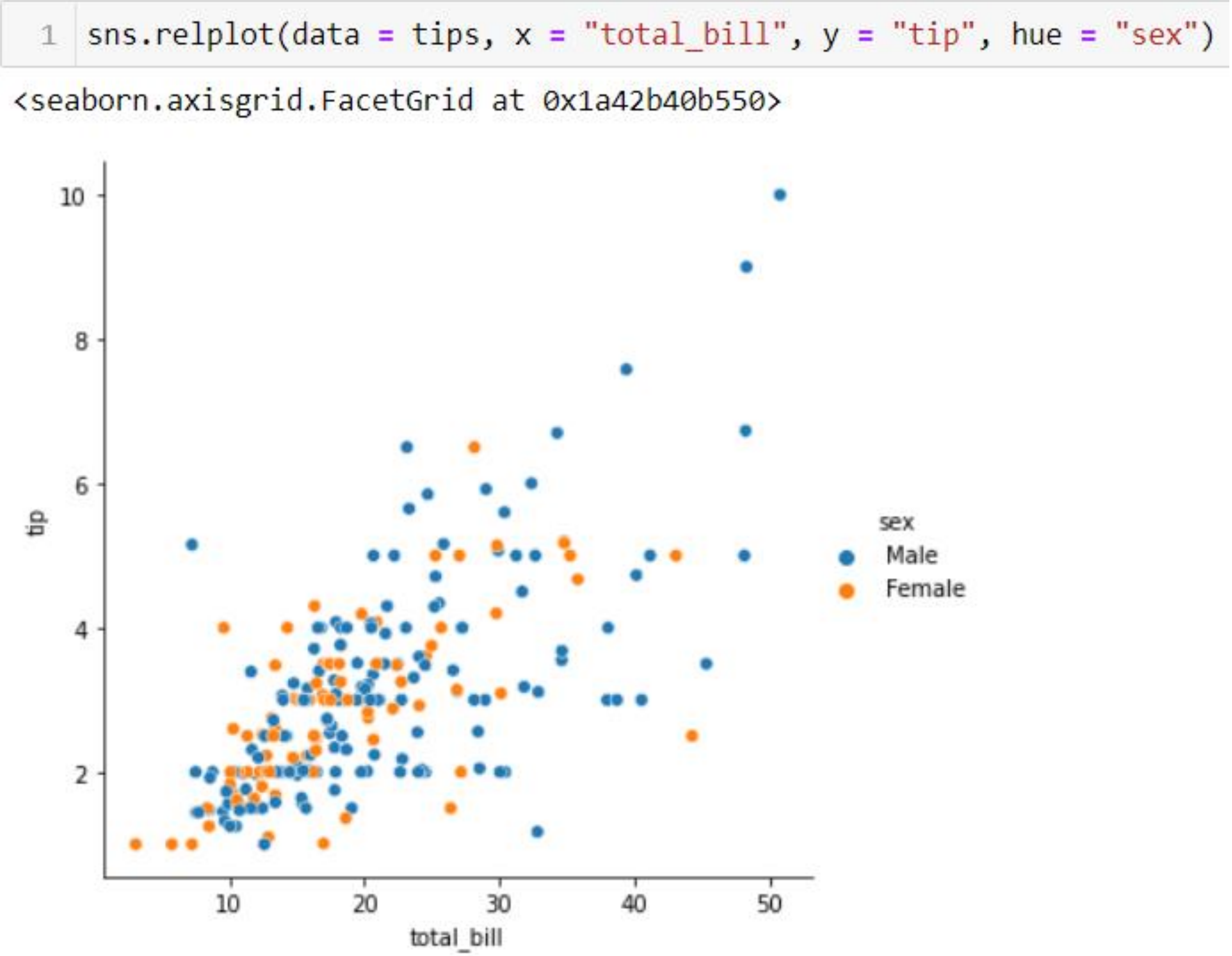


```
sns.relplot(data = tips, x = "total_bill", y = "tip")
```

```
<seaborn.axisgrid.FacetGrid at 0x1db90c011f0>
```



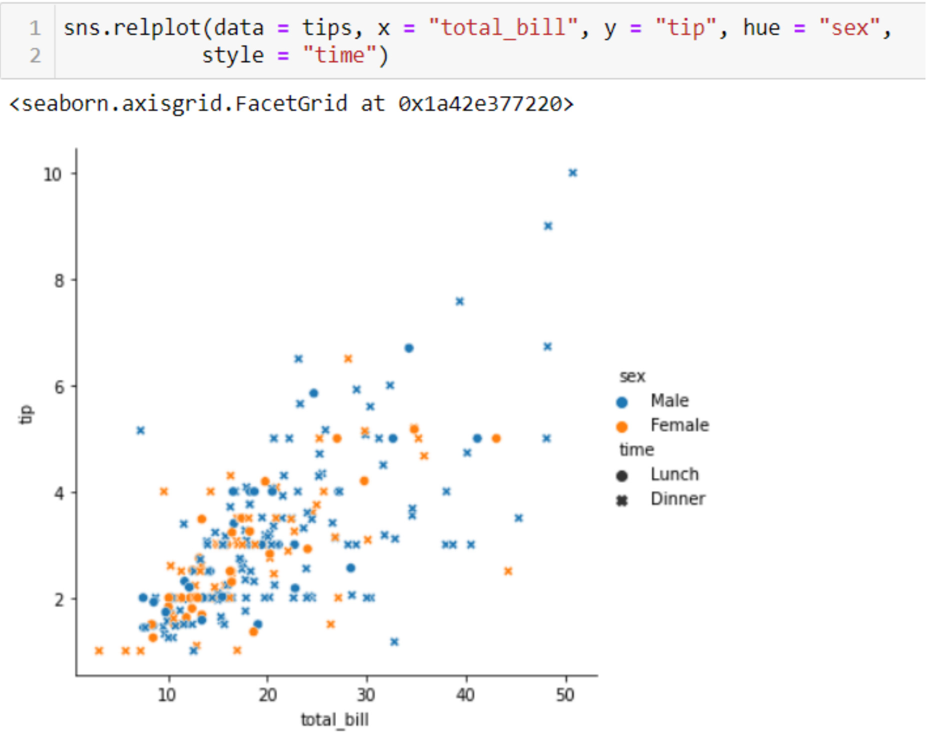
- Once again, we can change colors according to a grouping variable.



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

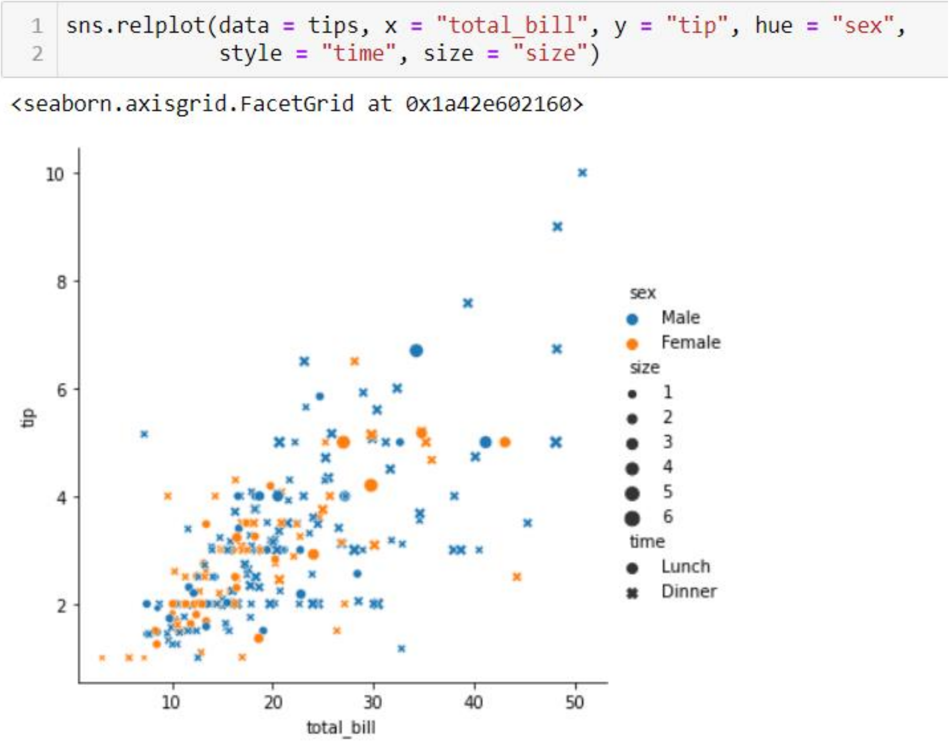
```
<seaborn.axisgrid.FacetGrid at 0x1a42b40b550>
```

- Additional parameters are the *style*, i.e. the shape of the dots and *size*, which can also be determined by grouping variables.



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

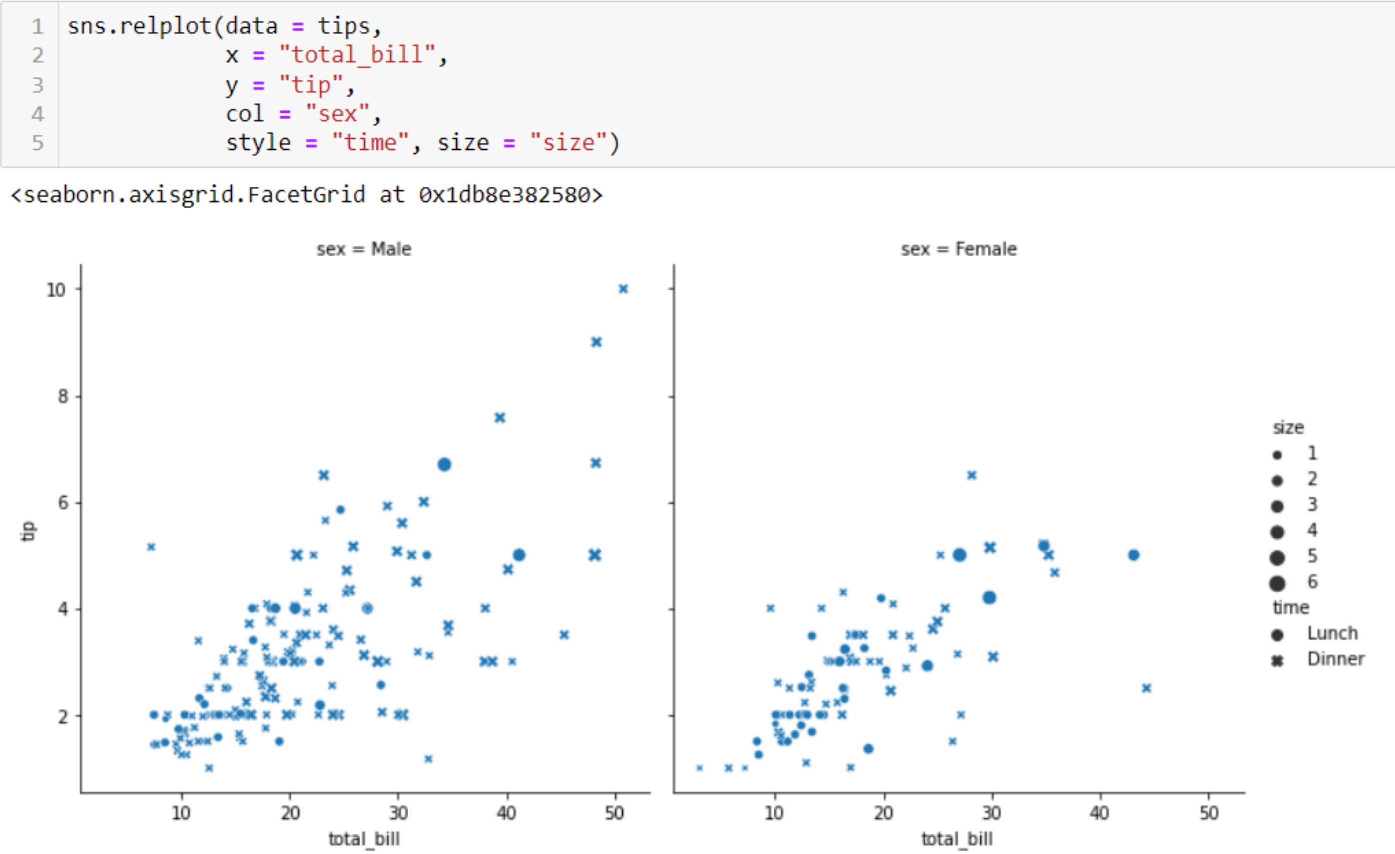
```
<seaborn.axisgrid.FacetGrid at 0x1a42e377220>
```



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

```
<seaborn.axisgrid.FacetGrid at 0x1a42e602160>
```

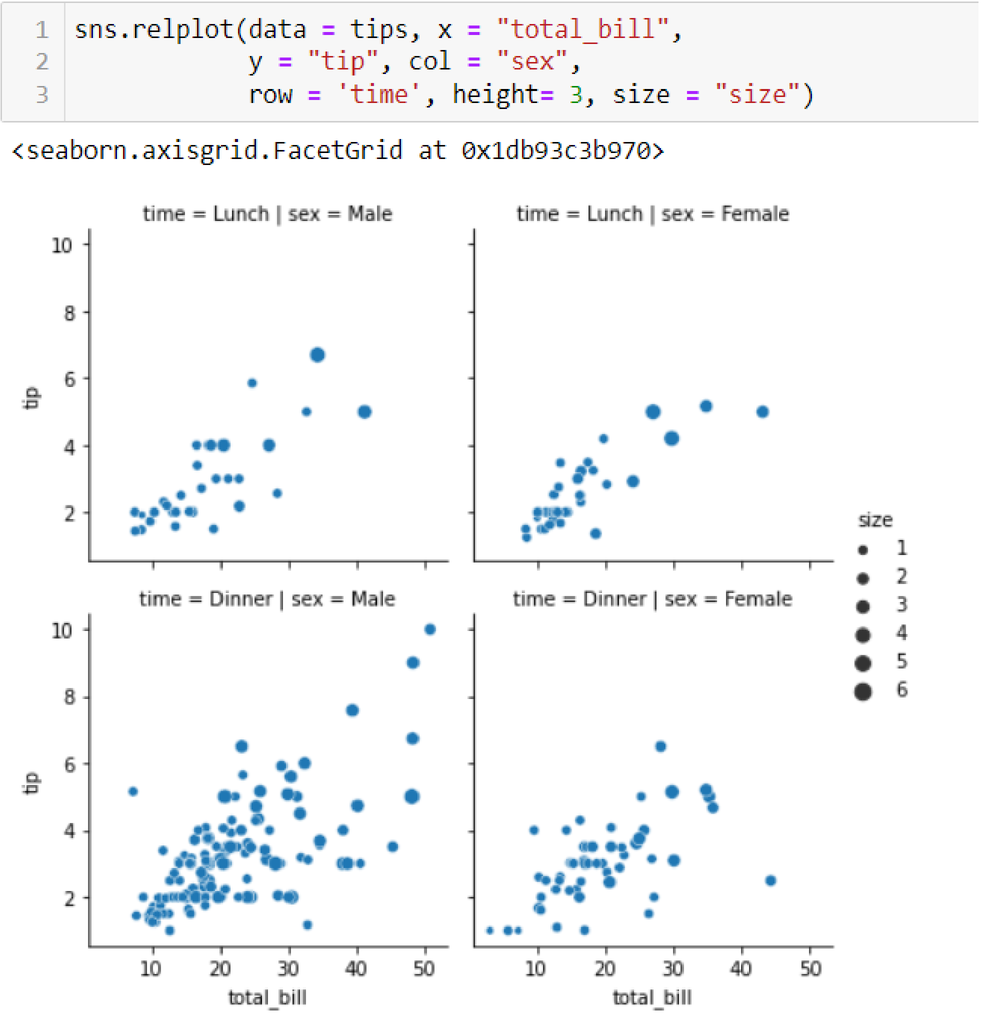
- Once again, we can use faceting with the *col* argument.



```
sns.relplot(data = tips,
            x = "total_bill",
            y = "tip",
            col = "sex",
            style = "time", size = "size")
```

```
<seaborn.axisgrid.FacetGrid at 0x1db8e382580>
```

- We can have 2D faceting if we use the arguments *col* and *row* together.

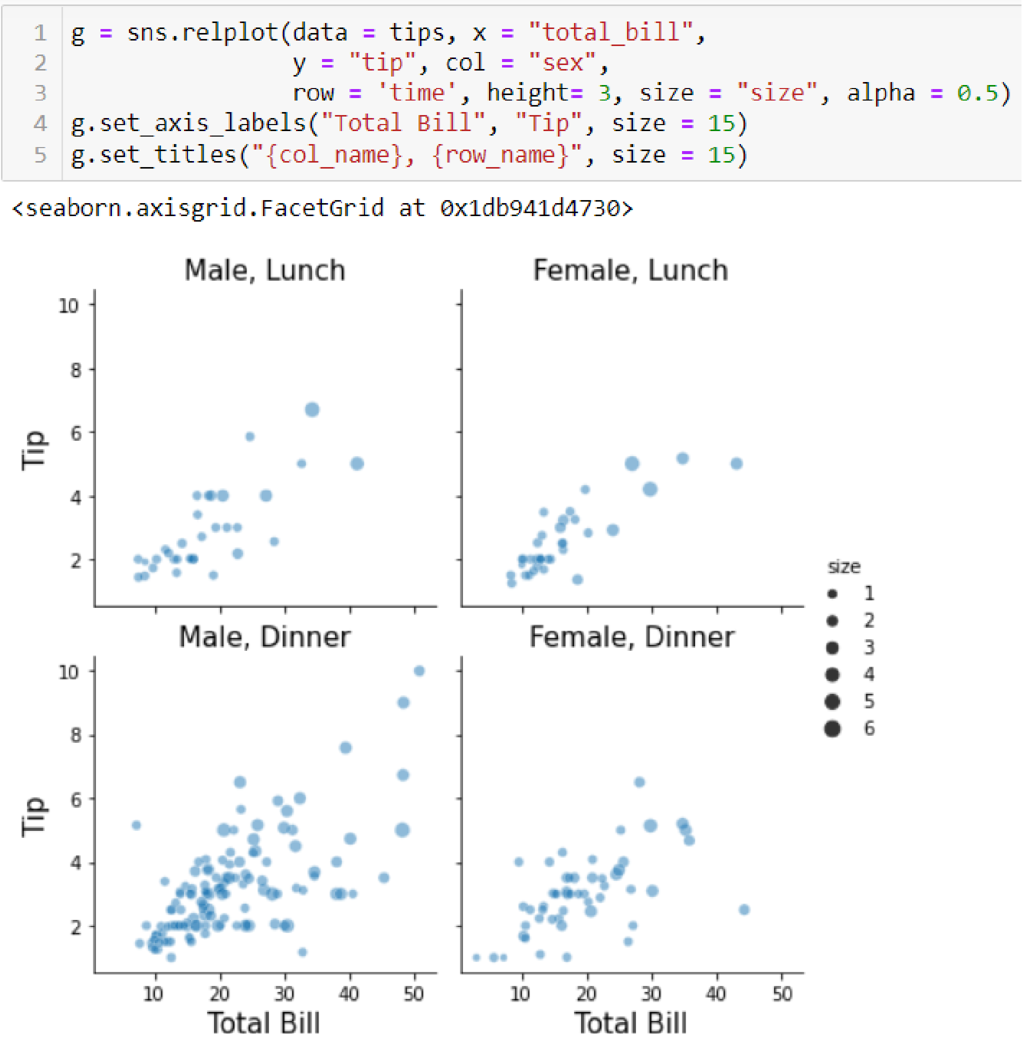


```
sns.relplot(data = tips, x = "total_bill",
            y = "tip", col = "sex",
            row = 'time', height= 3, size = "size")
```

```
<seaborn.axisgrid.FacetGrid at 0x1db93c3b970>
```

time = Lunch | sex = Male
time = Lunch | sex = Female
time = Dinner | sex = Male
time = Dinner | sex = Female
tip
tip
2
4
6
8
10
10
20
30
40
50
total_bill
total_bill
size
1
2
3
4
5
6

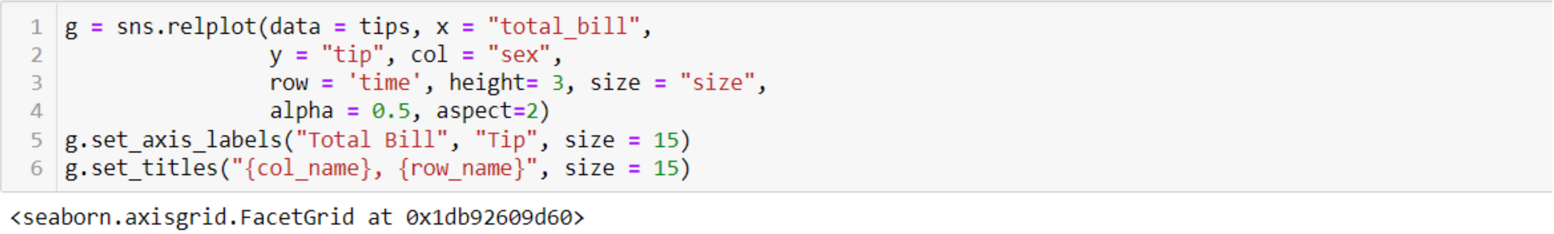
- We can control the transparency using the *alpha* argument and drop the grouping variable names from the facet’s titles.



```
g = sns.relplot(data = tips, x = "total_bill",
                y = "tip", col = "sex",
                row = 'time', height= 3, size = "size", alpha = 0.5)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

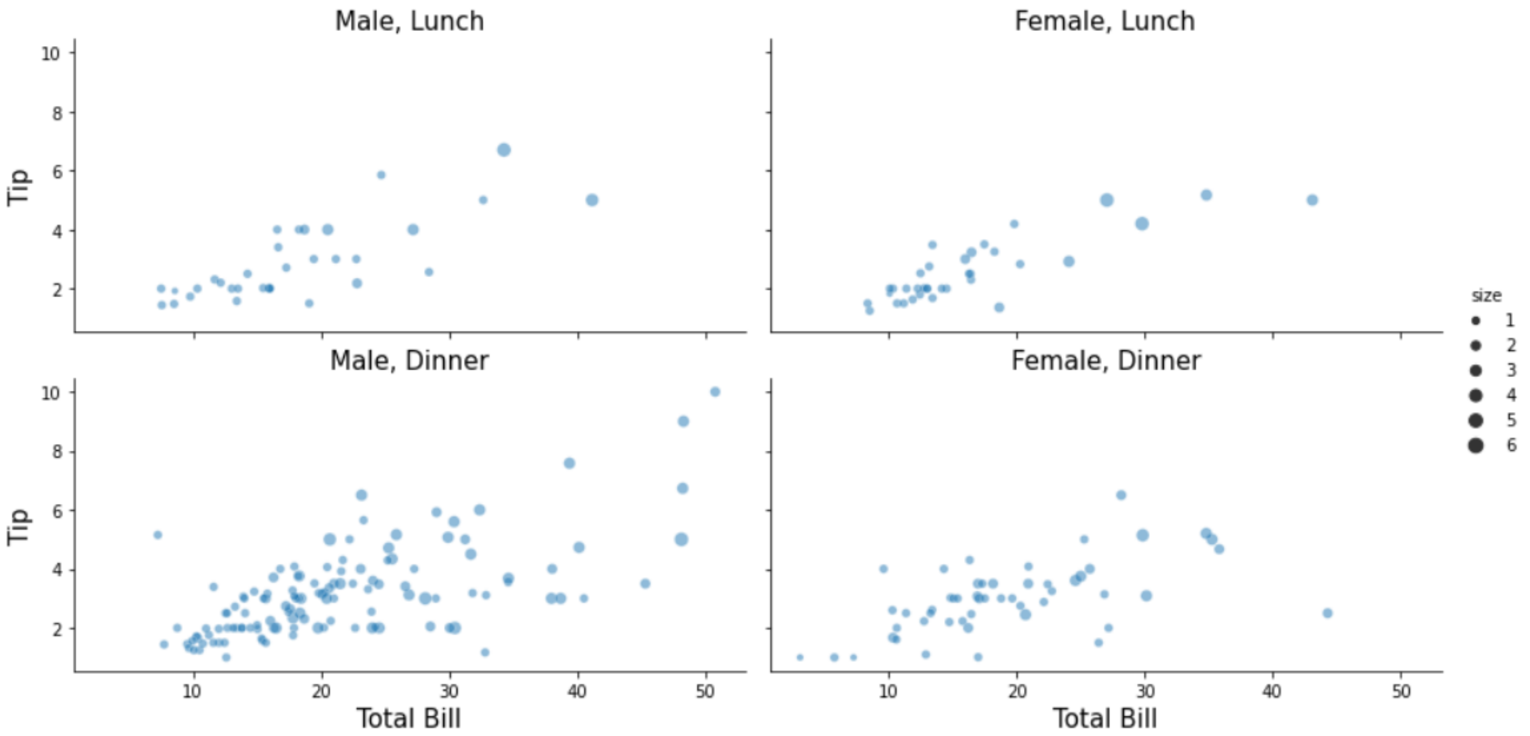
```
<seaborn.axisgrid.FacetGrid at 0x1db941d4730>
```

- An additional useful argument is the *aspect* which sets the ratio between the figures’ width and height.



```python
g = sns.relplot(data = tips, x = "total_bill",
                y = "tip", col = "sex",
                row = 'time', height= 3, size = "size",
                alpha = 0.5, aspect=2)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

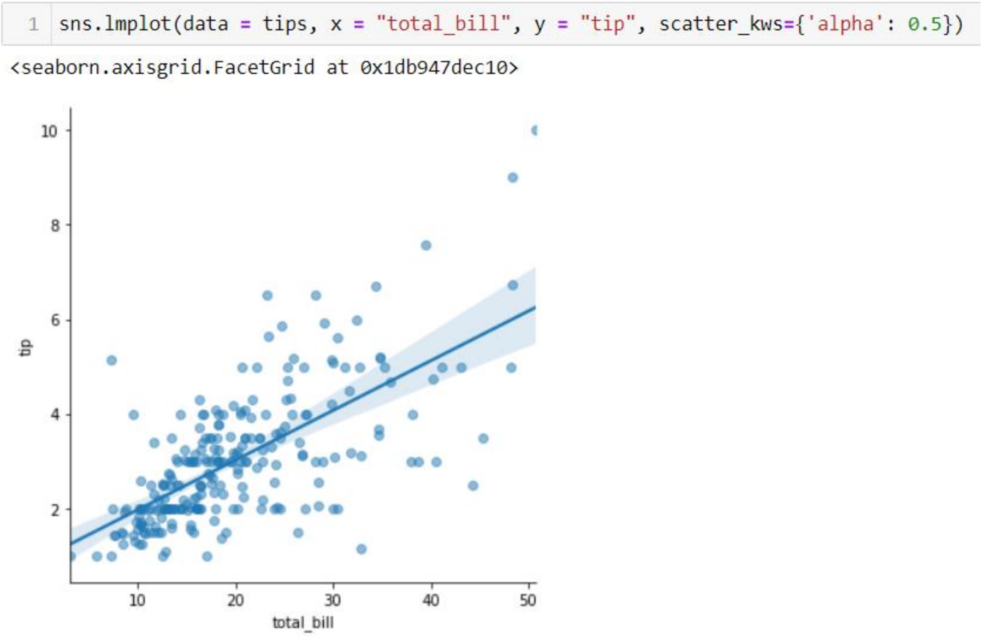
```
<seaborn.axisgrid.FacetGrid at 0x1db92609d60>
```



Lmplot – Linear models plot

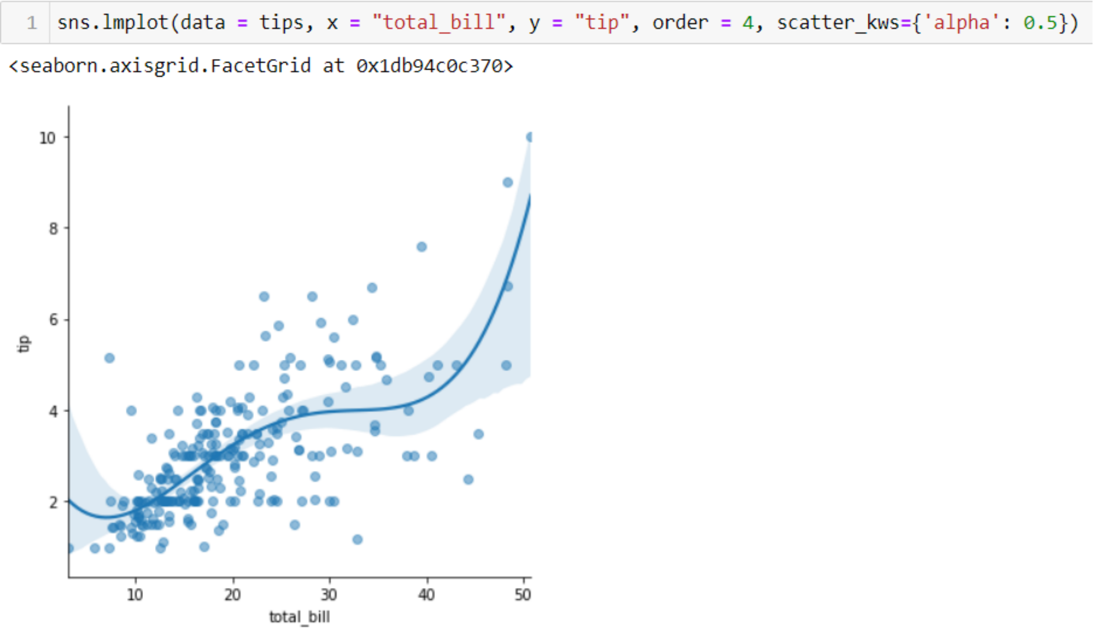
- One more family of plots is the lmplot which adds a trend line to the data points.
- Can be a linear/polynomial regression line or a robust regression line.
- Useful for interpreting and visualizing regression models.

- First example – Linear and polynomial regression



```
sns.lmplot(data = tips, x = "total_bill", y = "tip", scatter_kws={'alpha': 0.5})
```

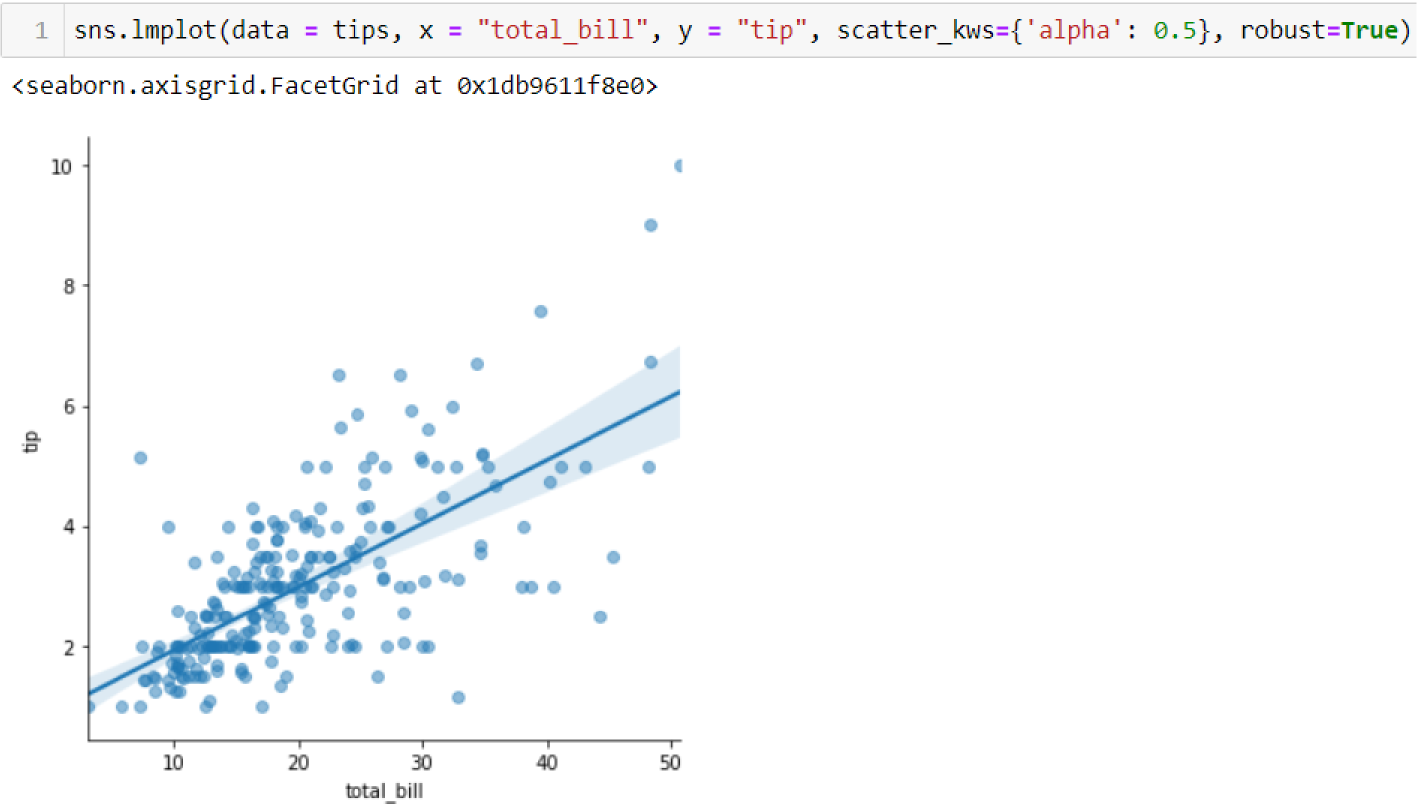
```
<seaborn.axisgrid.FacetGrid at 0x1db947dec10>
```



```
sns.lmplot(data = tips, x = "total_bill", y = "tip", order = 4, scatter_kws={'alpha': 0.5})
```

```
<seaborn.axisgrid.FacetGrid at 0x1db94c0c370>
```

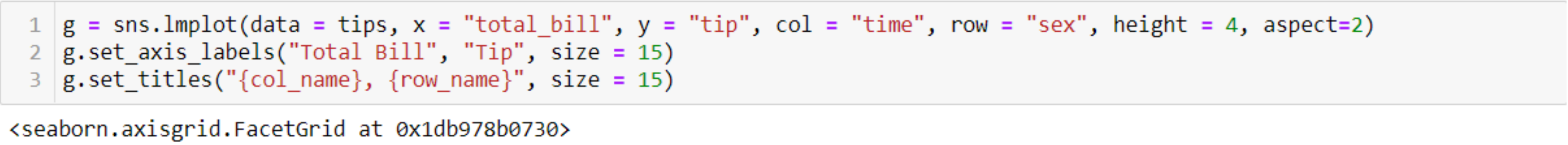
- Robust regression line (useful when we have extreme data points).
- Note that computation time is longer.



```python
sns.lmplot(data = tips, x = "total_bill", y = "tip", scatter_kws={'alpha': 0.5}, robust=True)
```

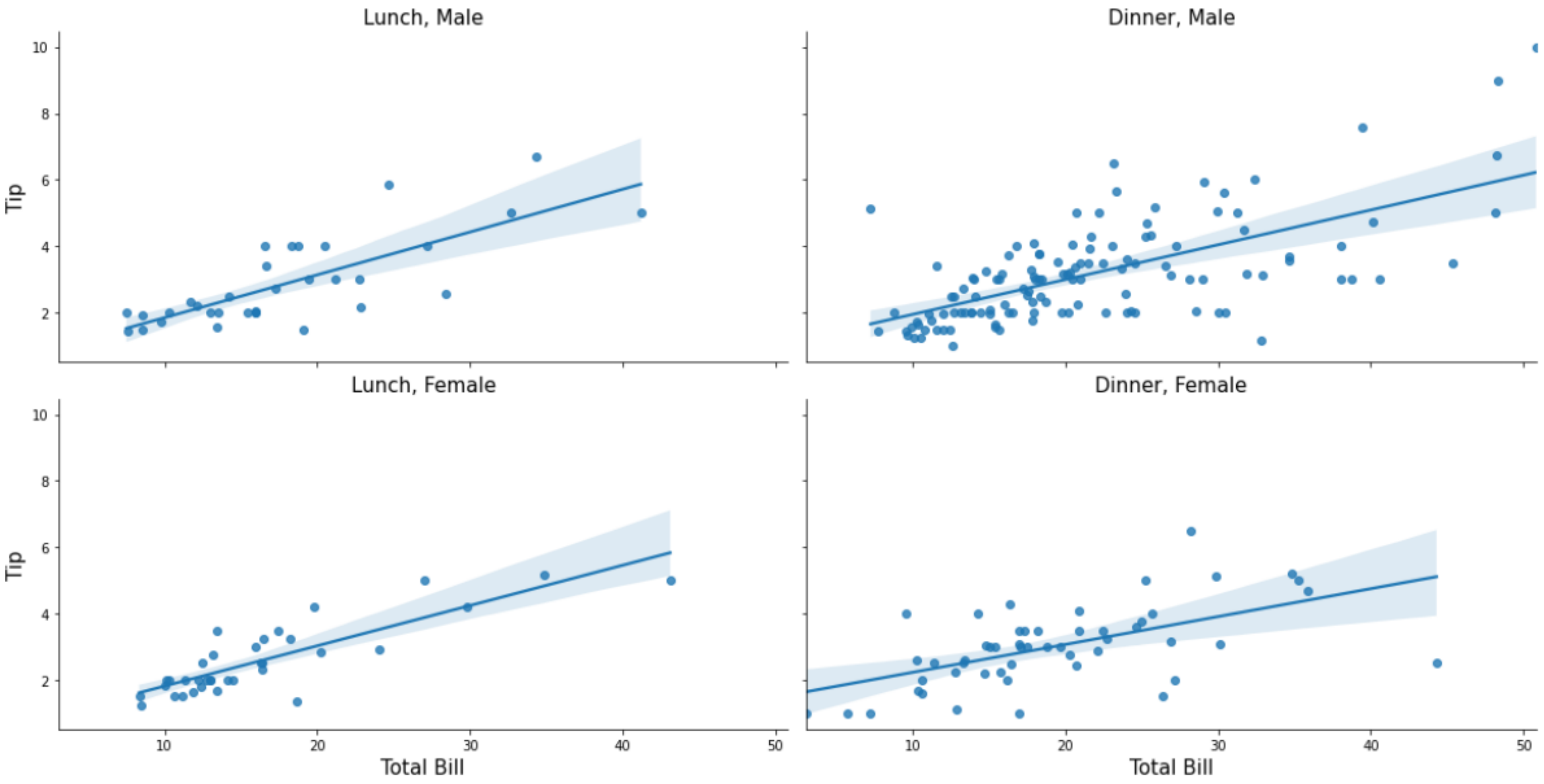
```
<seaborn.axisgrid.FacetGrid at 0x1db9611f8e0>
```

- Once again, we can use faceting and visualize a 2-way interaction.

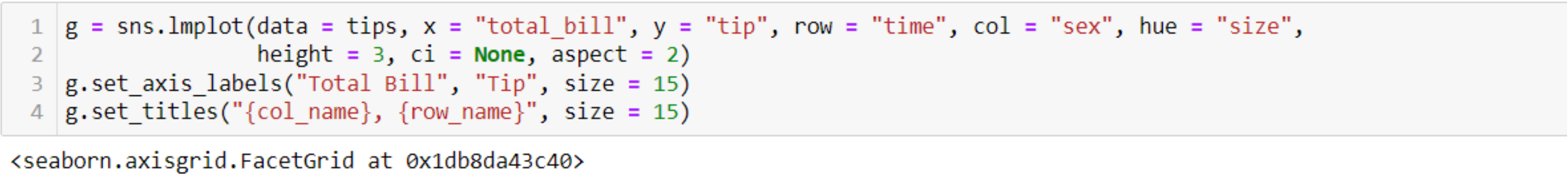


```
g = sns.lmplot(data = tips, x = "total_bill", y = "tip", col = "time", row = "sex", height = 4, aspect=2)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

```
<seaborn.axisgrid.FacetGrid at 0x1db978b0730>
```

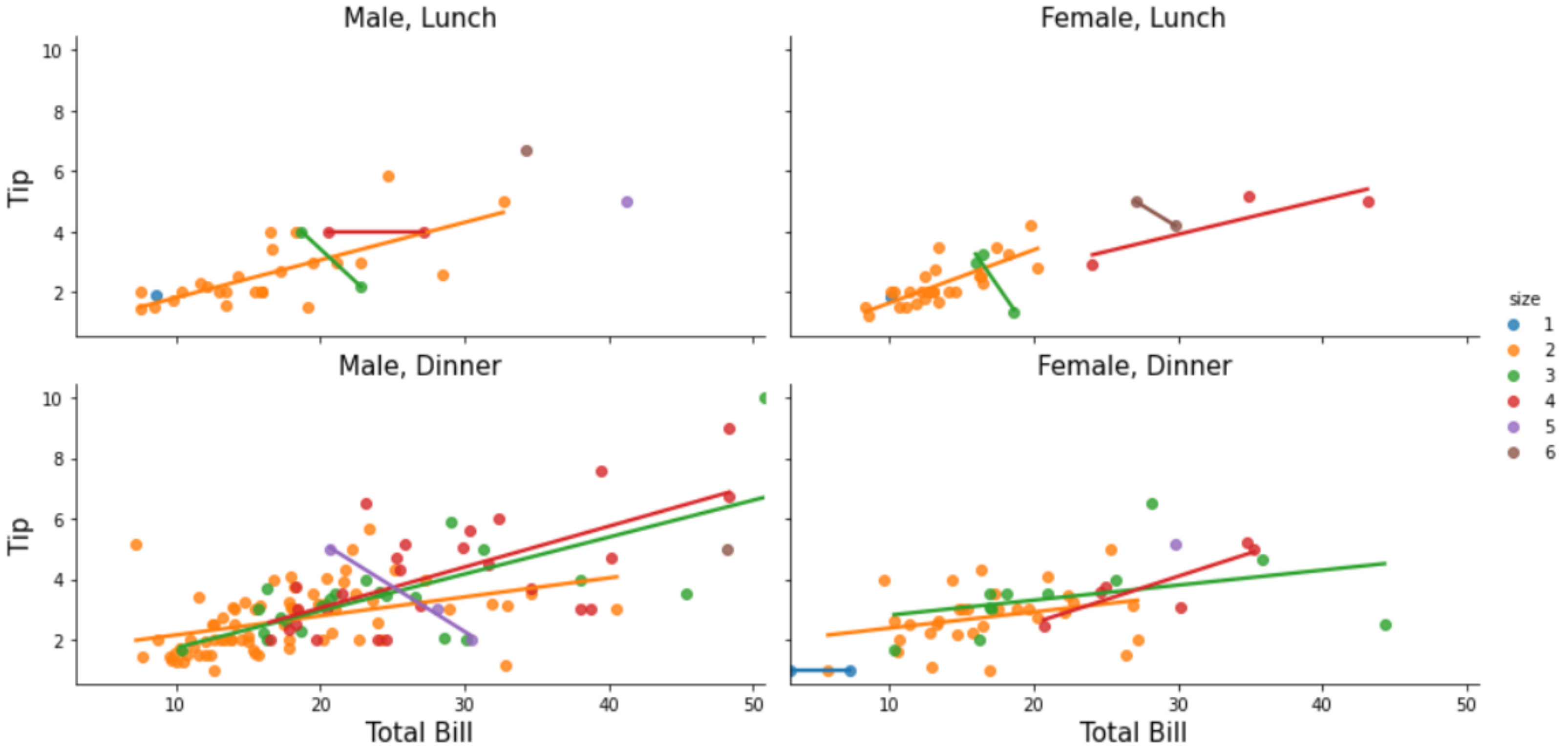


- What about a 3-way interaction?



```
g = sns.lmplot(data = tips, x = "total_bill", y = "tip", row = "time", col = "sex", hue = "size",
               height = 3, ci = None, aspect = 2)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

```
<seaborn.axisgrid.FacetGrid at 0x1db8da43c40>
```

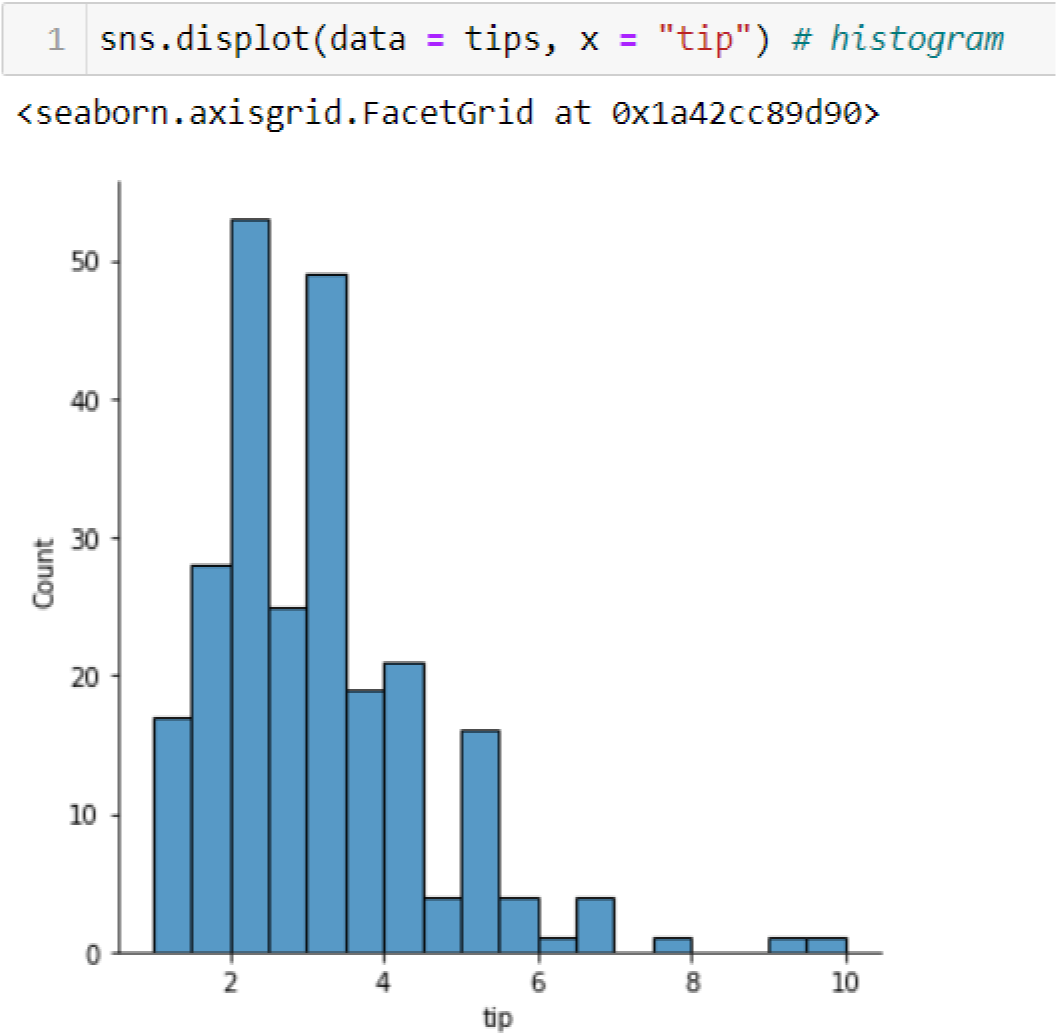


- Can be useful, or a disaster ☺

Displot – distribution plot

- As the name suggests, *displot* visualizes the given sample distribution using *histogram, kernel density estimator (kde)* or its *empirical distribution function (ecdf).*
- Useful for a visual inspection of normality (or other distributions).
- Provides visual interpretation of non-parametric tests such as Mann – Whitney or Kruskal – Wallis.

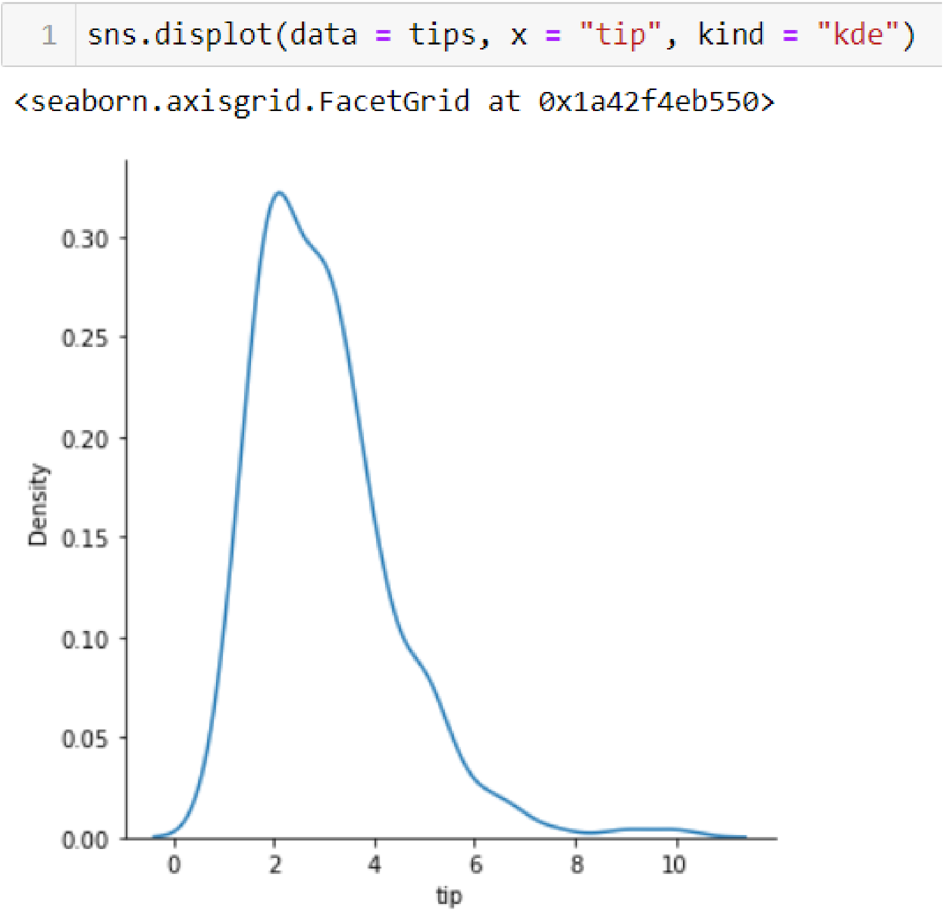
- By default, *displot* produces a histogram.



```
sns.displot(data = tips, x = "tip") # histogram
```

```
<seaborn.axisgrid.FacetGrid at 0x1a42cc89d90>
```

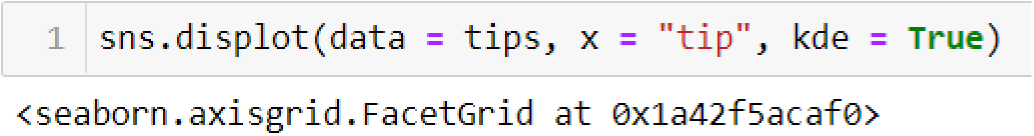
- By passing *kind = “kde”* we get a kernel density estimator of the distribution’s density which is another non-parametric methods such as histograms.



```
sns.displot(data = tips, x = "tip", kind = "kde")
```

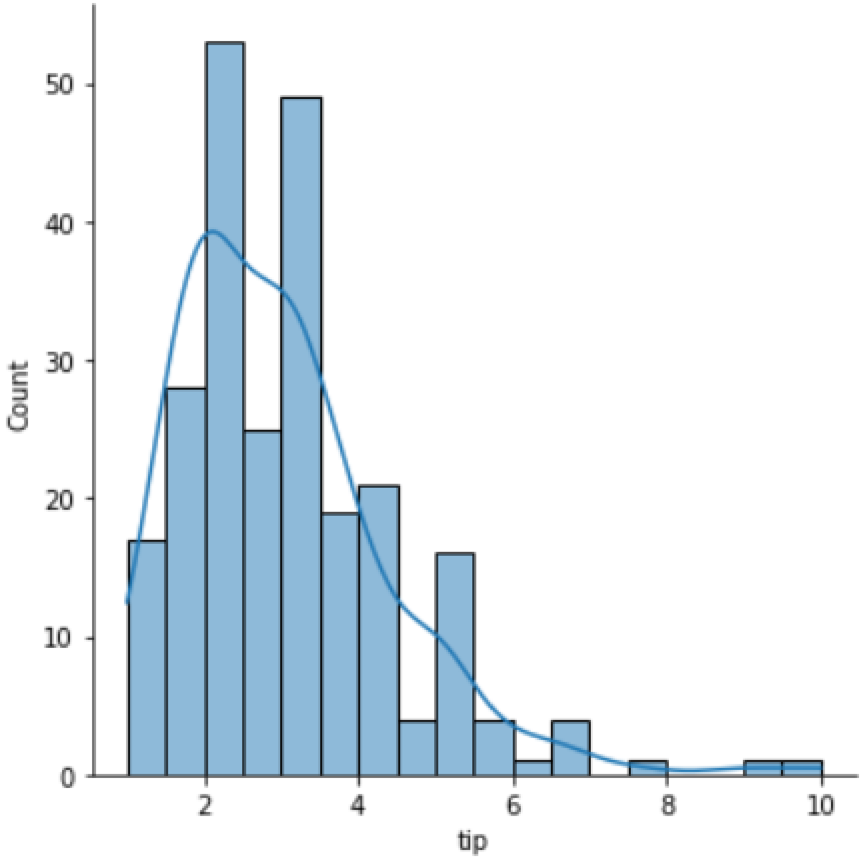
```
<seaborn.axisgrid.FacetGrid at 0x1a42f4eb550>
```

- We can combine them both if we use the histogram option and pass *kde = True.*

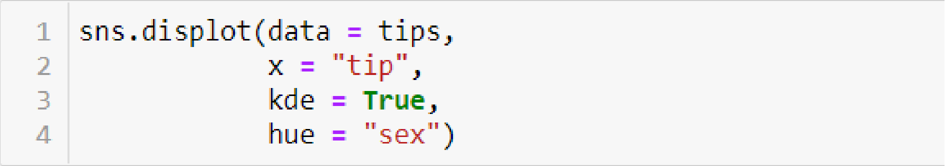


```
sns.displot(data = tips, x = "tip", kde = True)
```

```
<seaborn.axisgrid.FacetGrid at 0x1a42f5acaf0>
```

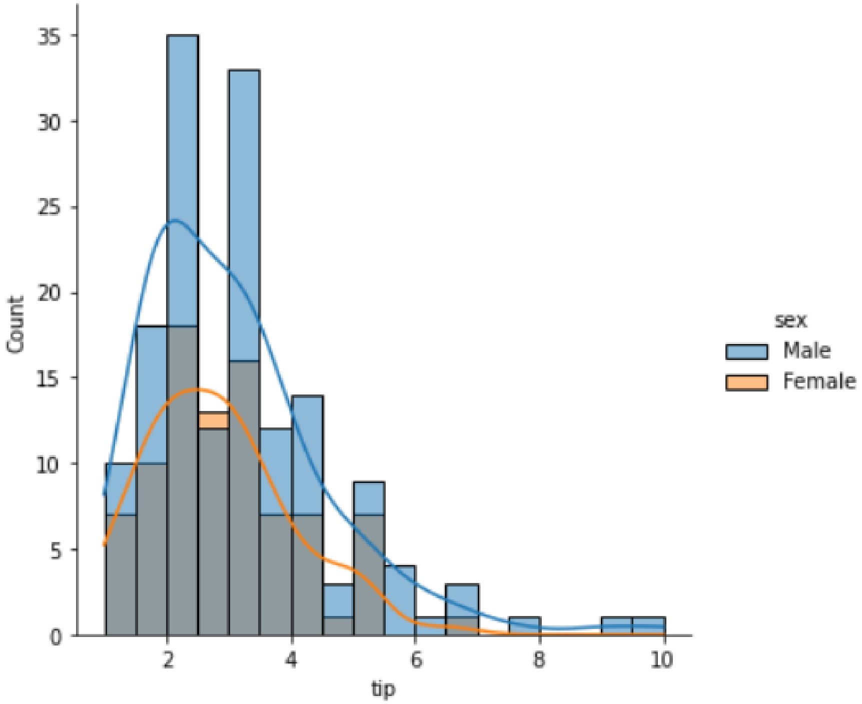


- The *hue* argument is still available.

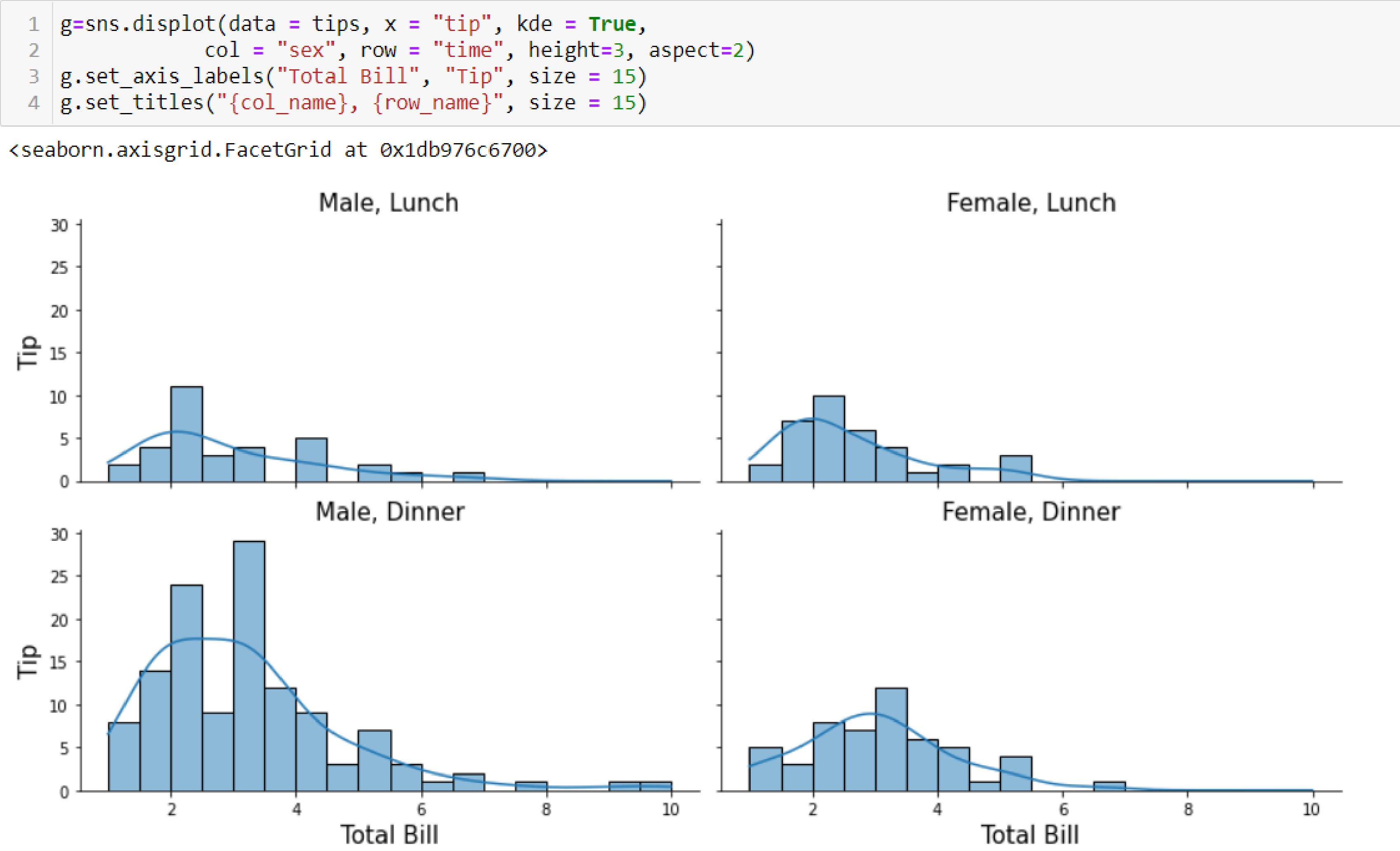


```
sns.displot(data = tips,
            x = "tip",
            kde = True,
            hue = "sex")
```

```
<seaborn.axisgrid.FacetGrid at 0x1a42f5e2160>
```



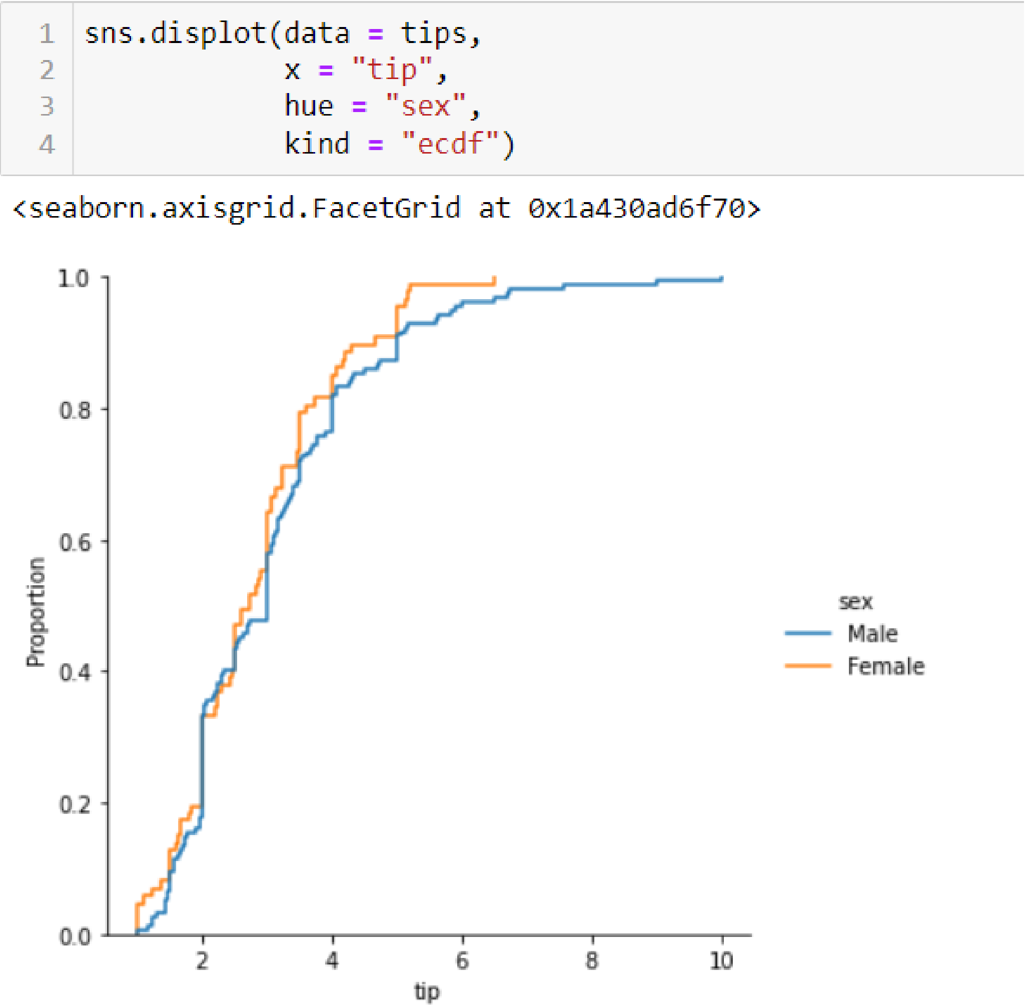
- Faceting is also optional.



```
g=sns.displot(data = tips, x = "tip", kde = True,
              col = "sex", row = "time", height=3, aspect=2)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

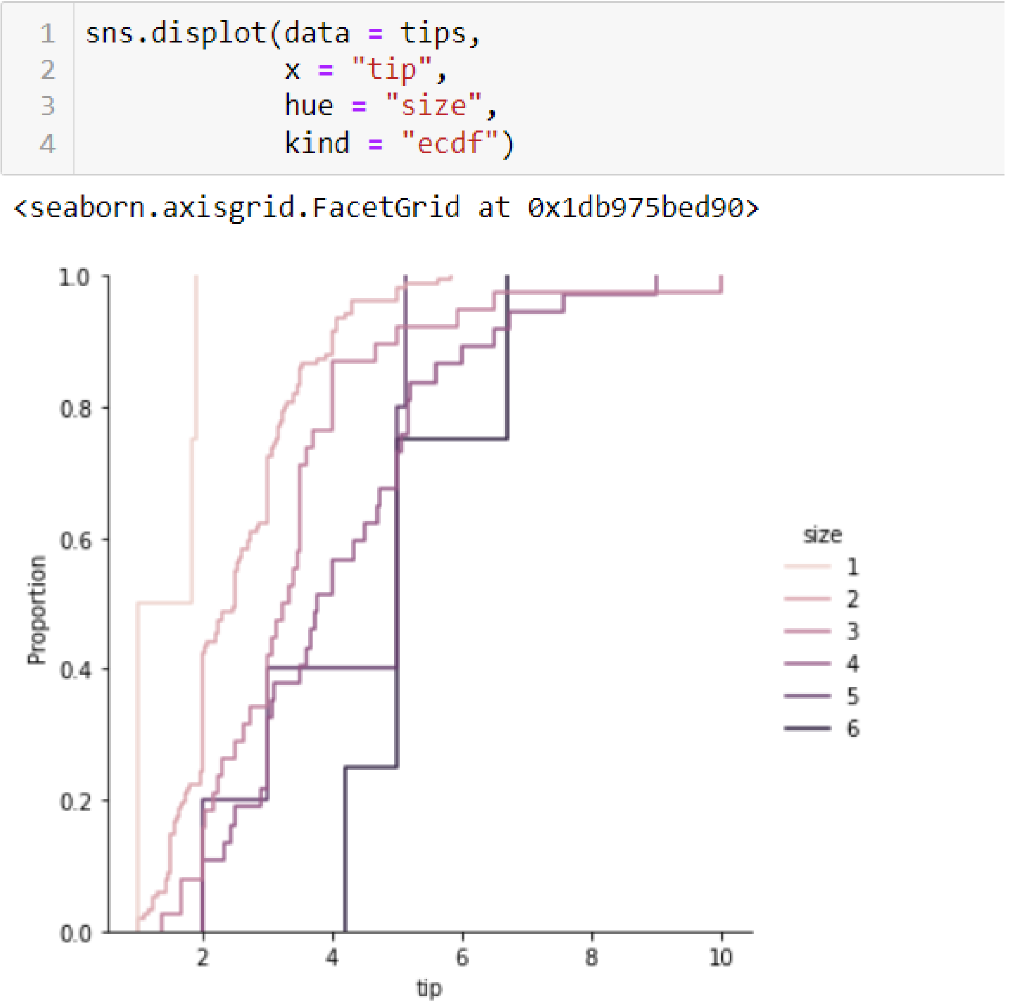
```
<seaborn.axisgrid.FacetGrid at 0x1db976c6700>
```

- When performing non-parametric tests, such as Mann – Whitney or Kruskal – Wallis, we compare the distribution functions of the response across the different groups and reject the null hypothesis if they are significantly different.
- Of course, the *real* distribution is unknown, hence it is estimated using the *Empirical cumulative distribution function (ecdf),* i.e., the estimated distribution function using the given sample.
- The *ecdf* is a non-decreasing step function has values between 0 to 1.
- If for example we perform the Mann – Whitney test and we reject the null hypothesis, we expect to see that the 2 *ecdf* curves are ‘far’ from each other.
- For plotting *ecdf* curves, use *kind = “ecdf”.*



```
sns.displot(data = tips,
            x = "tip",
            hue = "sex",
            kind = "ecdf")
```

```
<seaborn.axisgrid.FacetGrid at 0x1a430ad6f70>
```

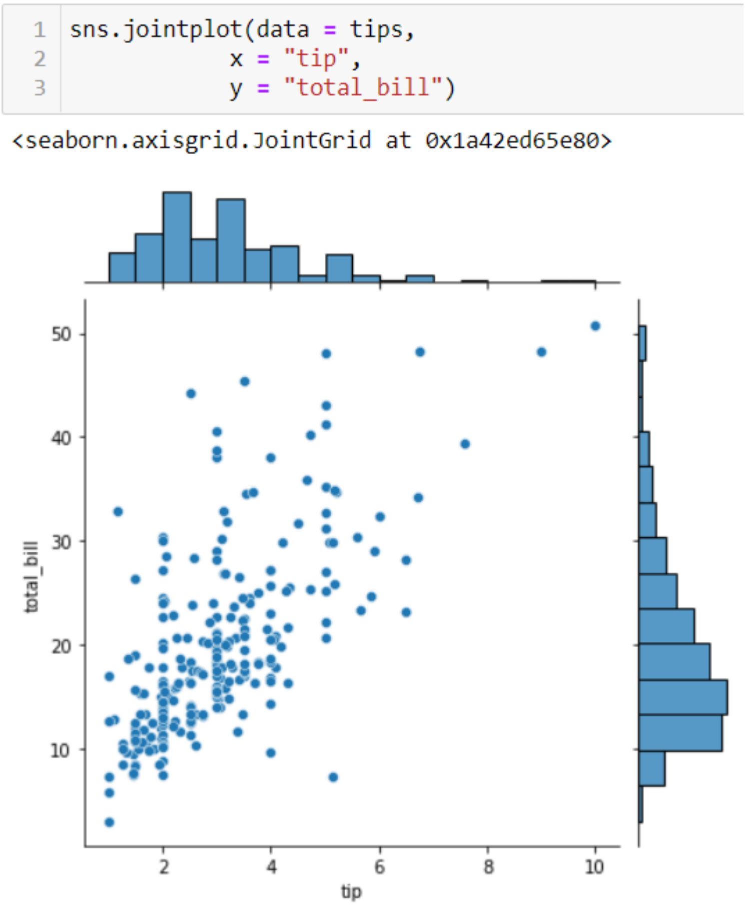


```
sns.displot(data = tips,
            x = "tip",
            hue = "size",
            kind = "ecdf")
```

```
<seaborn.axisgrid.FacetGrid at 0x1db975bed90>
```

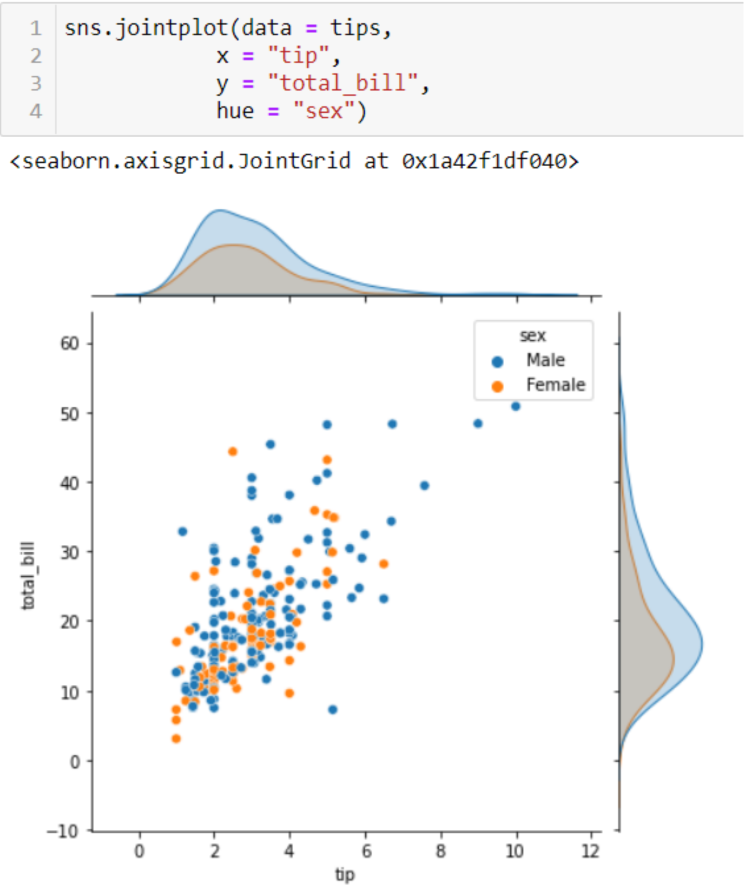
Jointplot

- A nice option for those who are looking to have both bivariate and univariate plots combined.



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

```
<seaborn.axisgrid.JointGrid at 0x1a42ed65e80>
```



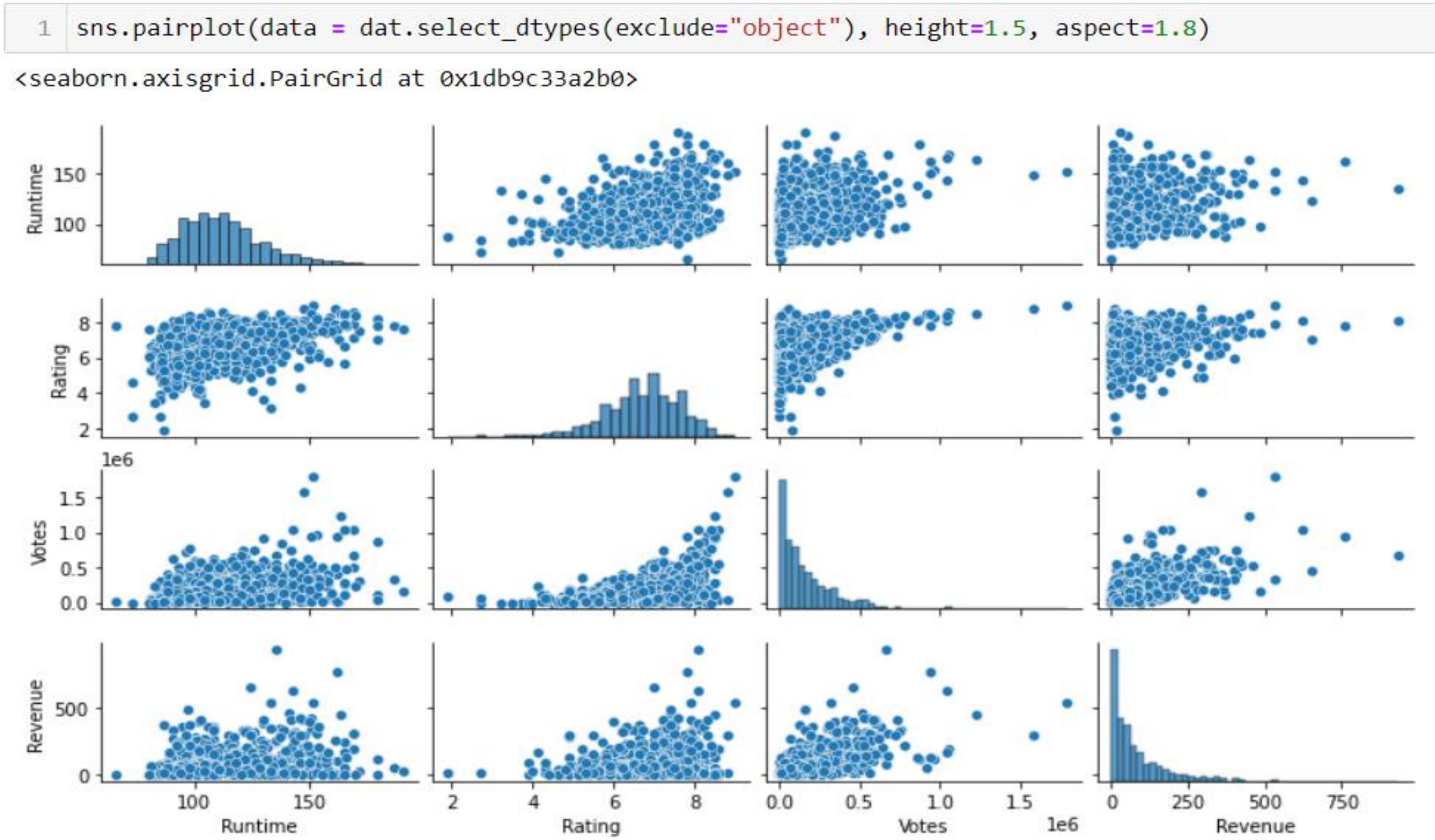
```
sns.jointplot(data = tips,
              x = "tip",
              y = "total_bill",
              hue = "sex")
```

```
<seaborn.axisgrid.JointGrid at 0x1a42f1df040>
```

Pairplot

- Creates a grid of all possible pairs in the data.
- Useful to detect extreme values, clusters within the data, highly correlated variables, etc.
- When the number of variables is large, the plot becomes very hard to interpret and running time can be (very) long.
- We will demonstrate it on our IMDB data.

- Plotting all numerical variables, *select_dtypes(exclude = “object)*

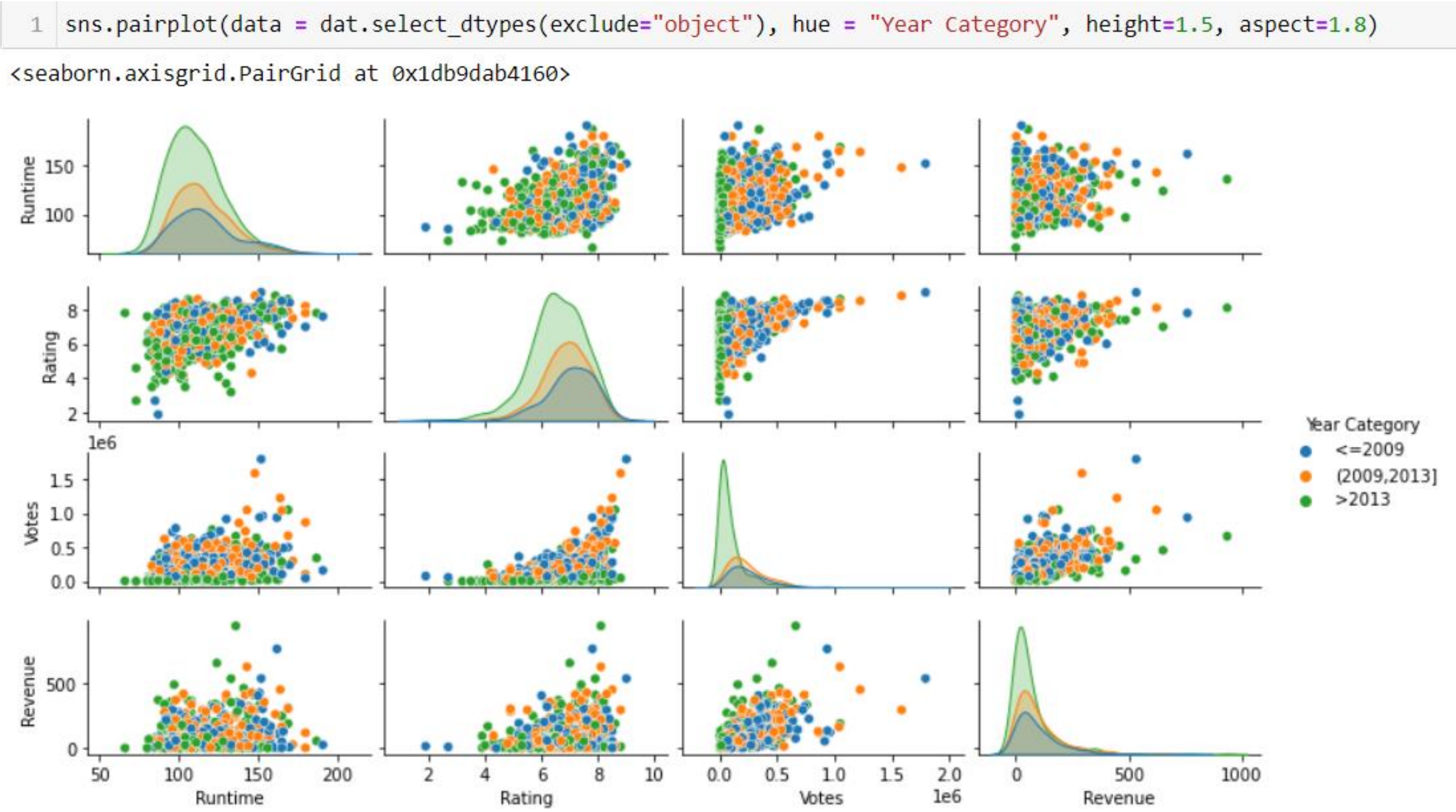


```
sns.pairplot(data = dat.select_dtypes(exclude="object"), height=1.5, aspect=1.8)
```

```
<seaborn.axisgrid.PairGrid at 0x1db9c33a2b0>
```

- *The main diagonal is by default the histogram of each variable, can be “kde” as well.*

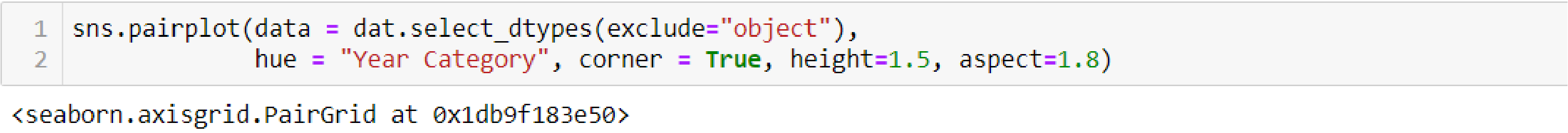
- Another example, when using a grouping variable as color.



```
sns.pairplot(data = dat.select_dtypes(exclude="object"), hue = "Year Category", height=1.5, aspect=1.8)
```

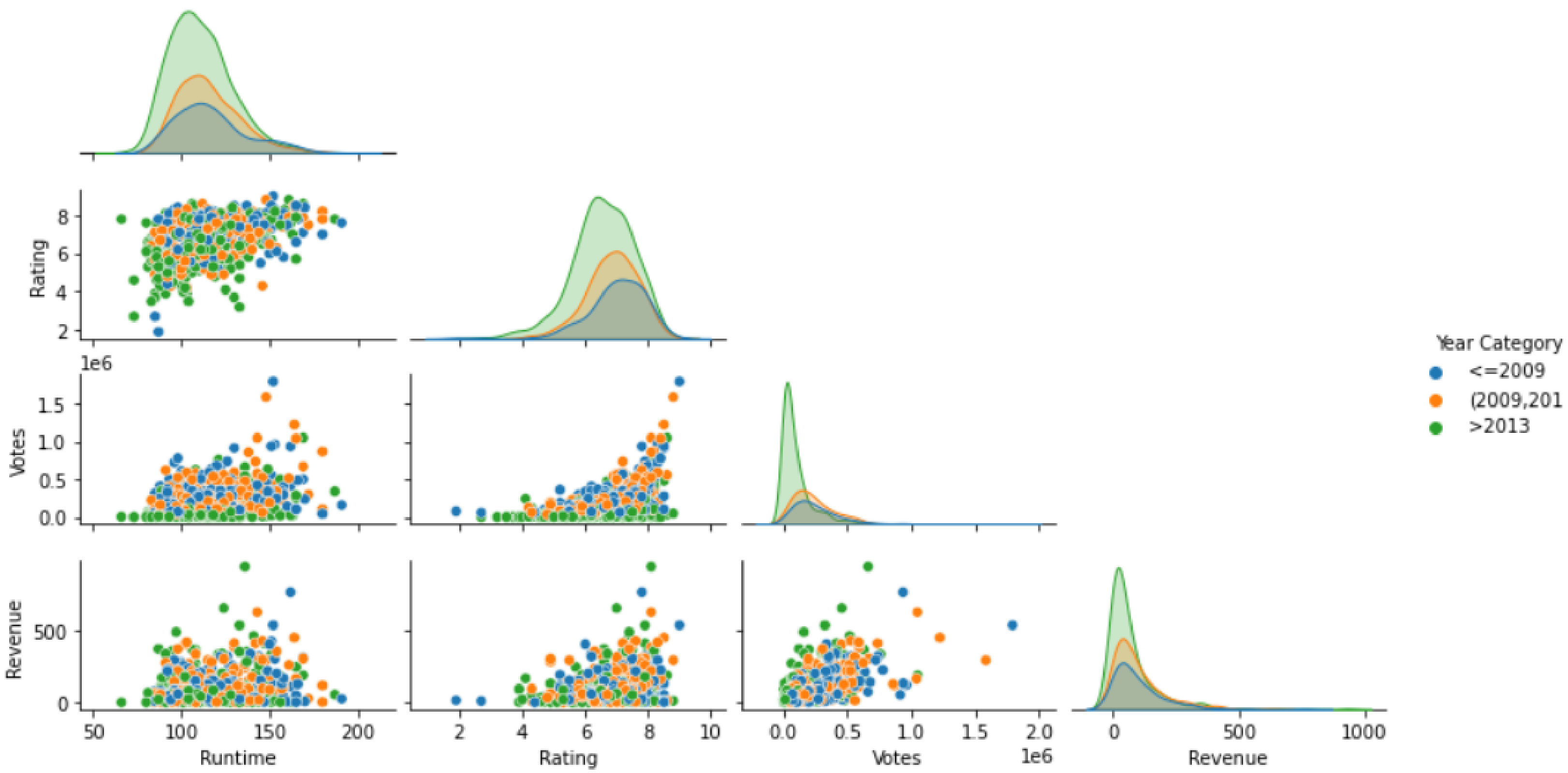
```
<seaborn.axisgrid.PairGrid at 0x1db9dab4160>
```

- We can include only the lower triangle of the grid by using *corner =True.*



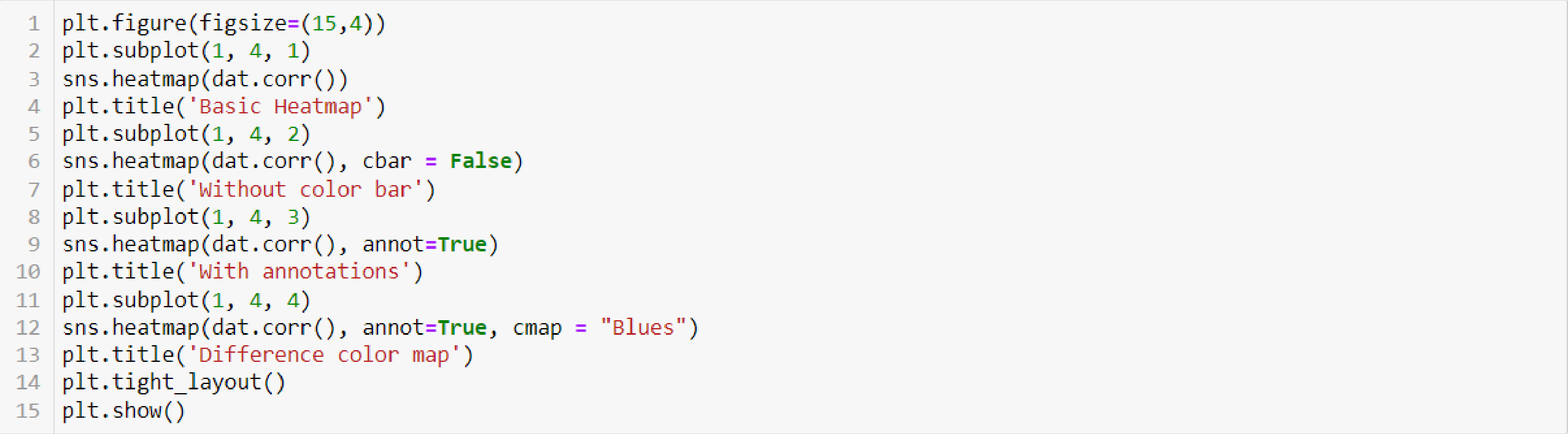
```
sns.pairplot(data = dat.select_dtypes(exclude="object"),
             hue = "Year Category", corner = True, height=1.5, aspect=1.8)
```

```
<seaborn.axisgrid.PairGrid at 0x1db9f183e50>
```

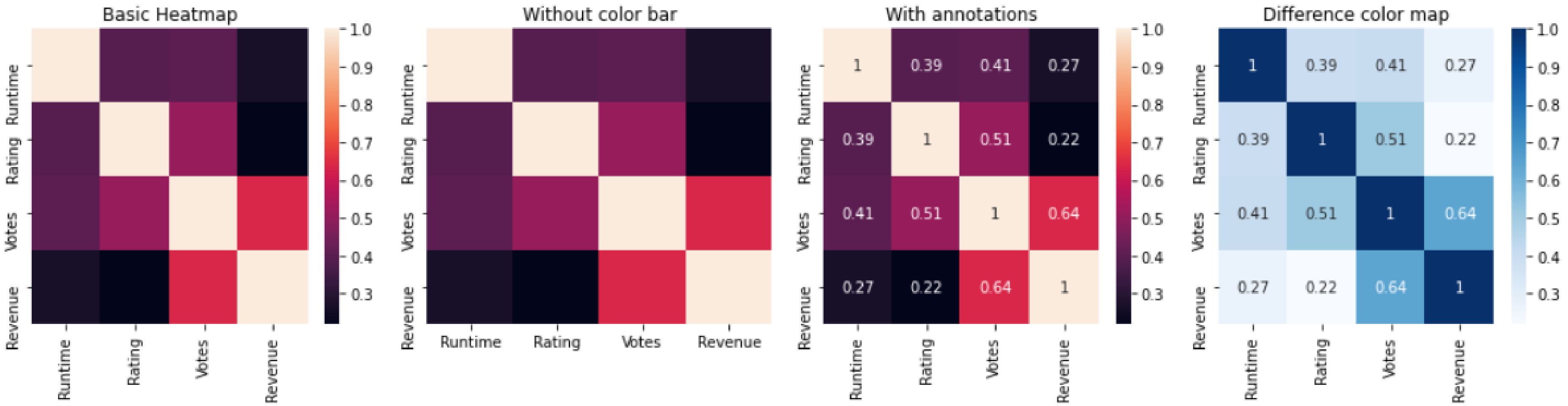


Heatmap

- A tool for visualizing 2D arrays, such as correlation matrices, confusion matrices and many other options.

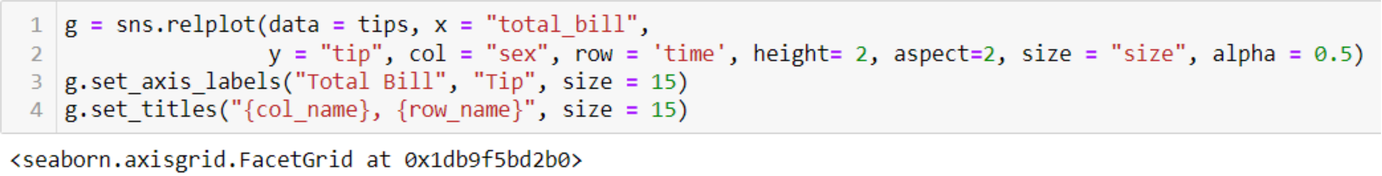


```
plt.figure(figsize=(15,4))
plt.subplot(1, 4, 1)
sns.heatmap(dat.corr())
plt.title('Basic Heatmap')
plt.subplot(1, 4, 2)
sns.heatmap(dat.corr(), cbar = False)
plt.title('Without color bar')
plt.subplot(1, 4, 3)
sns.heatmap(dat.corr(), annot=True)
plt.title('With annotations')
plt.subplot(1, 4, 4)
sns.heatmap(dat.corr(), annot=True, cmap = "Blues")
plt.title('Difference color map')
plt.tight_layout()
plt.show()
```



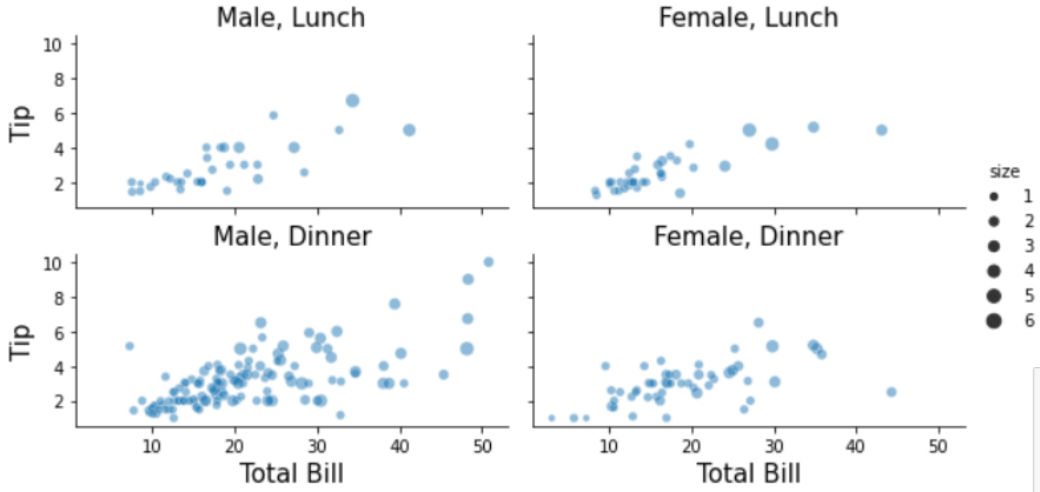
Set Theme, pallete and grid

- The seaborn module supports many graphics adjustments such as backgrounds, grid lines, color maps, palettes and many other.
- Since time is limited, we will only demonstrate a few of them.
- Most of the adjustments can be made using the *set_theme()* function, or through the plotting function (e.g. displot, catplot) itself.
- First example, if we use faceting, we can set the *x* and *y* axes to be different for each separate plot.



```
g = sns.relplot(data = tips, x = "total_bill",
                y = "tip", col = "sex", row = 'time', height= 2, aspect=2, size = "size", alpha = 0.5)
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

```
<seaborn.axisgrid.FacetGrid at 0x1db9f5bd2b0>
```

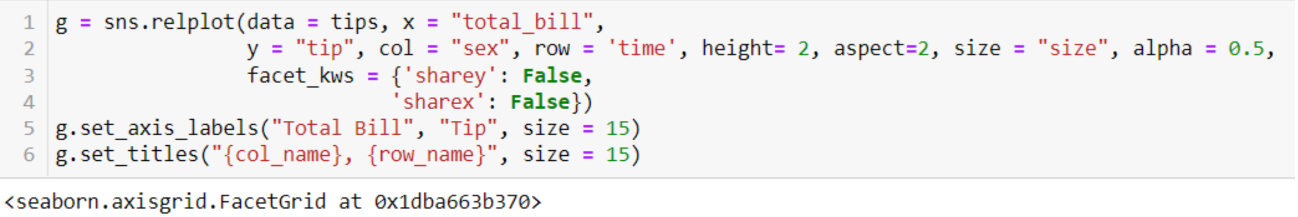


Male, Lunch

Female, Lunch

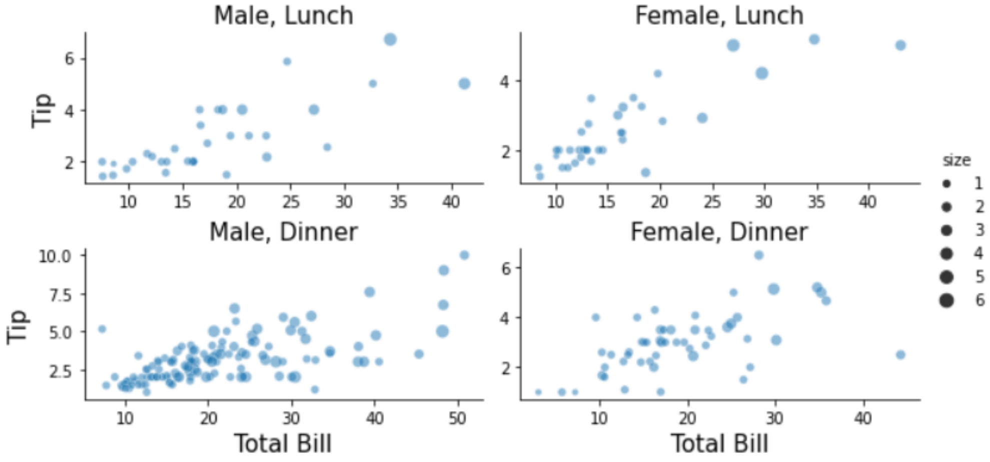
Male, Dinner

Female, Dinner



```
g = sns.relplot(data = tips, x = "total_bill",
                y = "tip", col = "sex", row = 'time', height= 2, aspect=2, size = "size", alpha = 0.5,
                facet_kws = {'sharey': False,
                             'sharex': False})
g.set_axis_labels("Total Bill", "Tip", size = 15)
g.set_titles("{col_name}, {row_name}", size = 15)
```

```
<seaborn.axisgrid.FacetGrid at 0x1dba663b370>
```



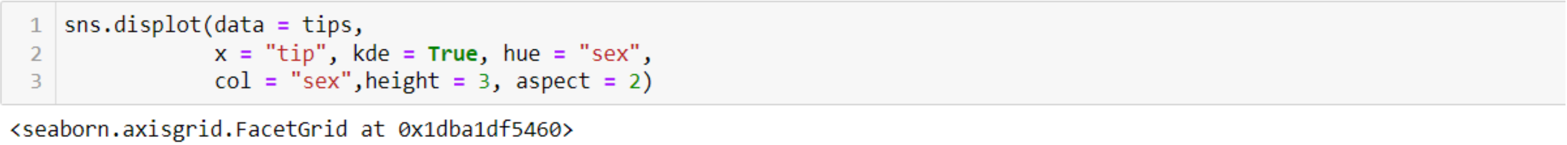
Male, Lunch

Female, Lunch

Male, Dinner

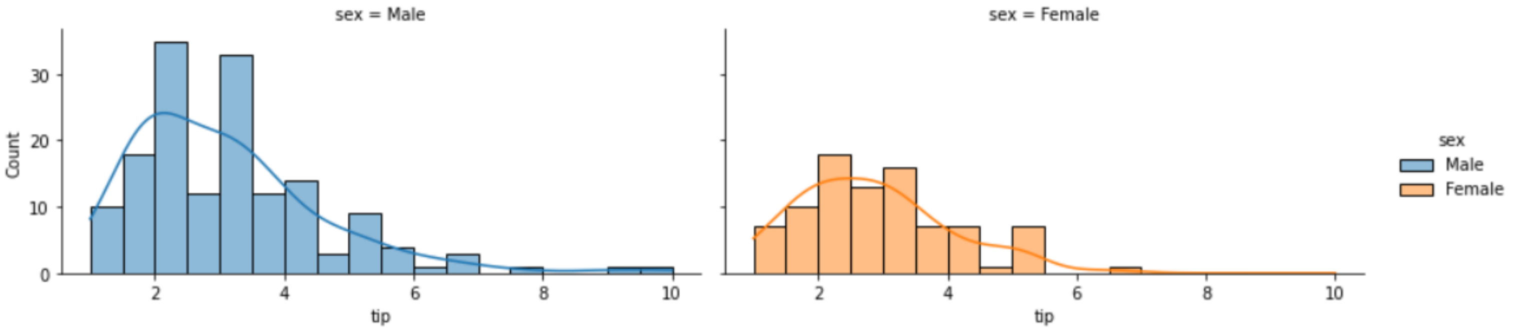
Female, Dinner

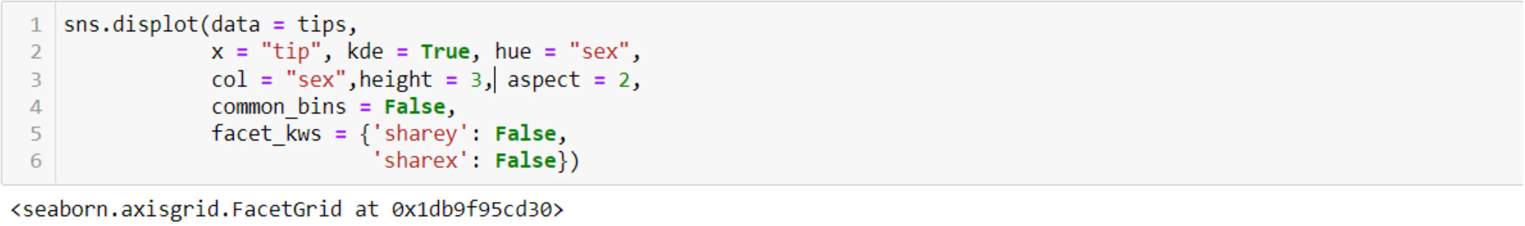
- If we using *displot* and histograms, we also need to use *common_bins = False.*



```
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2)
```

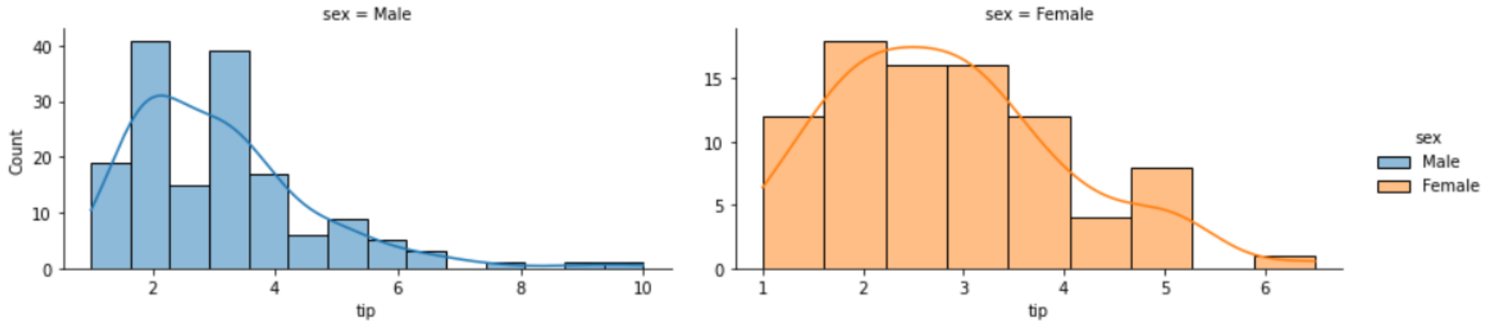
```
<seaborn.axisgrid.FacetGrid at 0x1dba1df5460>
```



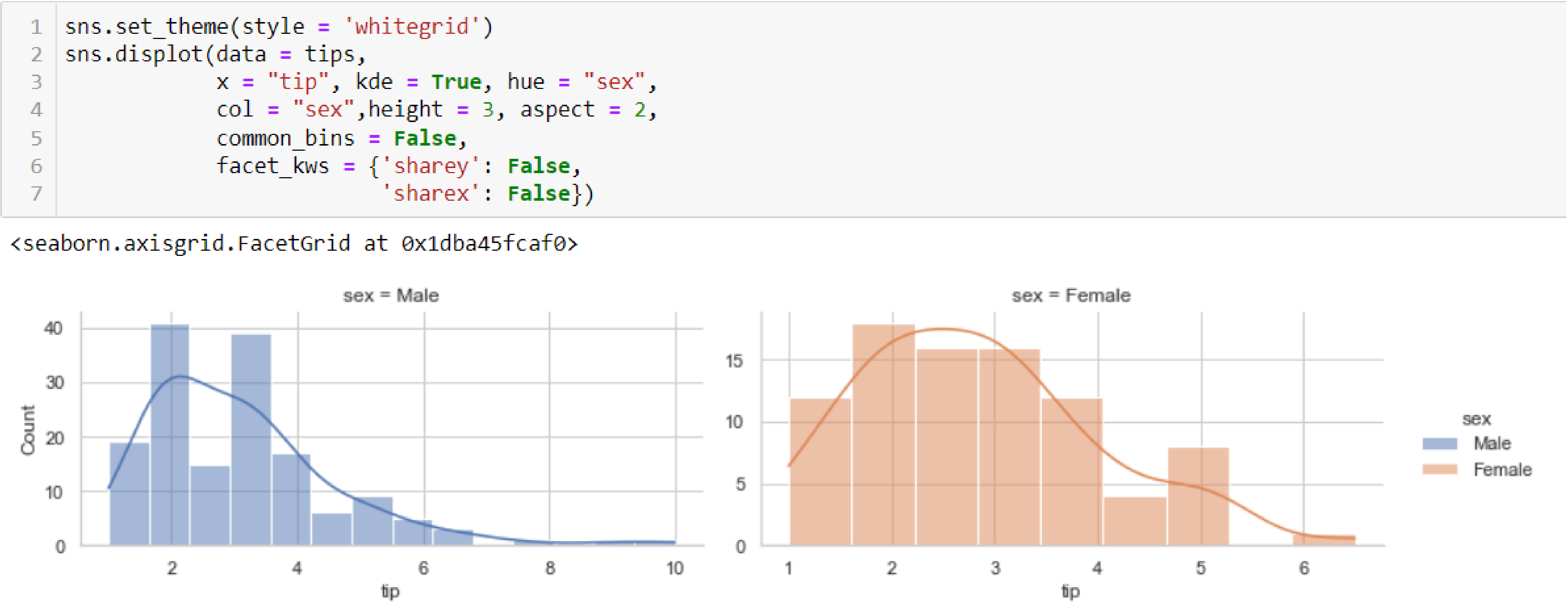


```
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

```
<seaborn.axisgrid.FacetGrid at 0x1db9f95cd30>
```



- In order to control the background style, we can do the following:

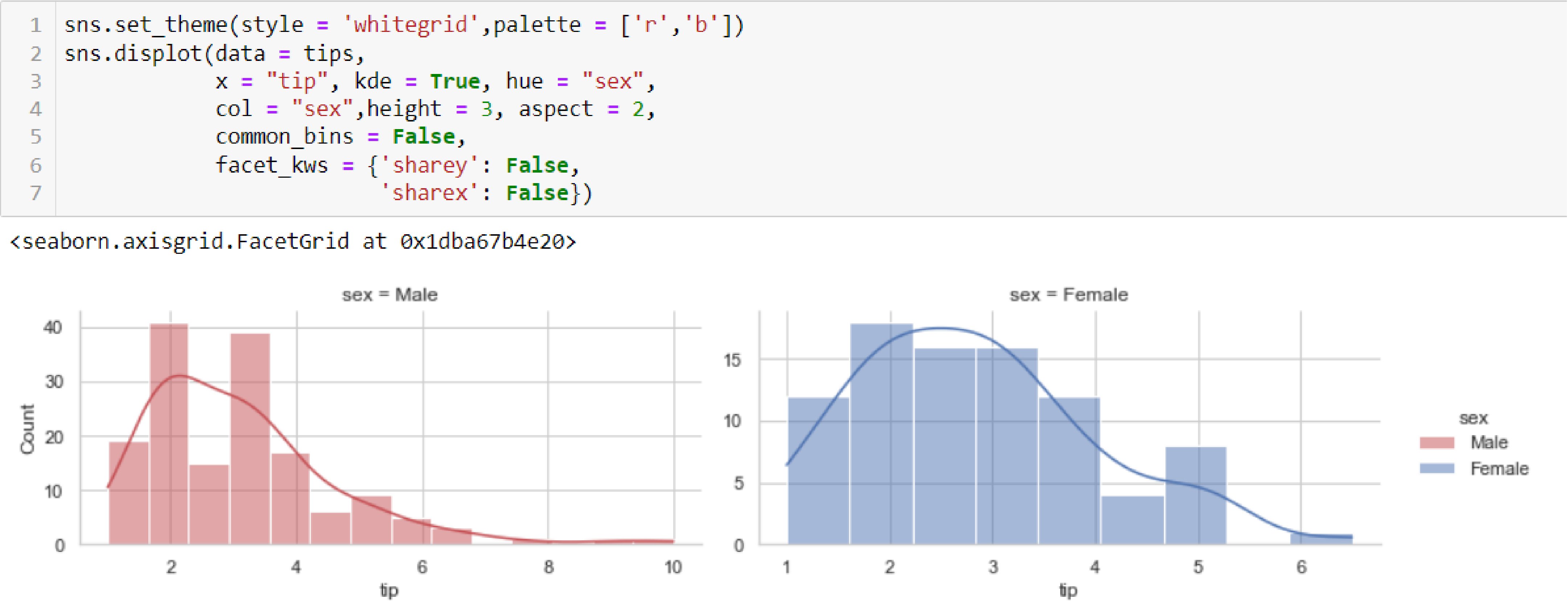


```
sns.set_theme(style = 'whitegrid')
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

```
<seaborn.axisgrid.FacetGrid at 0x1dba45fcaf0>
```

- The default *palette* is called *deep*, which is not the default when we’re not using this function, that is why we got different colors.

- We can use the *palette* argument and choose different colors.



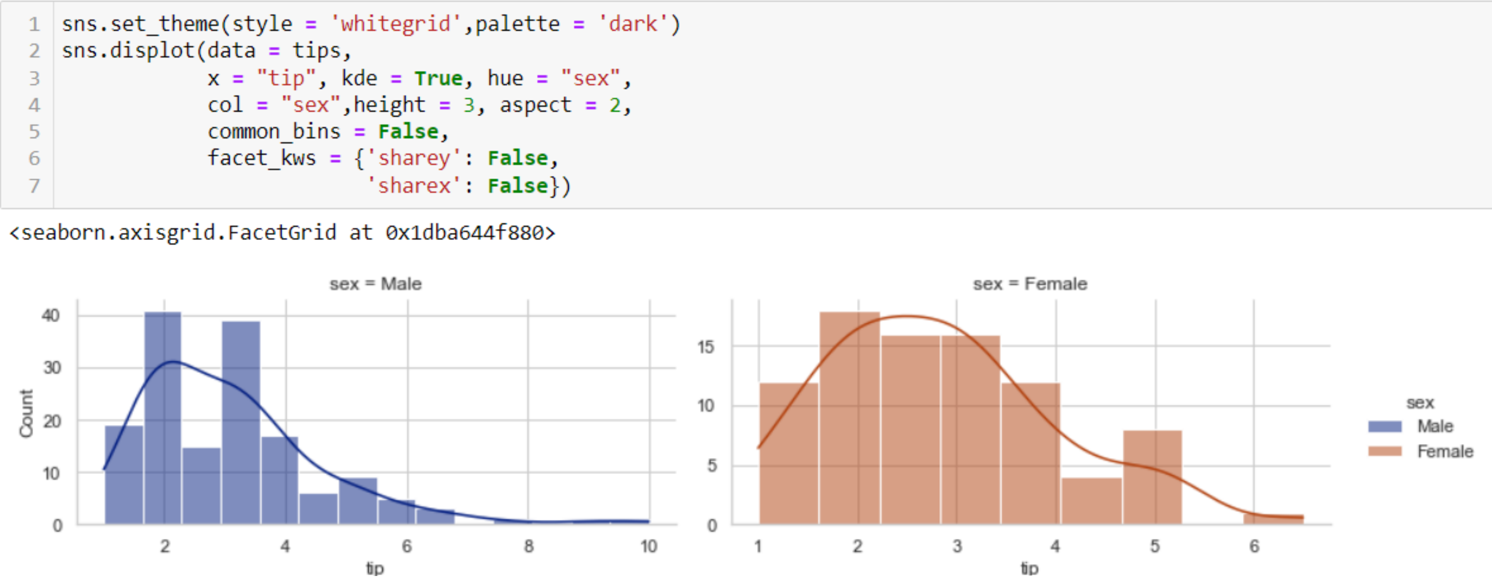
```
sns.set_theme(style = 'whitegrid',palette = ['r','b'])
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

```
<seaborn.axisgrid.FacetGrid at 0x1dba67b4e20>
```

sex = Male

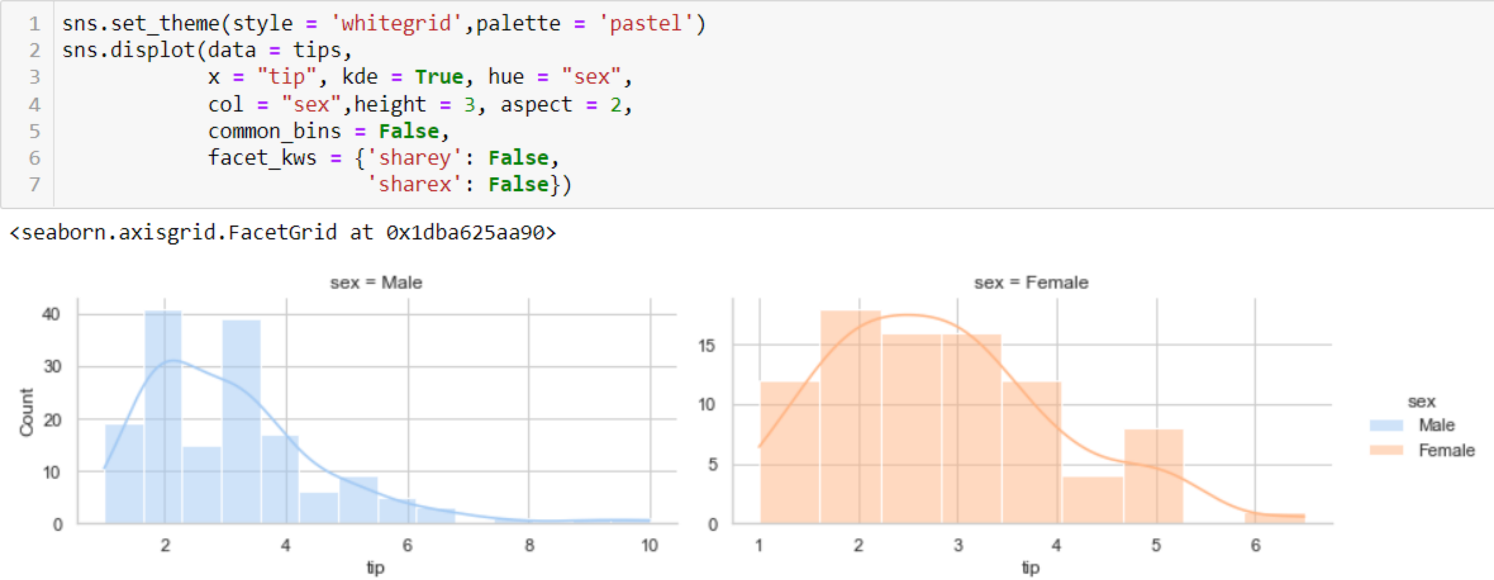
sex = Female

- Or use built-in palettes:



```
sns.set_theme(style = 'whitegrid',palette = 'dark')
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

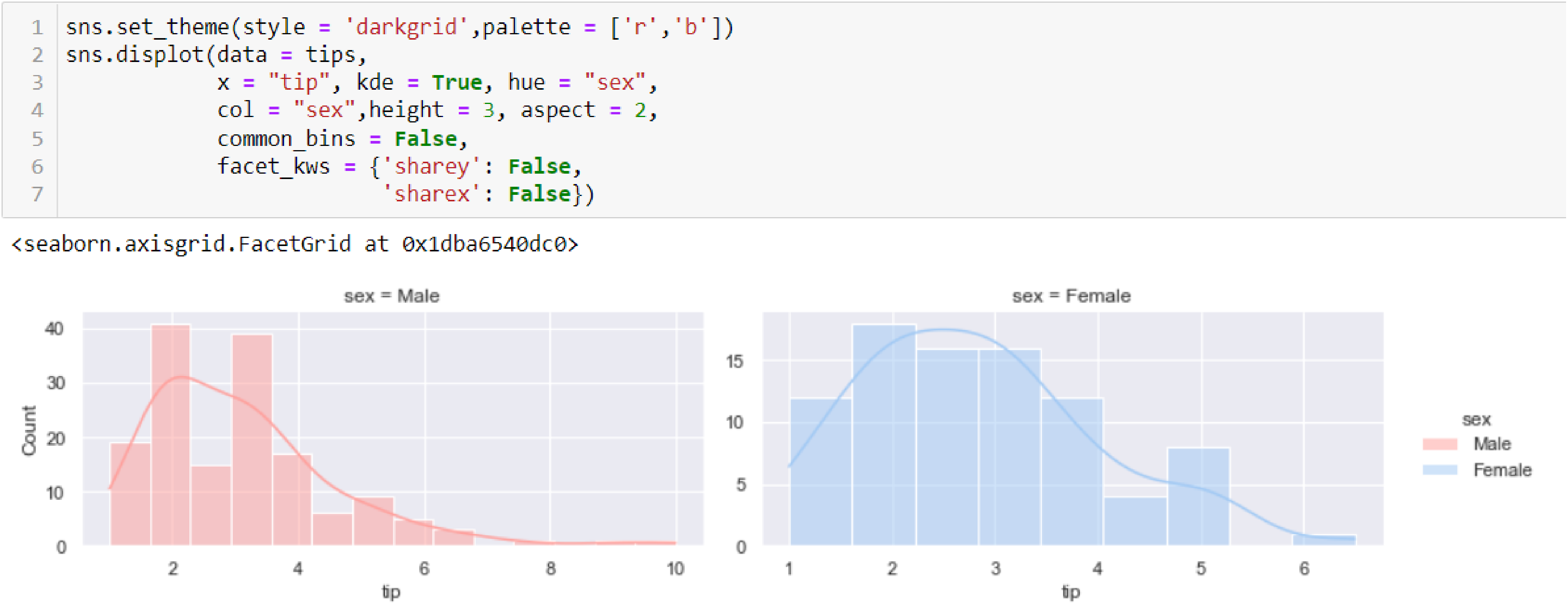
```
<seaborn.axisgrid.FacetGrid at 0x1dba644f880>
```



```
sns.set_theme(style = 'whitegrid',palette = 'pastel')
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

```
<seaborn.axisgrid.FacetGrid at 0x1dba625aa90>
```

- Additional style is the “*darkgrid*”.



```
sns.set_theme(style = 'darkgrid',palette = ['r','b'])
sns.displot(data = tips,
            x = "tip", kde = True, hue = "sex",
            col = "sex",height = 3, aspect = 2,
            common_bins = False,
            facet_kws = {'sharey': False,
                         'sharex': False})
```

```
<seaborn.axisgrid.FacetGrid at 0x1dba6540dc0>
```

Jupyter Time!