



INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

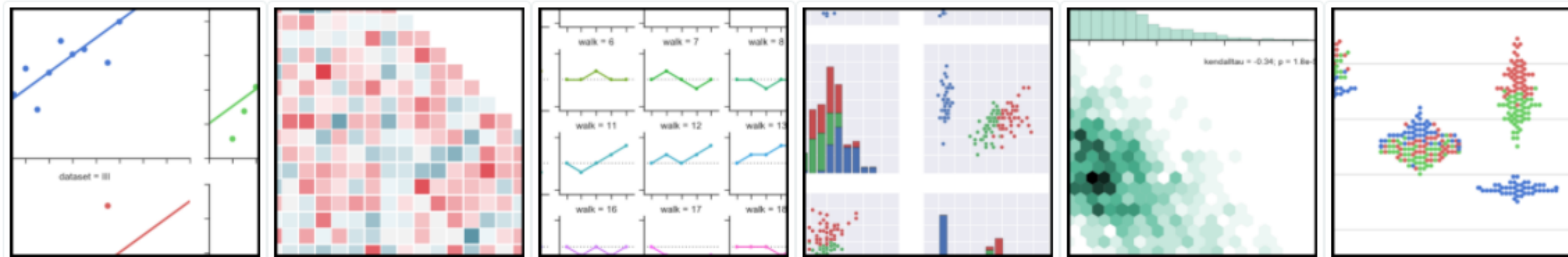
# Visualizing Regressions



# Seaborn

seaborn 0.7.1 API Tutorial Gallery Site ▾ Page ▾

## Seaborn: statistical data visualization



Seaborn is a Python visualization library based on matplotlib. It provides a high-level interface for drawing attractive statistical graphics.

[Documentation](#)

[Features](#)

<https://stanford.edu/~mwaskom/software/seaborn/>

# Recap: Pandas DataFrames

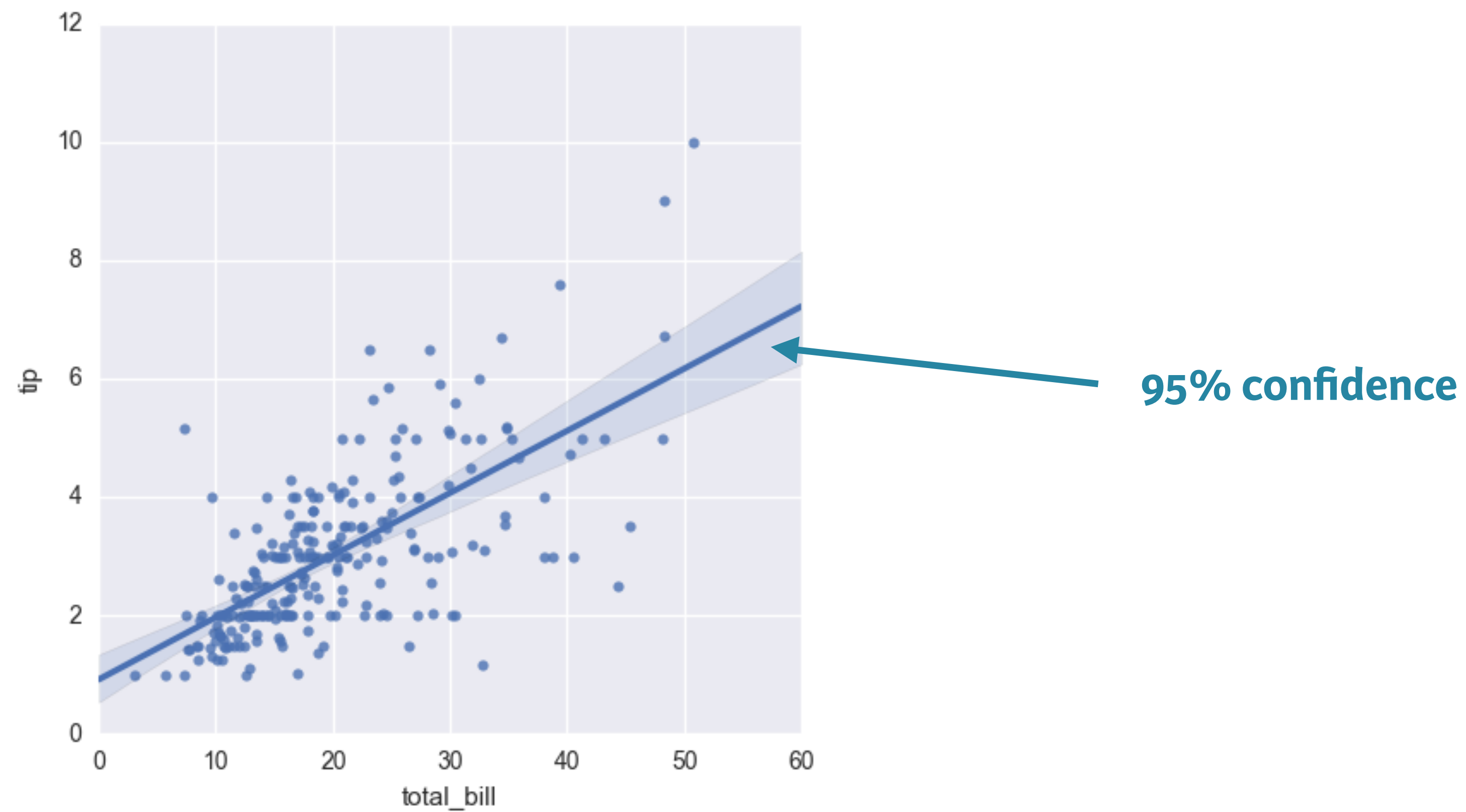
- Labelled tabular data structure
- Labels on rows: *index*
- Labels on columns: *columns*
- Columns are Pandas *Series*



# Recap: Pandas DataFrames

	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
...	...	...	...	...	...	...	...

# Linear regression plots





# Using Implot()

```
In [1]: import pandas as pd
```

```
In [2]: import matplotlib.pyplot as plt
```

```
In [3]: import seaborn as sns
```

```
In [4]: tips =sns.load_dataset('tips')
```

```
In [5]: sns.lmplot(x= 'total_bill', y='tip', data=tips)
```

```
In [6]: plt.show()
```

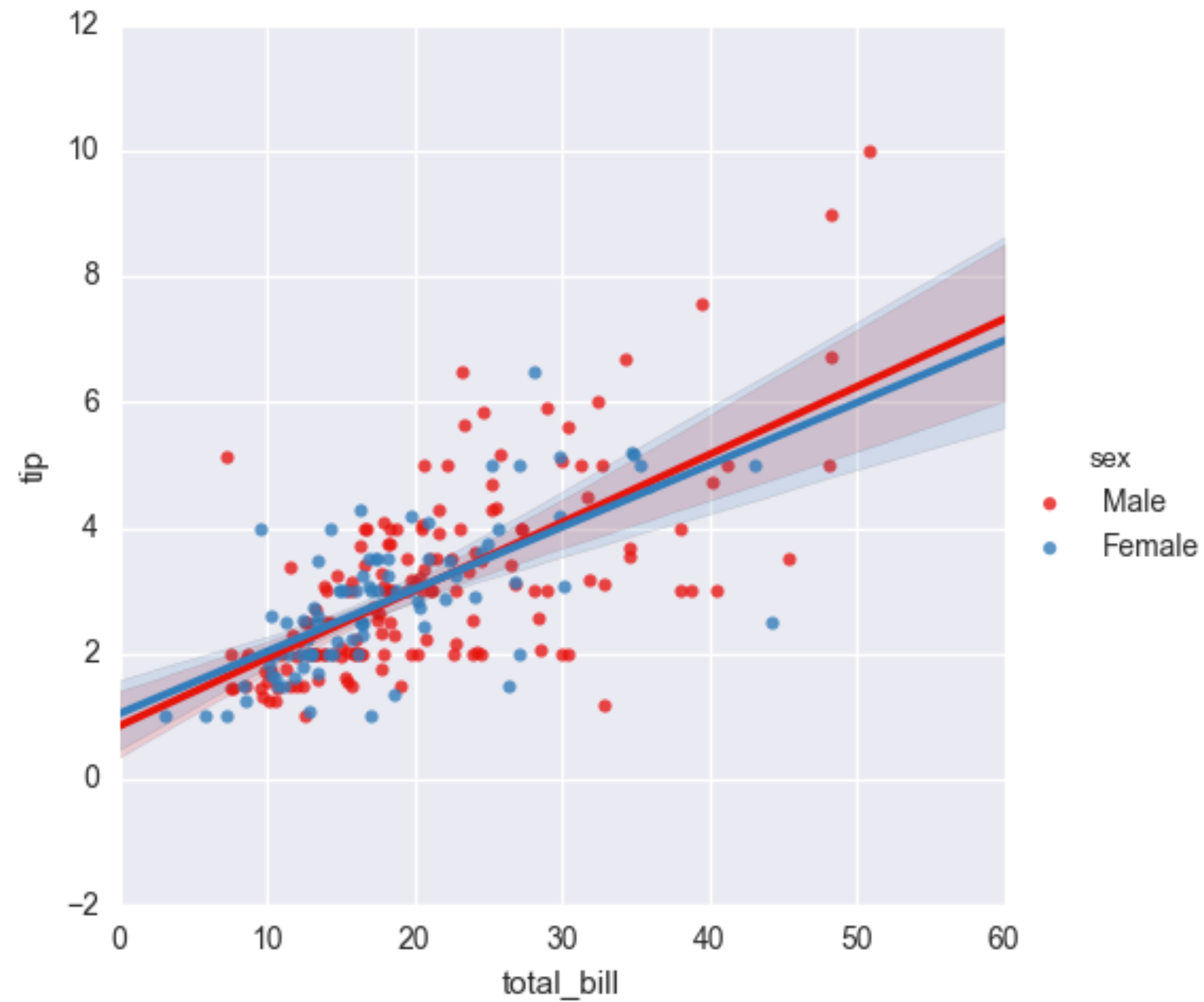


# Factors

	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
...	...	...	...	...	...	...	...



# Grouping factors (same plot)







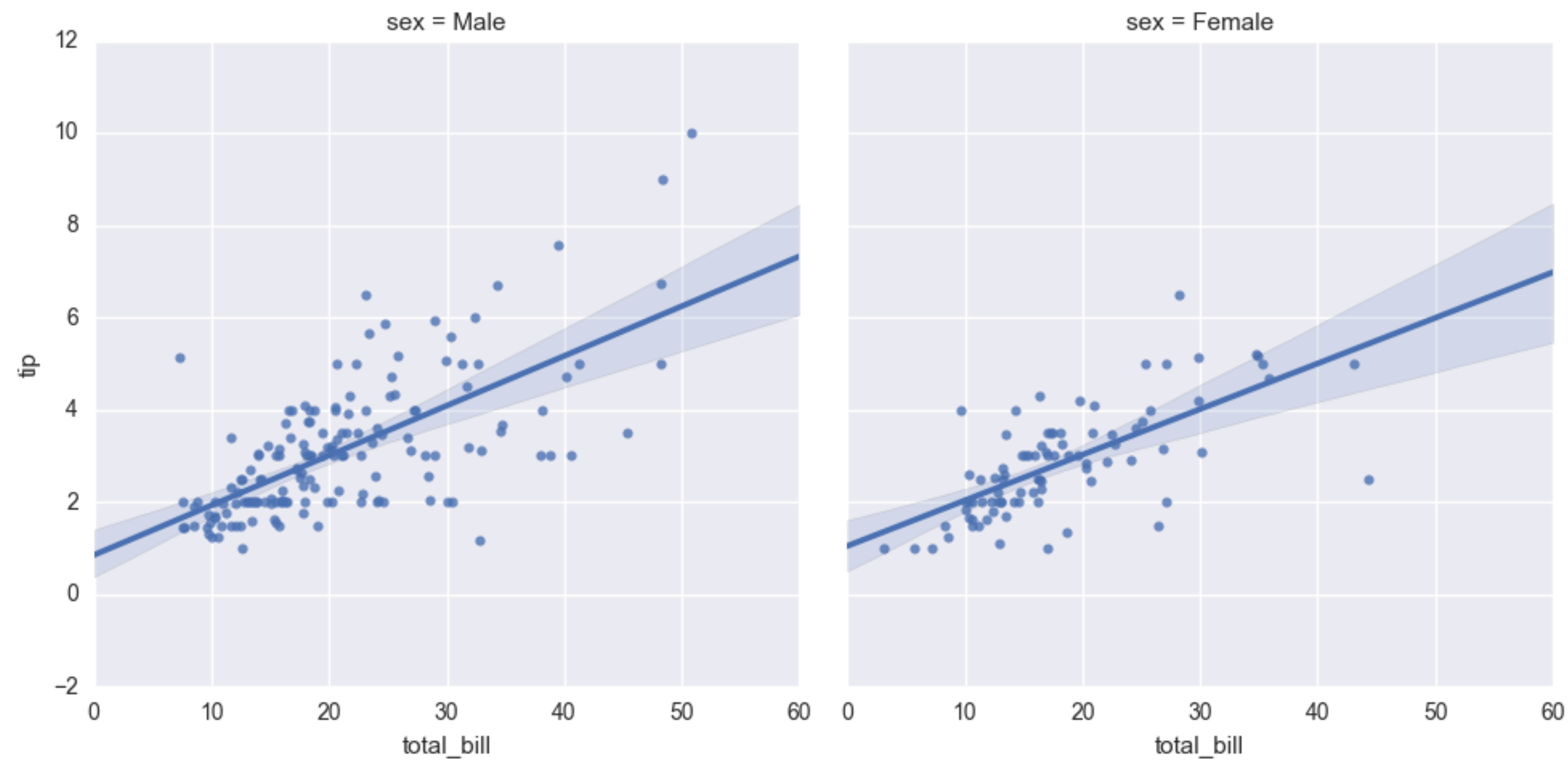
# Using hue=...

```
In [7]: sns.lmplot(x='total_bill', y='tip', data=tips, hue='sex',  
    ...:           palette='Set1')
```

```
In [8]: plt.show()
```



# Grouping factors (subplots)





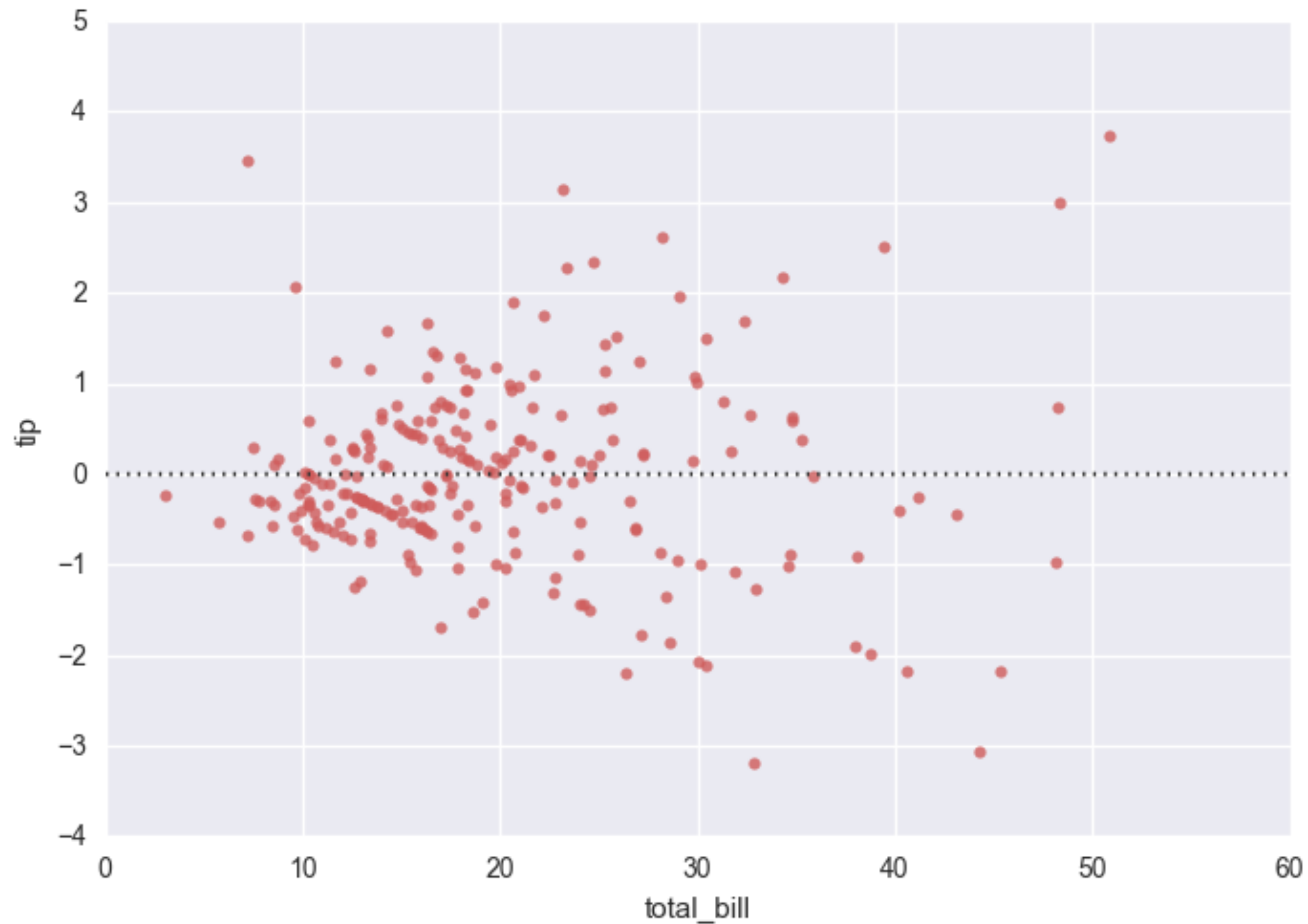
# Using col=...

```
In [9]: sns.lmplot(x='total_bill', y='tip', data=tips, col='sex')
```

```
In [10]: plt.show()
```



# Residual plots





# Using residplot()

```
In [11]: sns.residplot(x='age',y='fare',data=tips,color='indianred')
```

```
In [12]: plt.show()
```

- Similar arguments as lmpplot() but more flexible
  - x, y can be *arrays or strings*
  - data is DataFrame (optional)
- Optional arguments (e.g., color) as in Matplotlib



INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

**Let's practice!**



INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

# **Visualizing univariate distributions**

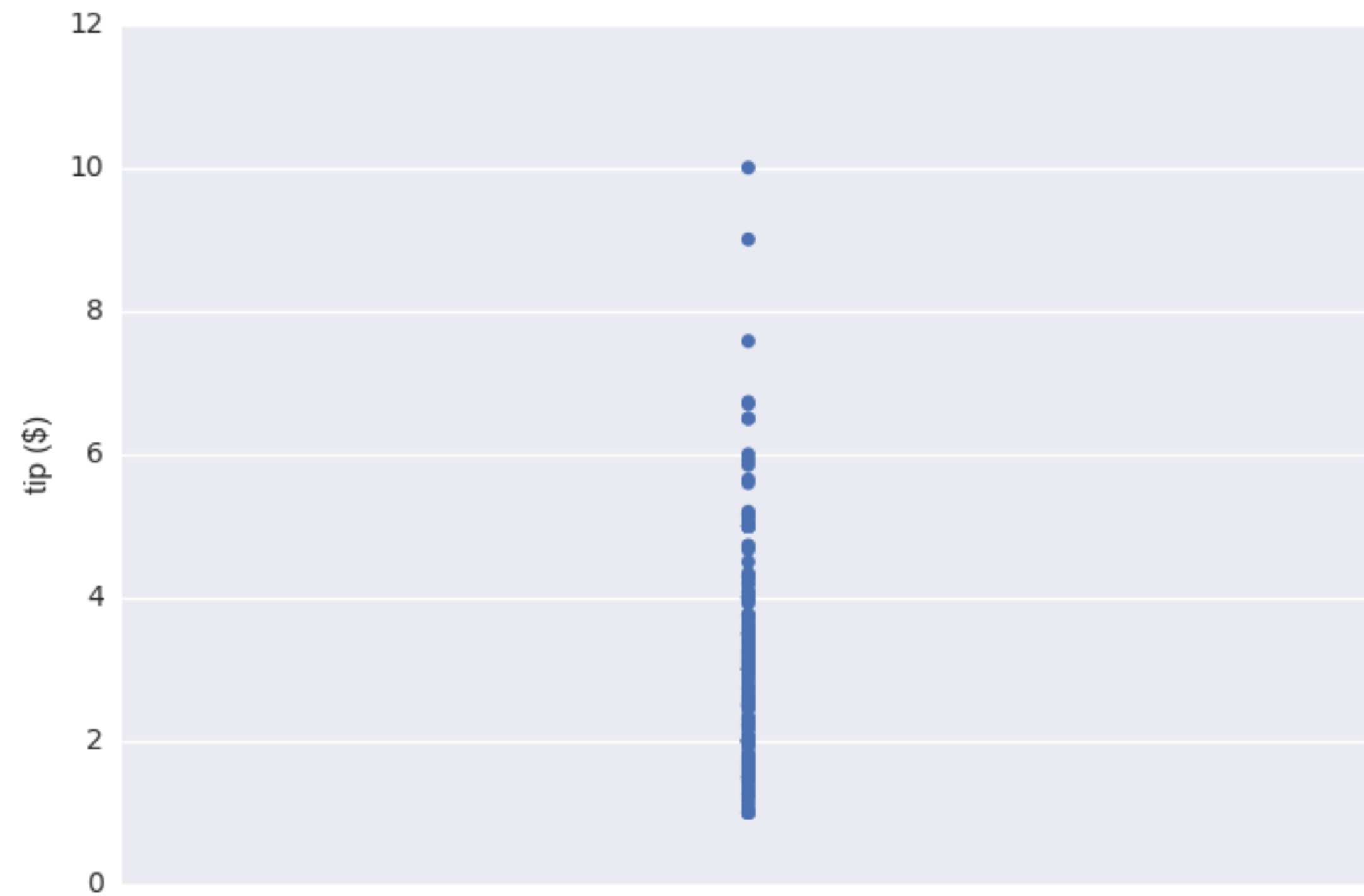
# Visualizing data

- Univariate → “one variable”
- Visualization techniques for sampled univariate data
  - Strip plots
  - Swarm plots
  - Violin plots





# Strip plot





# Using `stripplot()`

```
In [1]: sns.stripplot(y= 'tip', data=tips)
```

```
In [2]: plt.ylabel('tip ($)')
```

```
In [3]: plt.show()
```



# Grouping with `stripplot()`

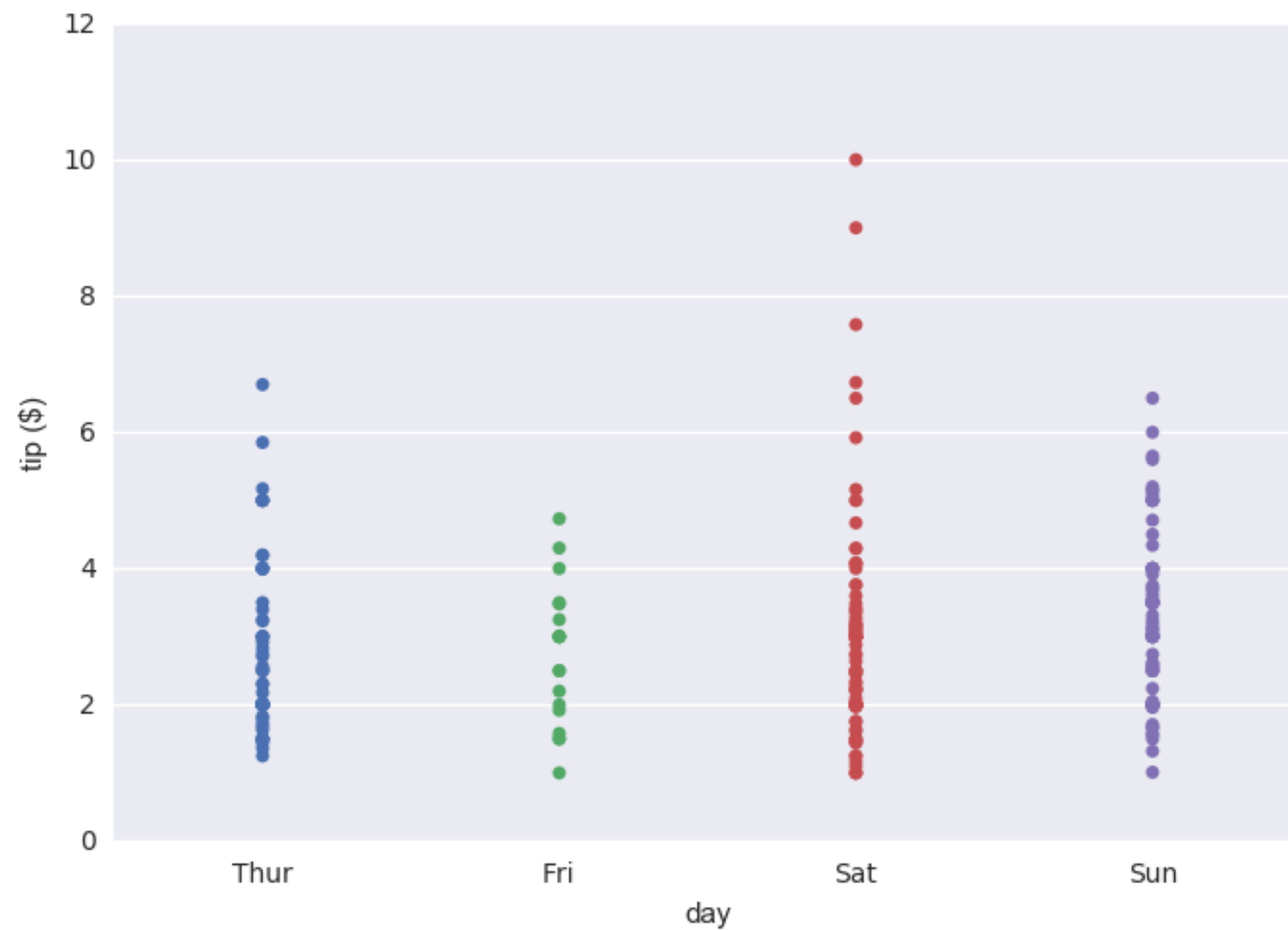
```
In [4]: sns.stripplot(x='day', y='tip', data=tip)
```

```
In [5]: plt.ylabel('tip ($)')
```

```
In [6]: plt.show()
```

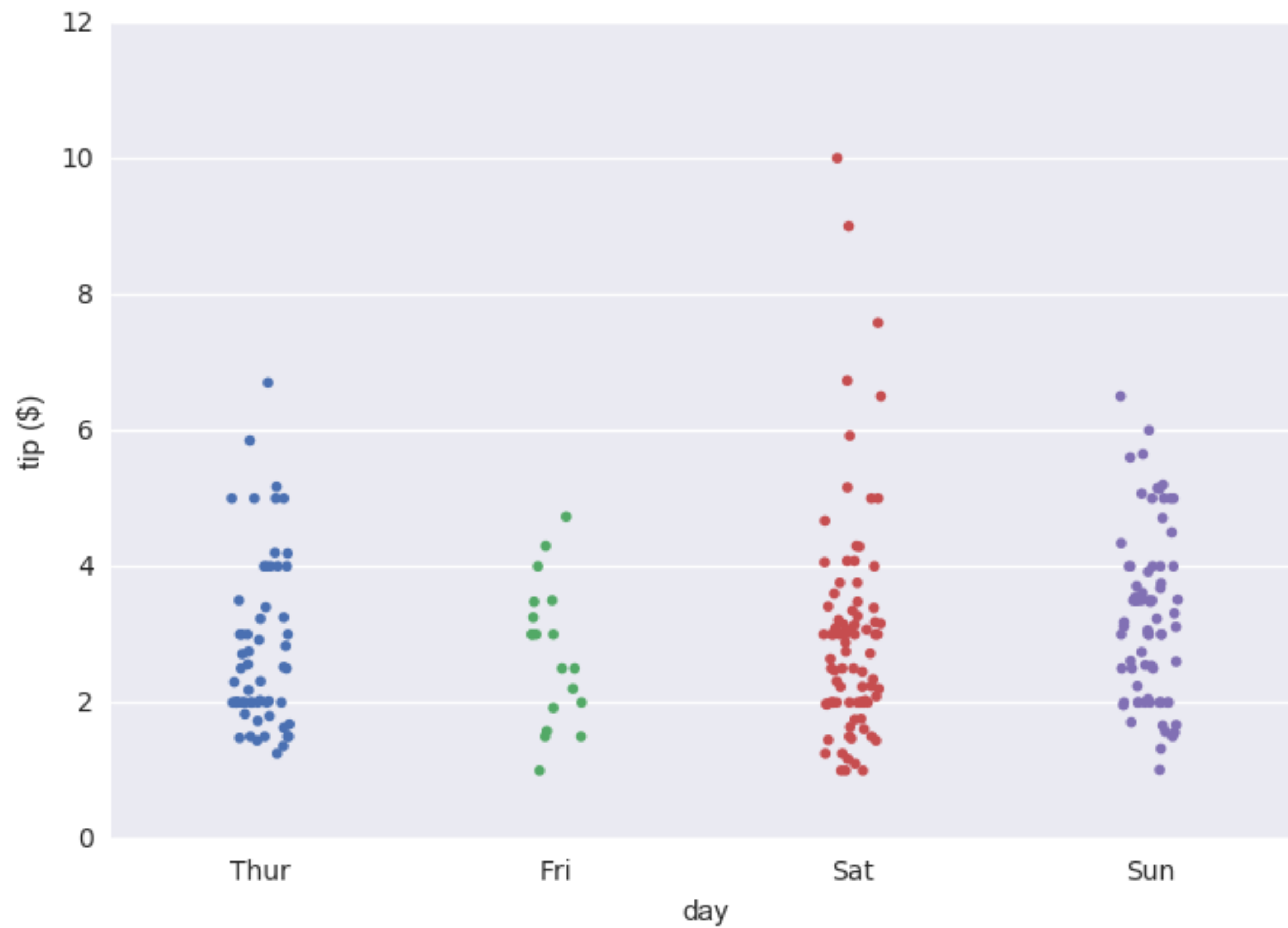


# Grouped strip plot





# Spreading out strip plots





# Spreading out strip plots

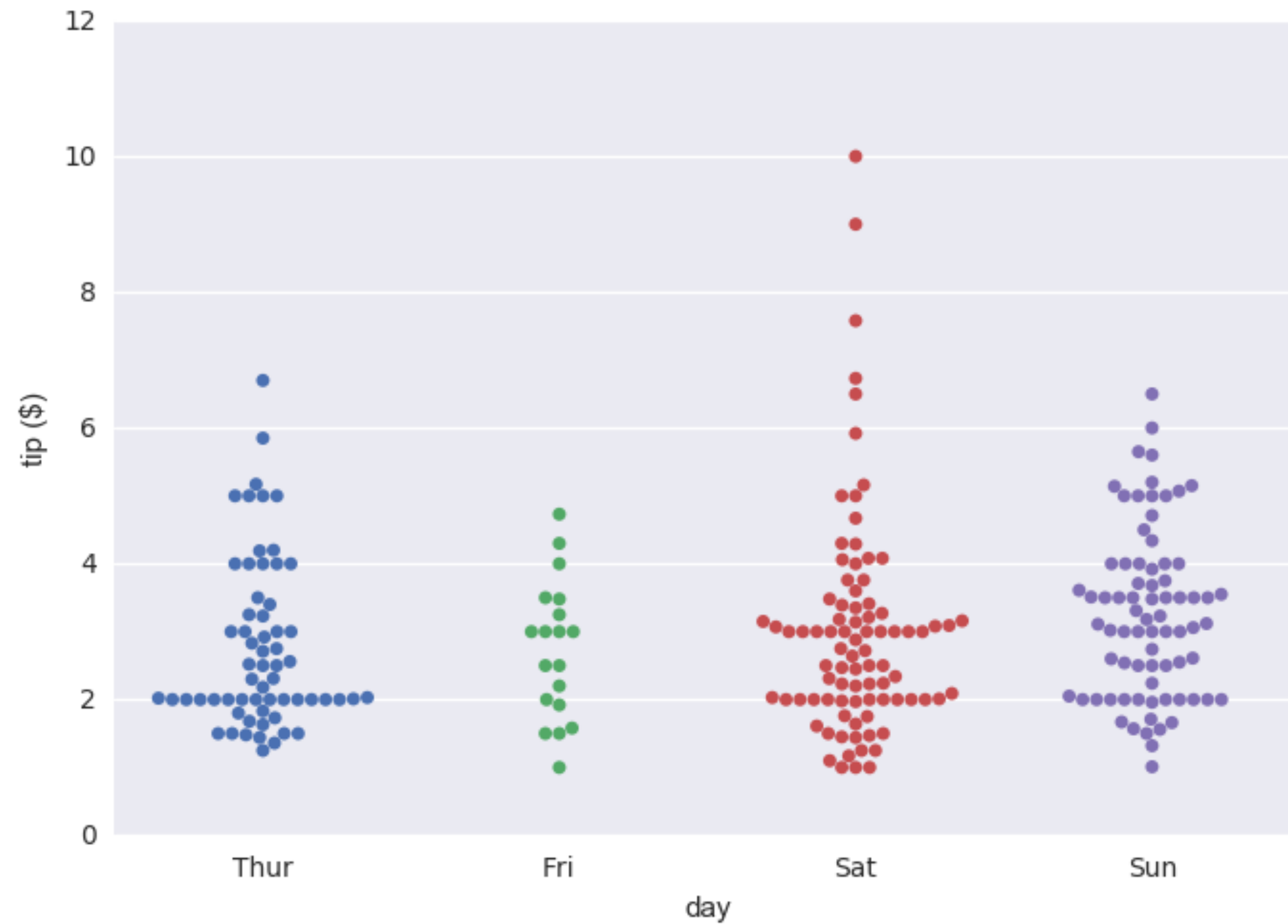
```
In [7]: sns.stripplot(x='day', y='tip', data=tip, size=4,  
    ....:              jitter=True)
```

```
In [8]: plt.ylabel('tip ($)')
```

```
In [9]: plt.show()
```



# Swarm plot





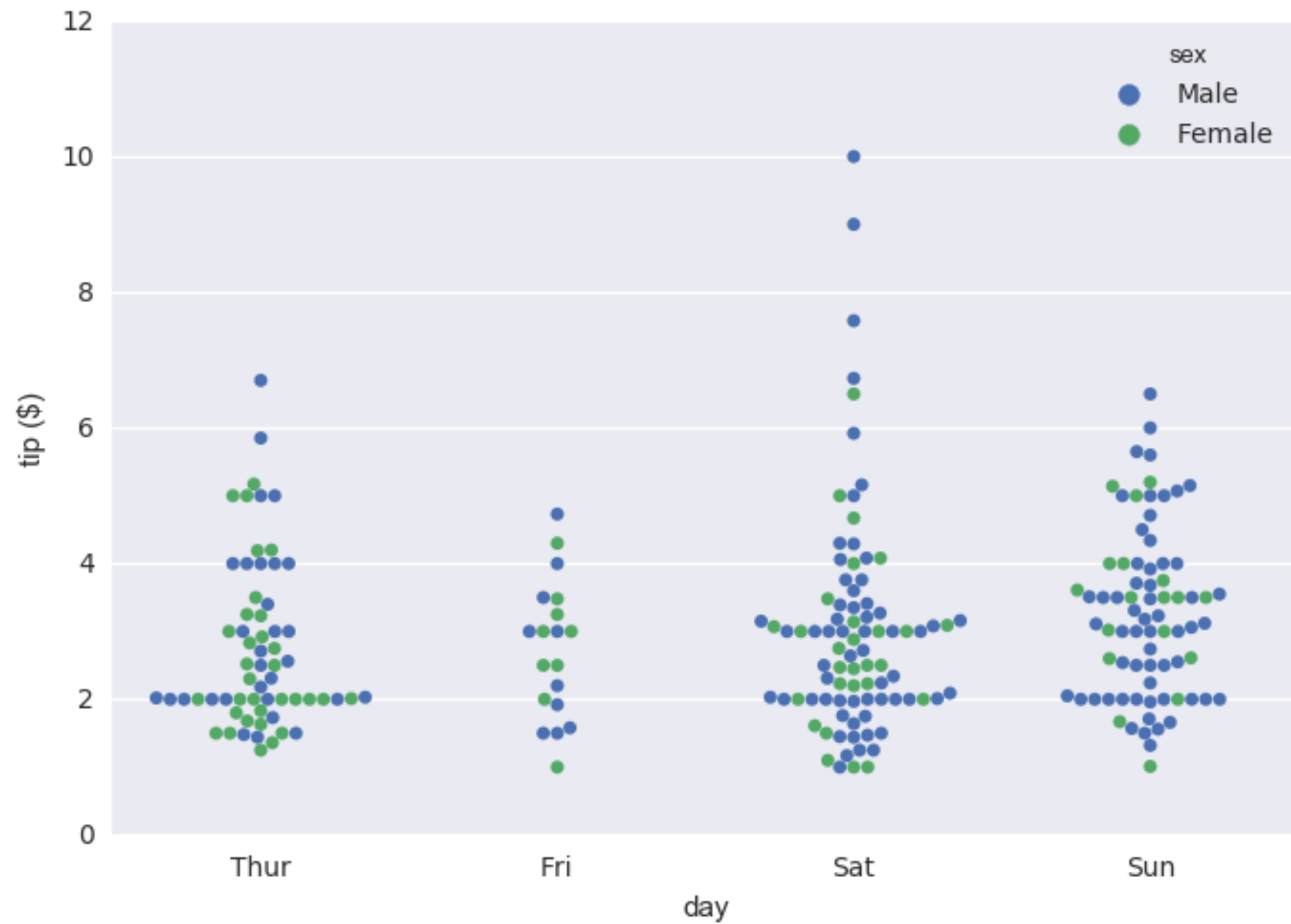
# Using `swarmplot()`

```
In [10]: sns.swarmplot(x='day', y='tip', data=tips)
```

```
In [11]: plt.ylabel('tip ($)')
```

```
In [12]: plt.show()
```







# More grouping with `swarmplot()`

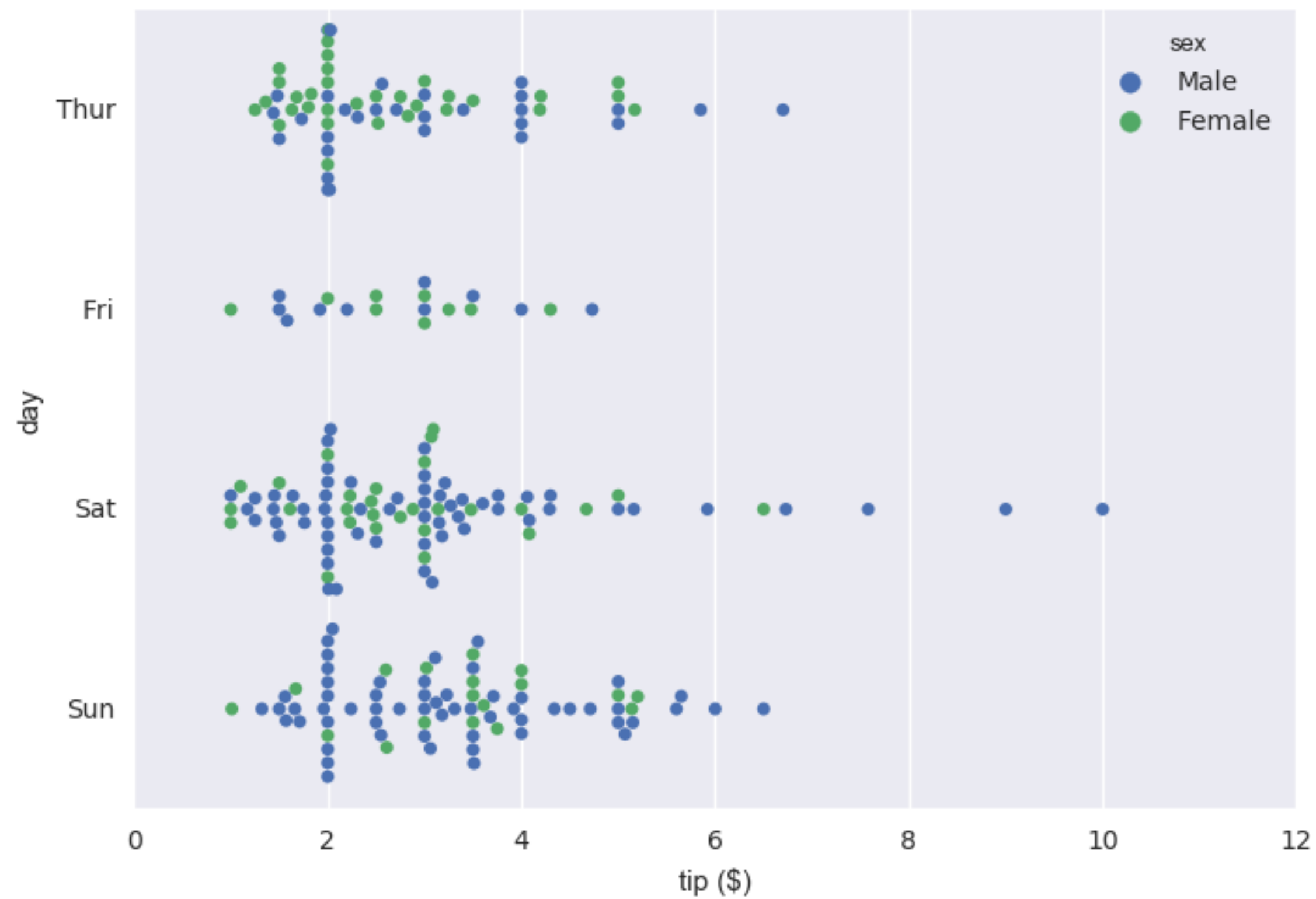
```
In [13]: sns.swarmplot(x='day', y='tip', data=tips, hue='sex')
```

```
In [14]: plt.ylabel('tip ($)')
```

```
In [15]: plt.show()
```



# Changing orientation





# Changing orientation

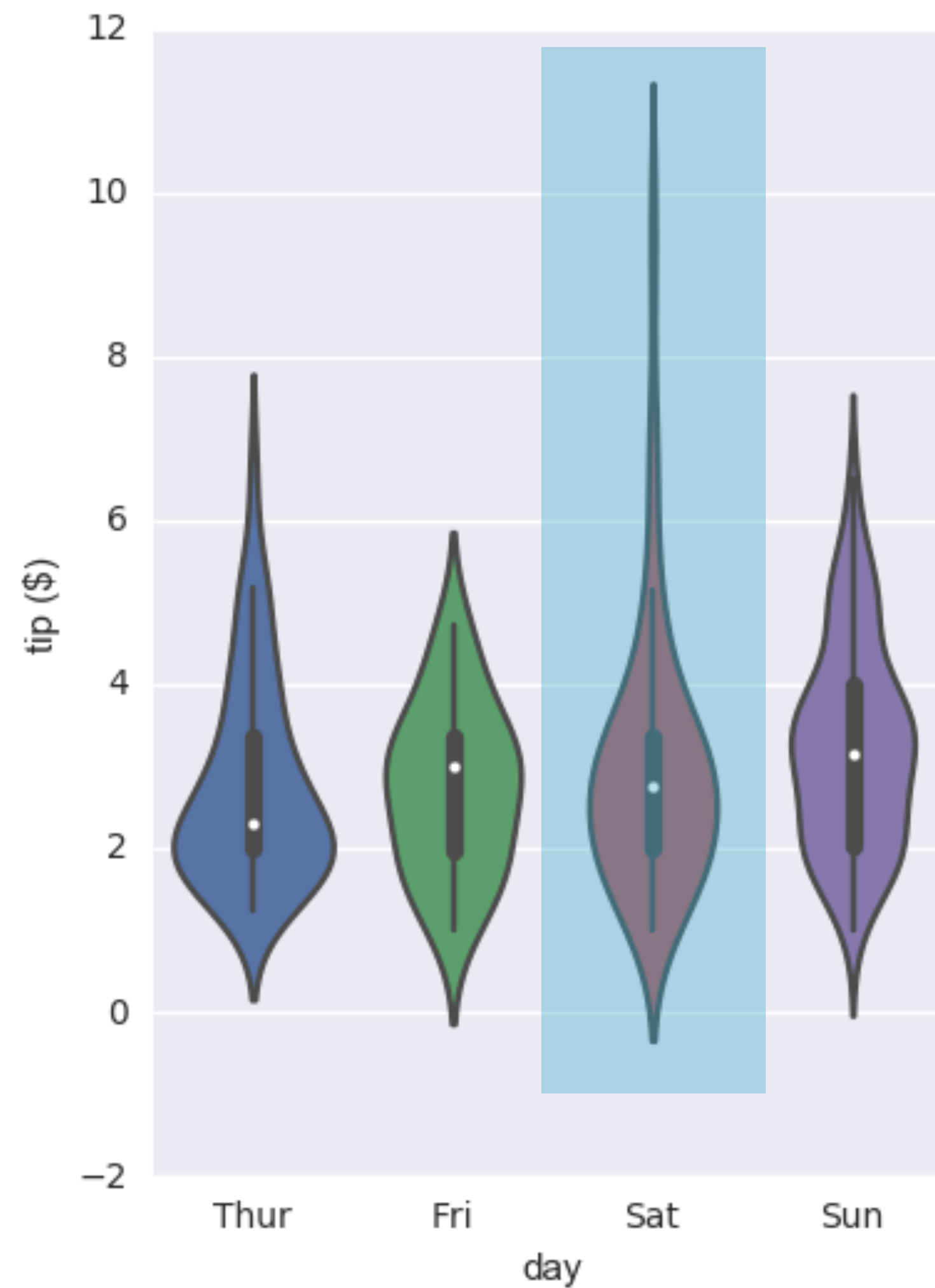
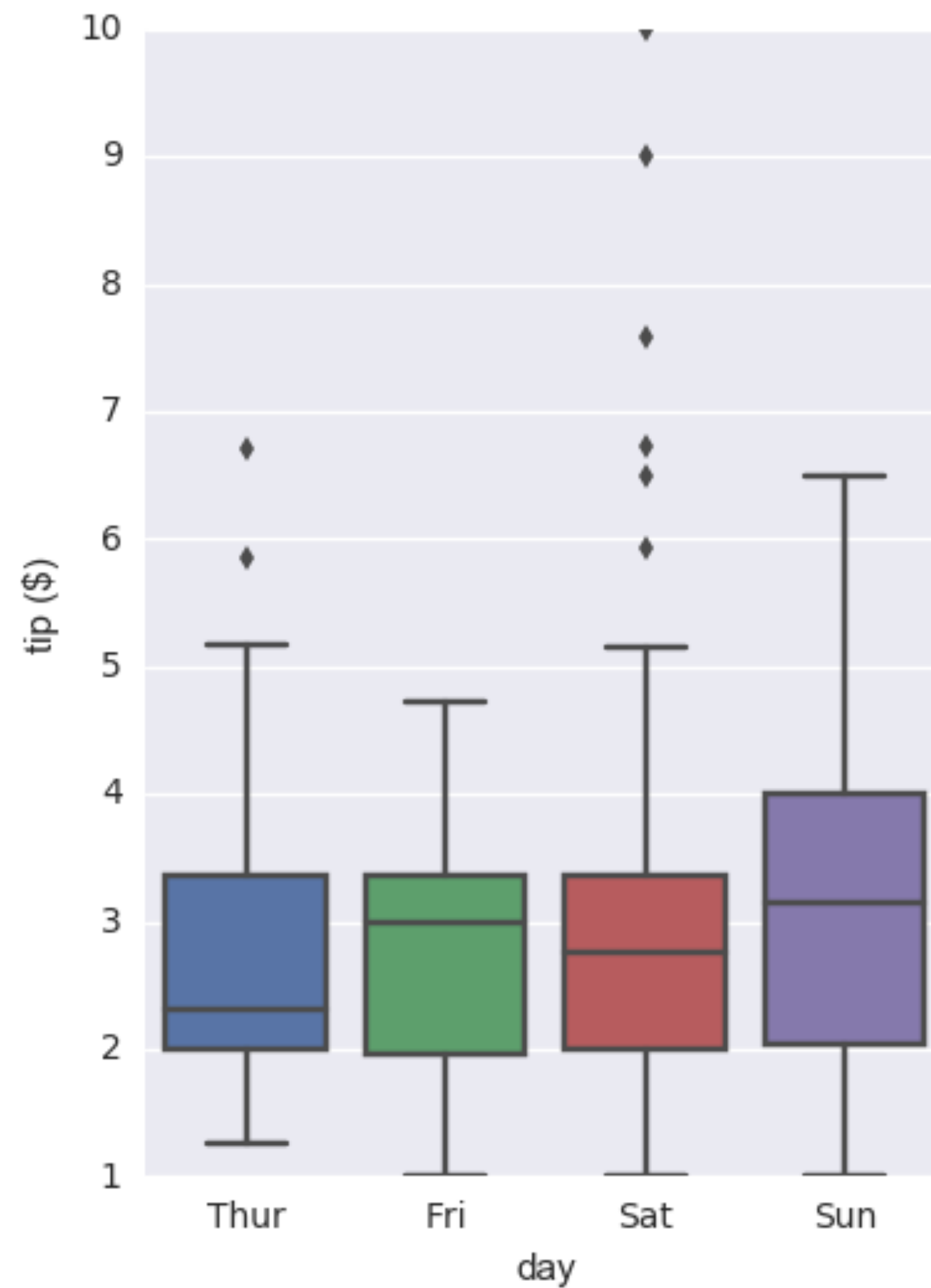
```
In [16]: sns.swarmplot(x='tip', y='day', data=tips, hue='sex',  
....:                  orient='h')
```

```
In [17]: plt.xlabel('tip ($)')
```

```
In [18]: plt.show()
```



# Violin plot





# Using violinplot()

```
In [19]: plt.subplot(1,2,1)
```

```
In [20]: sns.boxplot(x='day', y='tip', data=tips)
```

```
In [21]: plt.ylabel('tip ($)')
```

```
In [22]: plt.subplot(1,2,2)
```

```
In [23]: sns.violinplot(x='day', y='tip', data=tips)
```

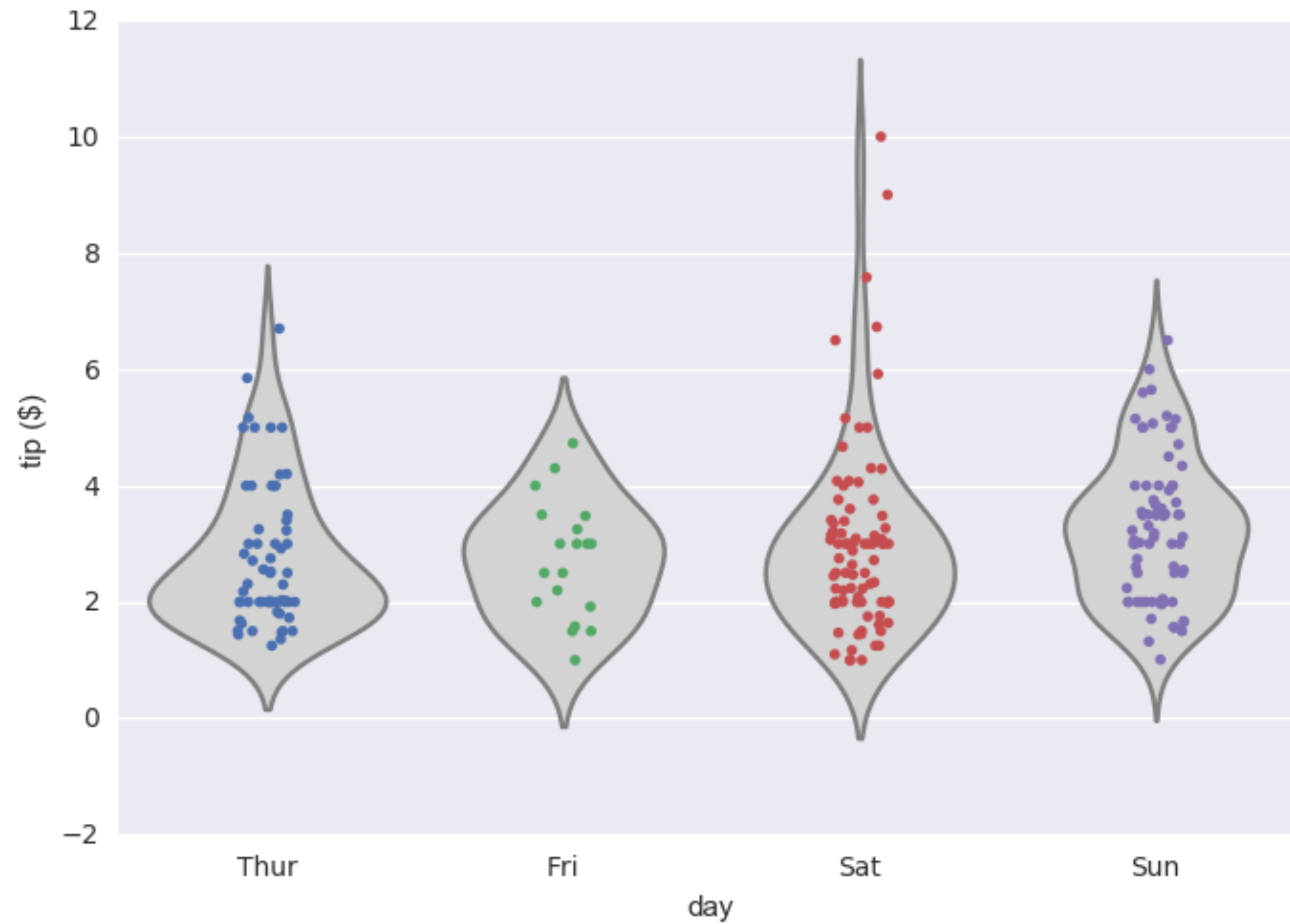
```
In [24]: plt.ylabel('tip ($)')
```

```
In [25]: plt.tight_layout()
```

```
In [26]: plt.show()
```



# Combining plots





# Combining plots

```
In [27]: sns.violinplot(x='day', y='tip', data=tips, inner=None,  
....:                  color='lightgray')
```

```
In [28]: sns.stripplot(x='day', y='tip', data=tips, size=4,  
....:                  jitter=True)
```

```
In [29]: plt.ylabel('tip ($)')
```

```
In [30]: plt.show()
```





INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

**Let's practice!**



INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

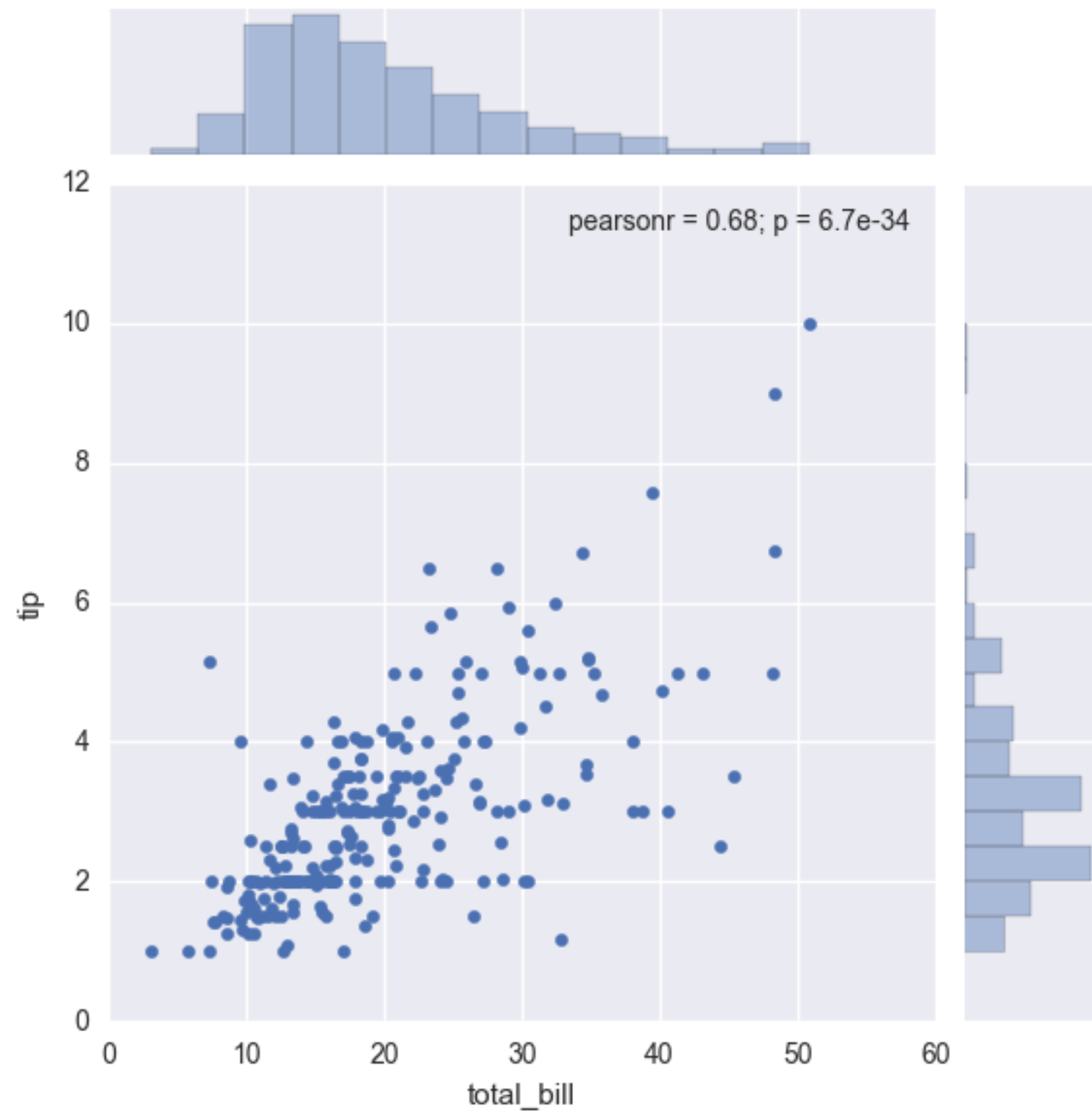
# **Visualizing Multivariate Distributions**

# Visualizing data

- Bivariate → “two variables”
- Multivariate → “multiple variables”
- Visualizing relationships in multivariate data
  - Joint plots
  - Pair plots
  - Heat maps



# Joint plot





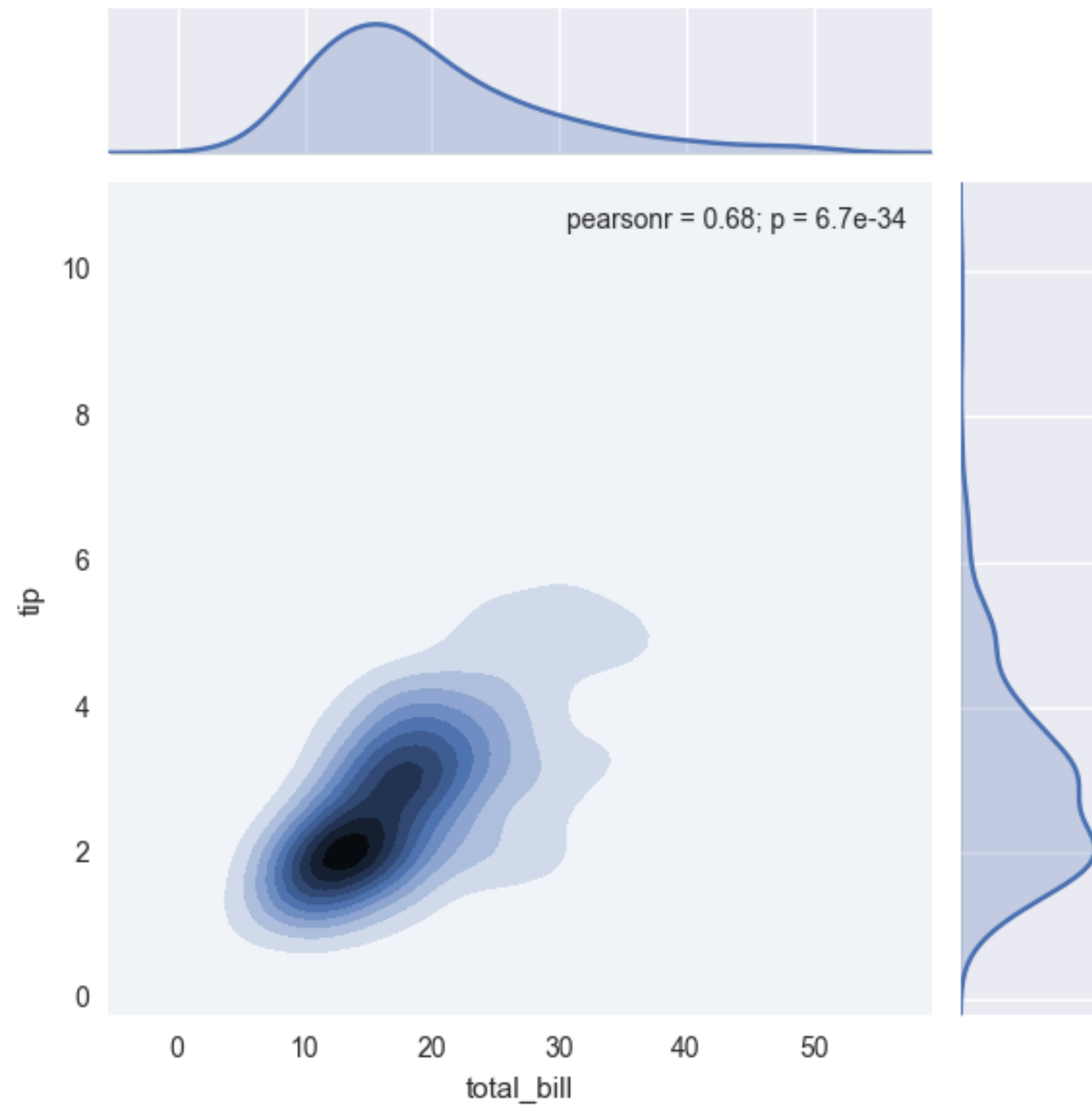
# Using jointplot()

```
In [1]: sns.jointplot(x= 'total_bill', y= 'tip', data=tips)
```

```
In [2]: plt.show()
```



# Joint plot using KDE



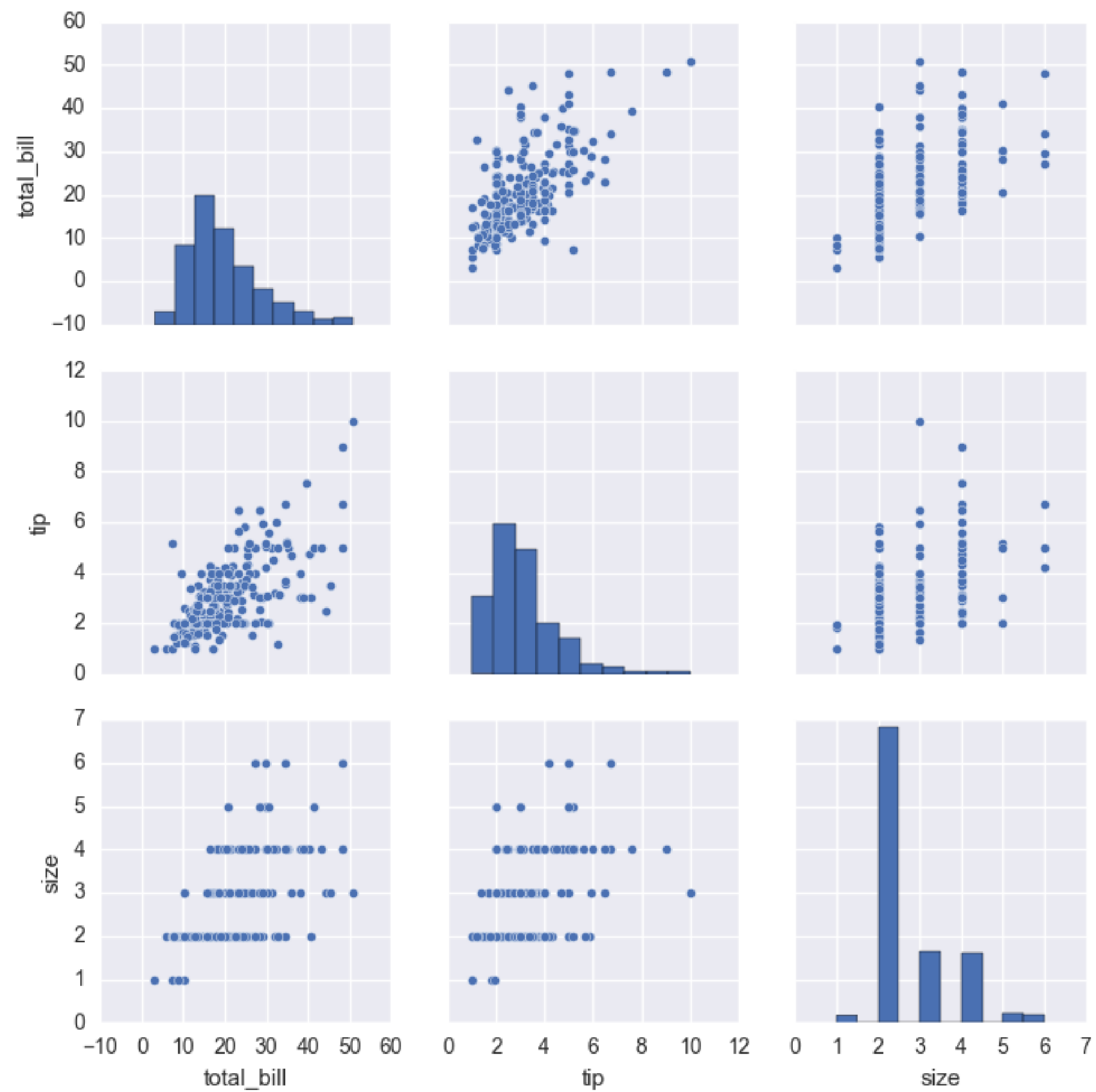


# Using kde=True

```
In [3]: sns.jointplot(x='total_bill', y='tip', data=tips,  
    ....:              kind='kde')  
  
In [4]: plt.show()
```



# Pair plot





# Using pairplot()

```
In [5]: sns.pairplot(tips)
```

```
In [6]: plt.show()
```



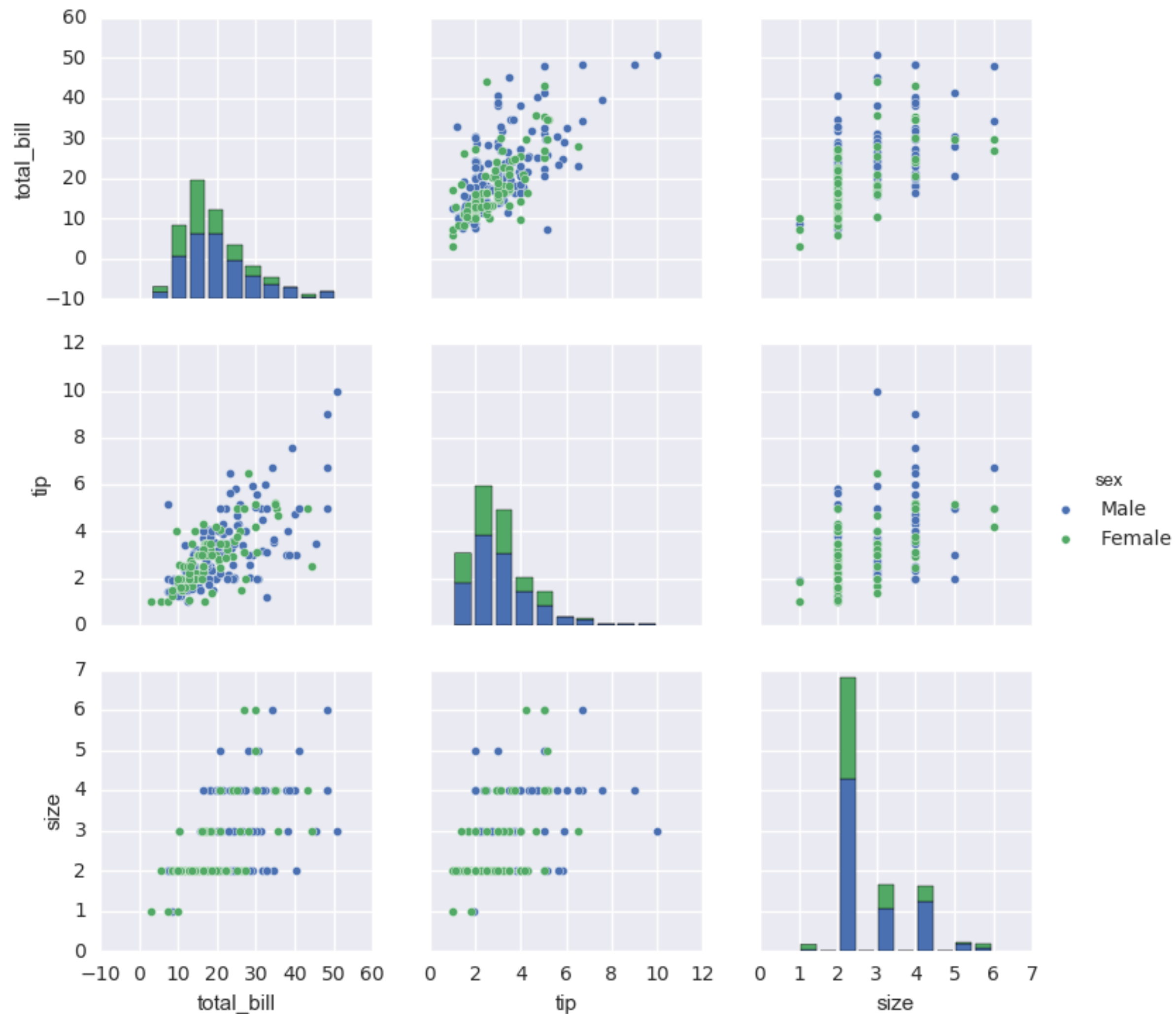
# Using pairplot() with hue

```
In [7]: sns.pairplot(tips, hue='sex')
```

```
In [8]: plt.show()
```

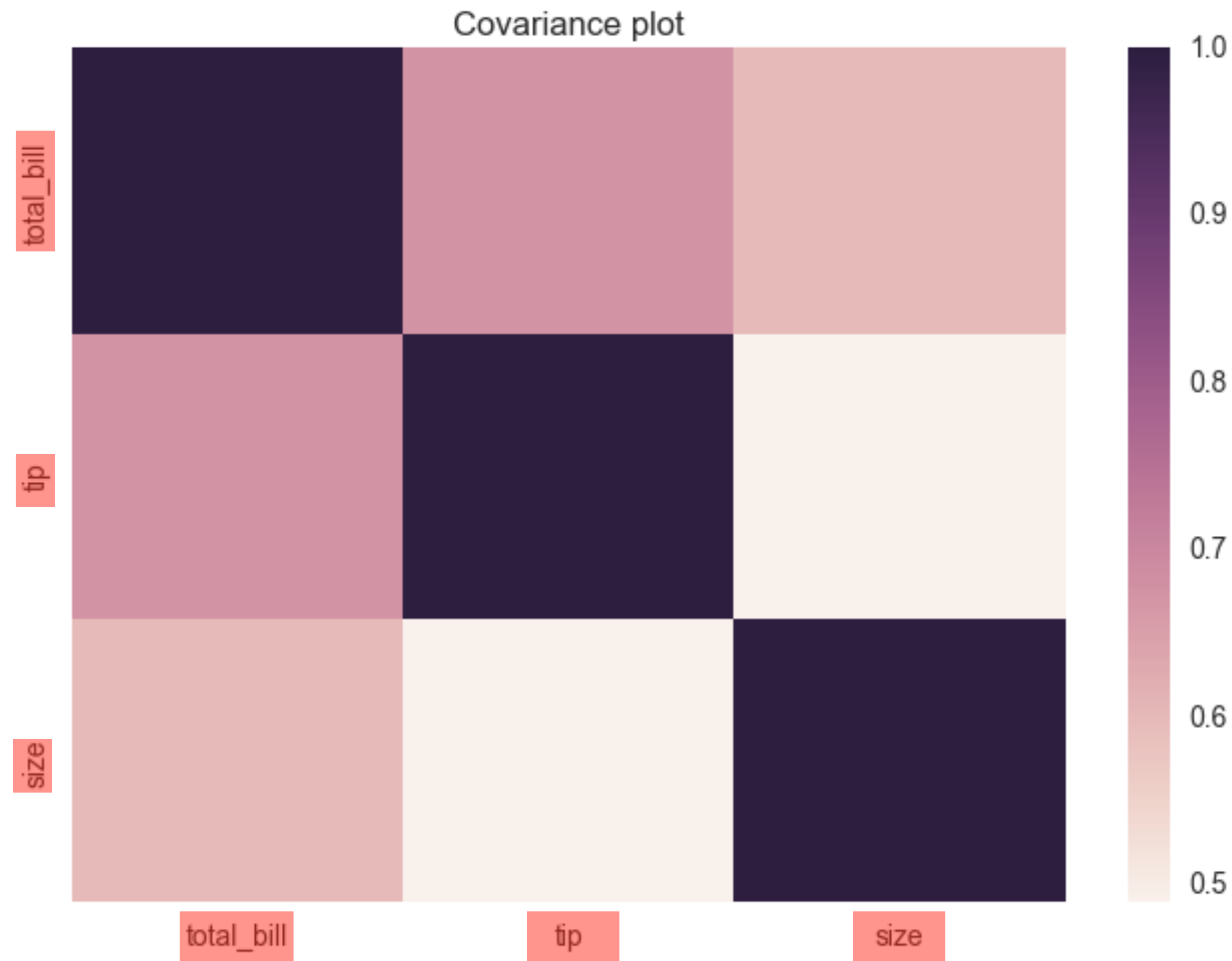


# Using pairplot() with hue





# Covariance heat map of tips data





# Using heatmap()

```
In [9]: print(covariance)
```

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

```
In [10]: sns.heatmap(covariance)
```

```
In [11]: plt.title('Covariance plot')
```

```
In [12]: plt.show()
```



INTRODUCTION TO DATA VISUALIZATION WITH PYTHON

**Let's practice!**