

seaborn

August 11, 2024

1 Seaborn

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

- Provides a layer of abstraction hence simpler to use
- better aesthetics
- more graphs included

1.0.1 Main Components of Seaborn

Types of Functions 1. Figure Level 2. Axis Level

Main Classification - Relational Plot - Distribution Plot - Categorical Plot - Regression Plot - Matrix Plot - Multiplots

```
[80]: import seaborn as sns
import numpy as np
import plotly.express as px
```

1.1 1. Relational Plot

Generally used for - see the statistical relation between 2 or more variables. - Bivariate Analysis

Figure level - relplot()

Axis level - 1. scatterplot() 2. lineplot()

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

```
[2]:      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
```

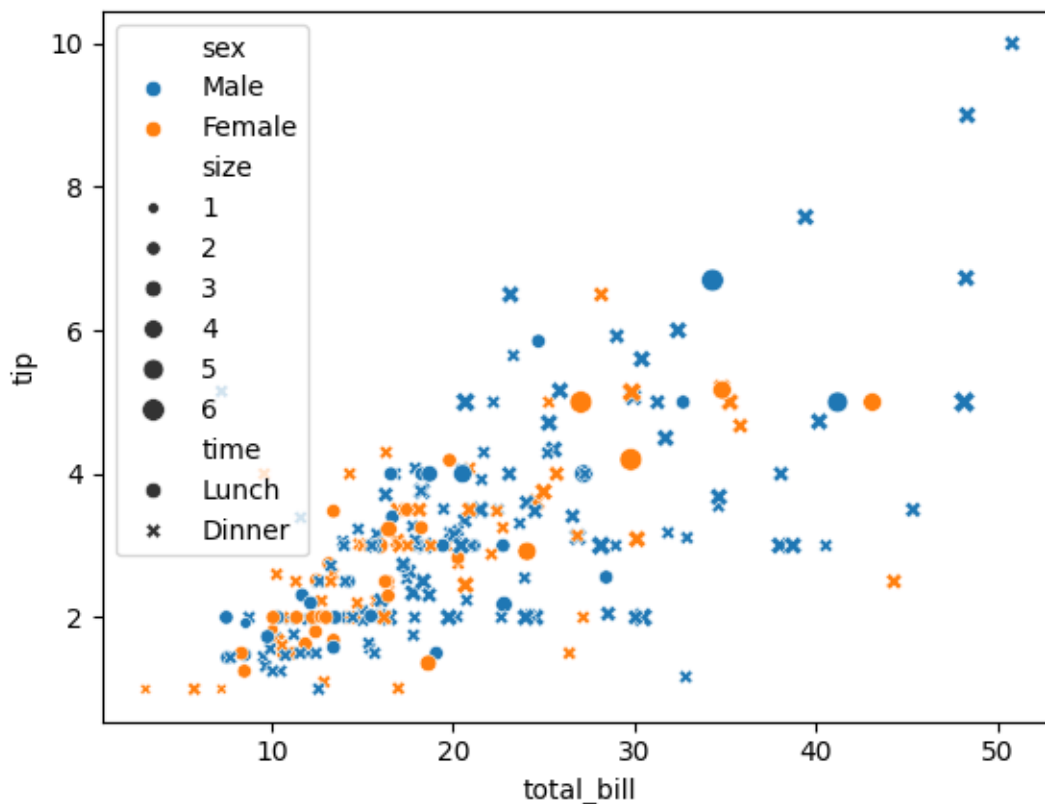
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

[244 rows x 7 columns]

1.1.1 sns.scatterplot(data, x, y, hue, style, size)

```
[6]: sns.scatterplot(data=tips, x='total_bill', y='tip', hue='sex', style='time',
    ↪size='size')
```

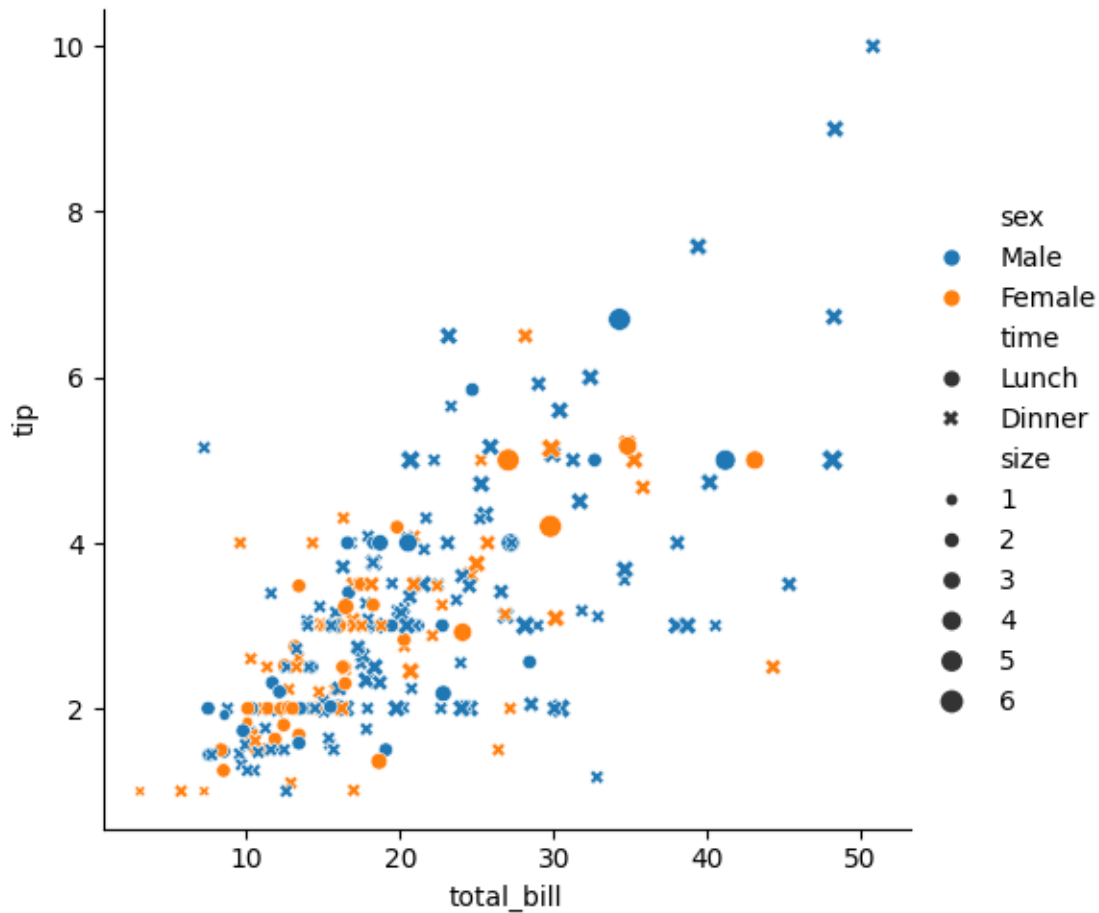
```
[6]: <Axes: xlabel='total_bill', ylabel='tip'>
```



sns.relplot(data, x, y, kind, hue, style, size)

```
[8]: sns.relplot(data=tips, x='total_bill', y='tip', kind='scatter', hue='sex',
    ↪style='time', size='size')
```

```
[8]: <seaborn.axisgrid.FacetGrid at 0x250fe9cc980>
```



1.1.2 sns.lineplot(data, x, y)

```
[11]: gap = px.data.gapminder()
temp = gap[gap['country'] == 'India']
temp
```

```
[11]:
```

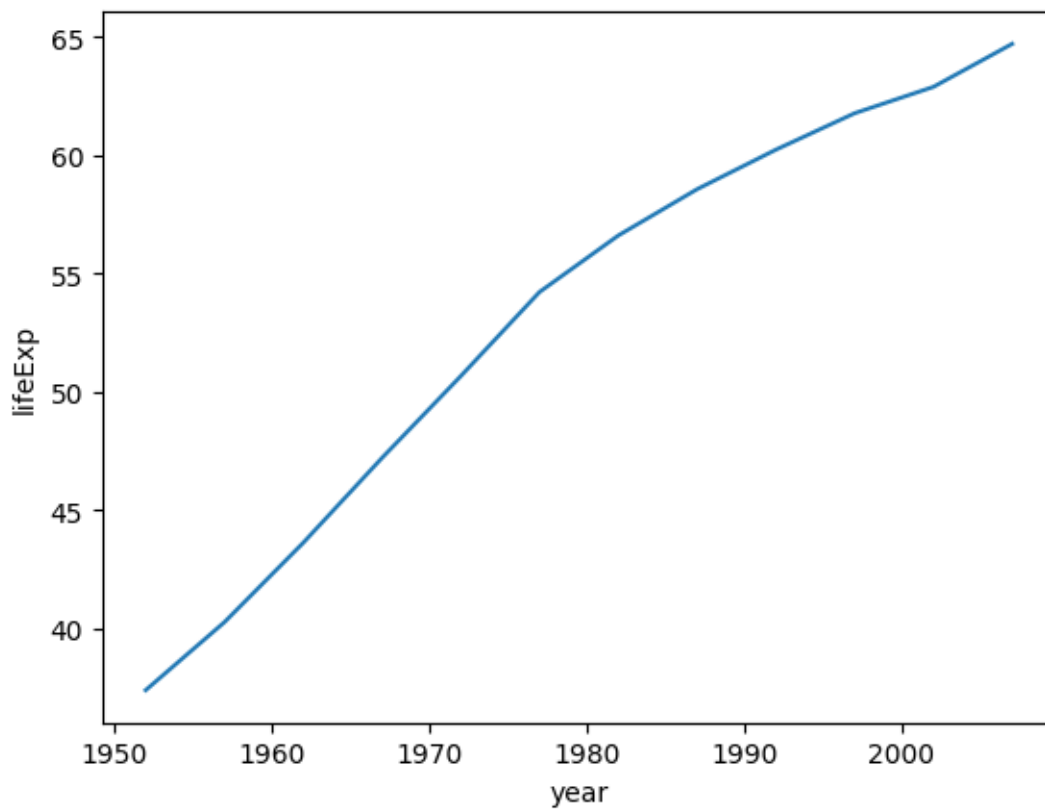
	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	\
696	India	Asia	1952	37.373	372000000	546.565749	IND	
697	India	Asia	1957	40.249	409000000	590.061996	IND	
698	India	Asia	1962	43.605	454000000	658.347151	IND	
699	India	Asia	1967	47.193	506000000	700.770611	IND	
700	India	Asia	1972	50.651	567000000	724.032527	IND	
701	India	Asia	1977	54.208	634000000	813.337323	IND	
702	India	Asia	1982	56.596	708000000	855.723538	IND	
703	India	Asia	1987	58.553	788000000	976.512676	IND	
704	India	Asia	1992	60.223	872000000	1164.406809	IND	
705	India	Asia	1997	61.765	959000000	1458.817442	IND	
706	India	Asia	2002	62.879	1034172547	1746.769454	IND	

707	India	Asia	2007	64.698	1110396331	2452.210407	IND
-----	-------	------	------	--------	------------	-------------	-----

	iso_num
696	356
697	356
698	356
699	356
700	356
701	356
702	356
703	356
704	356
705	356
706	356
707	356

```
[13]: sns.lineplot(data=temp, x='year', y='lifeExp')
```

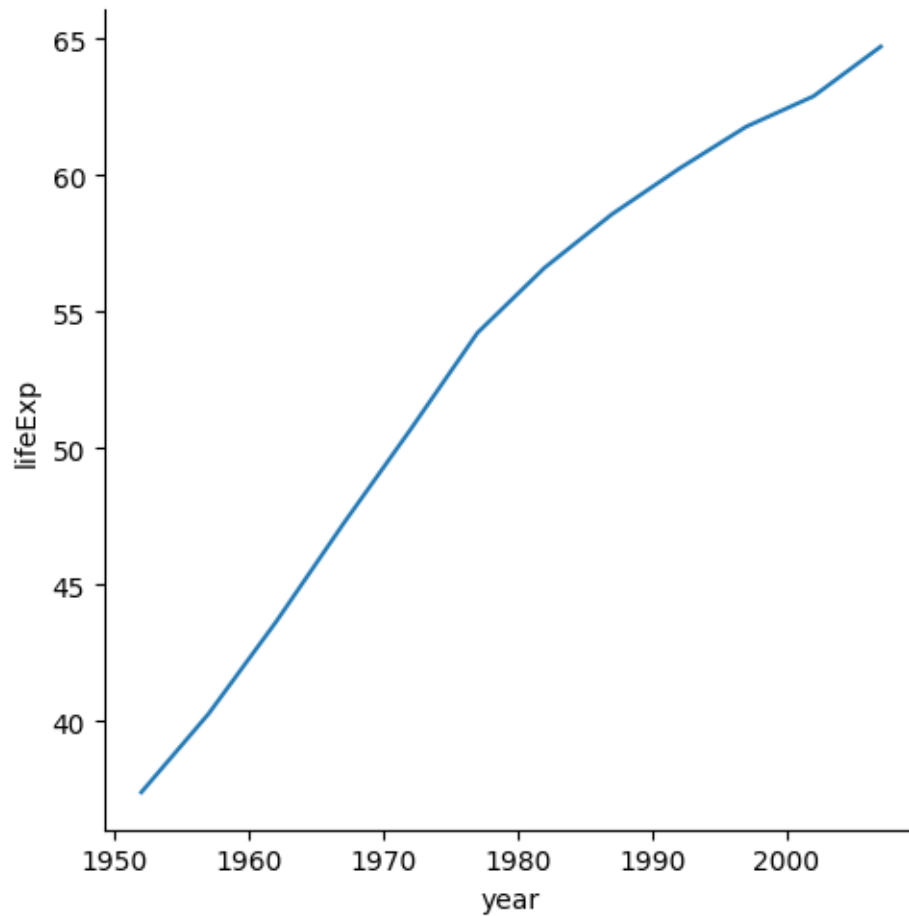
```
[13]: <Axes: xlabel='year', ylabel='lifeExp'>
```



```
sns.relplot(data, x, y, kind, hue, style, size)
```

```
[14]: sns.relplot(data=temp, x='year', y='lifeExp', kind='line')
```

```
[14]: <seaborn.axisgrid.FacetGrid at 0x250fe821d90>
```



```
[17]: temp = gap[gap['country'].isin(['India', 'Brazil', 'Germany'])]
temp
```

```
[17]:
```

	country	continent	year	lifeExp	pop	gdpPercap	iso_alpha	\
168	Brazil	Americas	1952	50.917	56602560	2108.944355	BRA	
169	Brazil	Americas	1957	53.285	65551171	2487.365989	BRA	
170	Brazil	Americas	1962	55.665	76039390	3336.585802	BRA	
171	Brazil	Americas	1967	57.632	88049823	3429.864357	BRA	
172	Brazil	Americas	1972	59.504	100840058	4985.711467	BRA	
173	Brazil	Americas	1977	61.489	114313951	6660.118654	BRA	
174	Brazil	Americas	1982	63.336	128962939	7030.835878	BRA	
175	Brazil	Americas	1987	65.205	142938076	7807.095818	BRA	
176	Brazil	Americas	1992	67.057	155975974	6950.283021	BRA	
177	Brazil	Americas	1997	69.388	168546719	7957.980824	BRA	

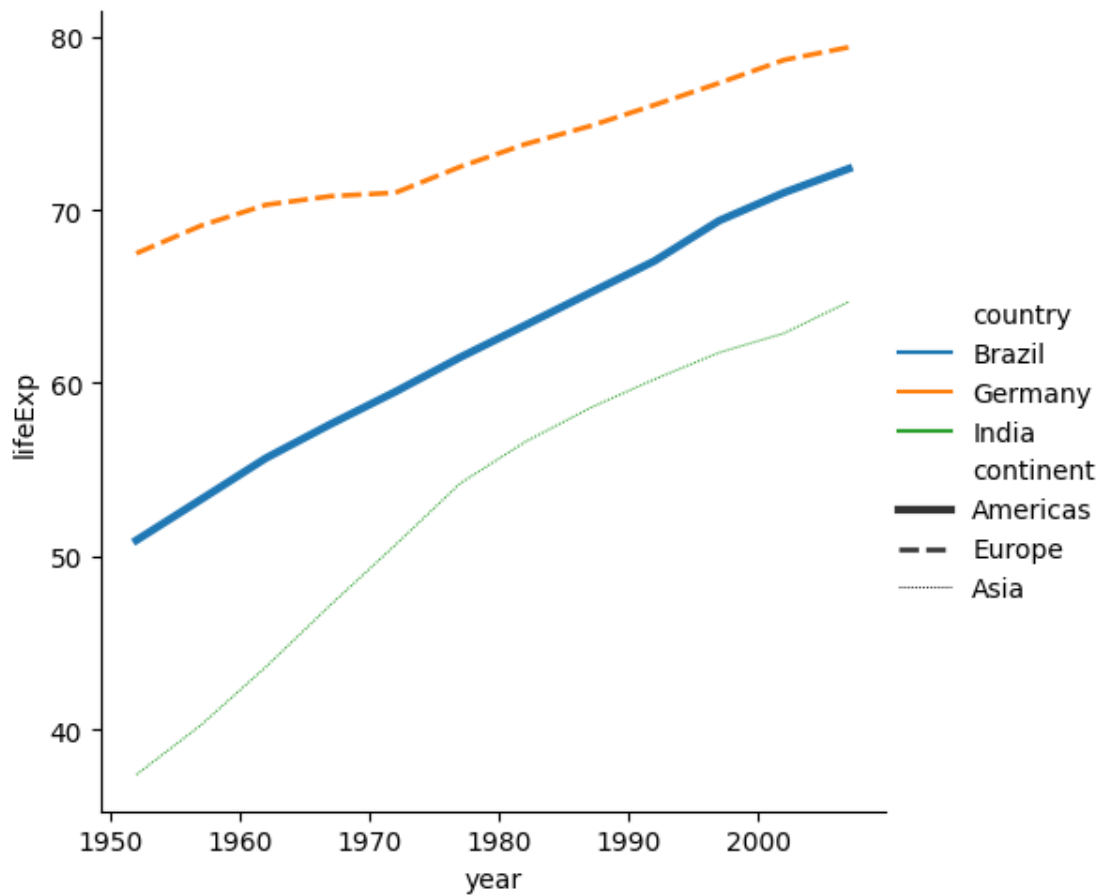
178	Brazil	Americas	2002	71.006	179914212	8131.212843	BRA
179	Brazil	Americas	2007	72.390	190010647	9065.800825	BRA
564	Germany	Europe	1952	67.500	69145952	7144.114393	DEU
565	Germany	Europe	1957	69.100	71019069	10187.826650	DEU
566	Germany	Europe	1962	70.300	73739117	12902.462910	DEU
567	Germany	Europe	1967	70.800	76368453	14745.625610	DEU
568	Germany	Europe	1972	71.000	78717088	18016.180270	DEU
569	Germany	Europe	1977	72.500	78160773	20512.921230	DEU
570	Germany	Europe	1982	73.800	78335266	22031.532740	DEU
571	Germany	Europe	1987	74.847	77718298	24639.185660	DEU
572	Germany	Europe	1992	76.070	80597764	26505.303170	DEU
573	Germany	Europe	1997	77.340	82011073	27788.884160	DEU
574	Germany	Europe	2002	78.670	82350671	30035.801980	DEU
575	Germany	Europe	2007	79.406	82400996	32170.374420	DEU
696	India	Asia	1952	37.373	372000000	546.565749	IND
697	India	Asia	1957	40.249	409000000	590.061996	IND
698	India	Asia	1962	43.605	454000000	658.347151	IND
699	India	Asia	1967	47.193	506000000	700.770611	IND
700	India	Asia	1972	50.651	567000000	724.032527	IND
701	India	Asia	1977	54.208	634000000	813.337323	IND
702	India	Asia	1982	56.596	708000000	855.723538	IND
703	India	Asia	1987	58.553	788000000	976.512676	IND
704	India	Asia	1992	60.223	872000000	1164.406809	IND
705	India	Asia	1997	61.765	959000000	1458.817442	IND
706	India	Asia	2002	62.879	1034172547	1746.769454	IND
707	India	Asia	2007	64.698	1110396331	2452.210407	IND

	iso_num
168	76
169	76
170	76
171	76
172	76
173	76
174	76
175	76
176	76
177	76
178	76
179	76
564	276
565	276
566	276
567	276
568	276
569	276
570	276

571	276
572	276
573	276
574	276
575	276
696	356
697	356
698	356
699	356
700	356
701	356
702	356
703	356
704	356
705	356
706	356
707	356

```
[21]: sns.relplot(data=temp, x='year', y='lifeExp', kind='line', hue='country',  
↳ style='continent', size='continent')
```

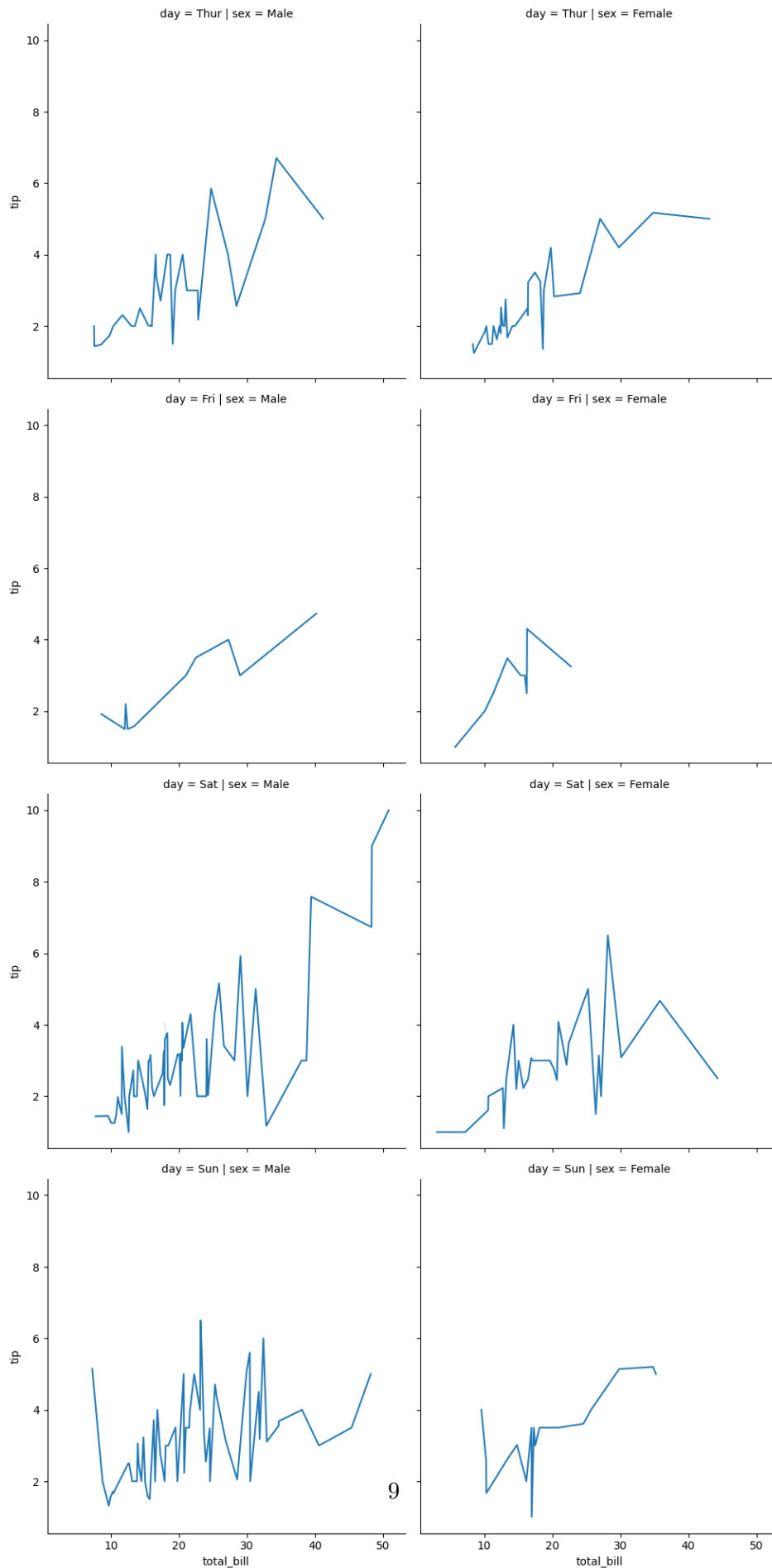
```
[21]: <seaborn.axisgrid.FacetGrid at 0x250835a25d0>
```



Facet Plot Figure level function, only work with relplot

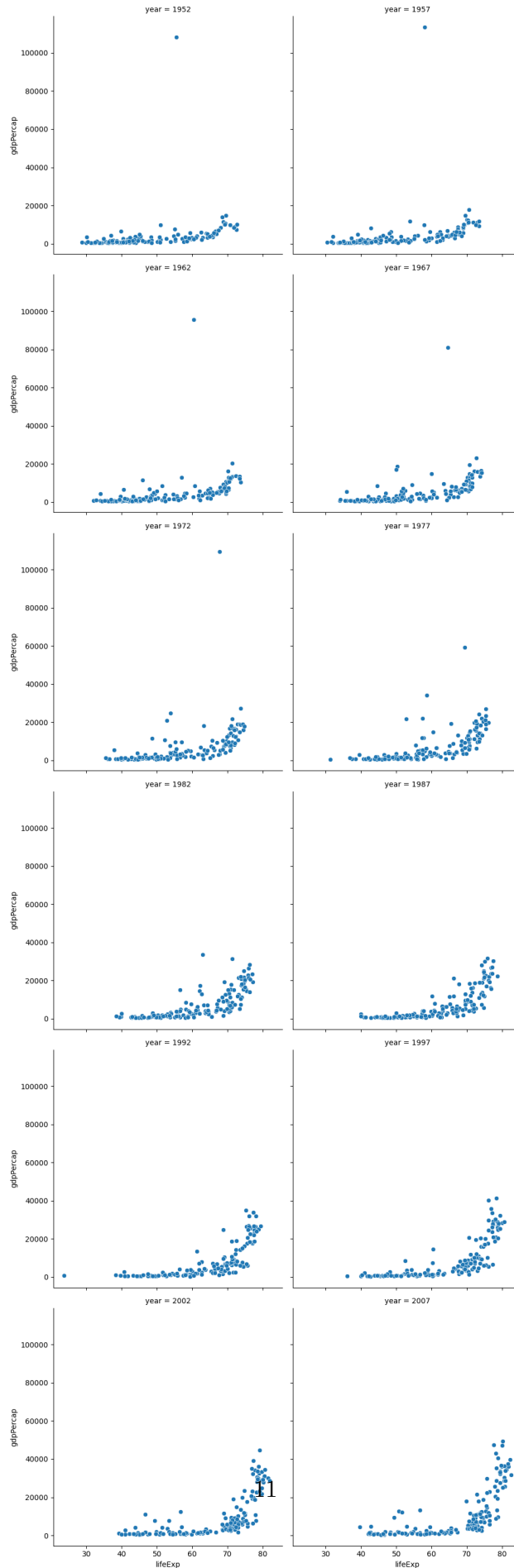
```
[22]: sns.relplot(data=tips, x='total_bill', y='tip', kind='line', col='sex',
    ↪row='day')
```

```
[22]: <seaborn.axisgrid.FacetGrid at 0x2508358b770>
```

```
[23]: # Col Wrap
sns.relplot(data=gap, x='lifeExp', y='gdpPercap', kind='scatter', col='year',
          ↳col_wrap=2)
```

```
[23]: <seaborn.axisgrid.FacetGrid at 0x25082ca3770>
```



1.2 2. Distribution Plots

Generally used for - Univariate Analysis - used to find out the distribution of data - shows the range of the data - central tendency - used to check whether there are any outliers or the data bimodal?

Figure level - `displot()`

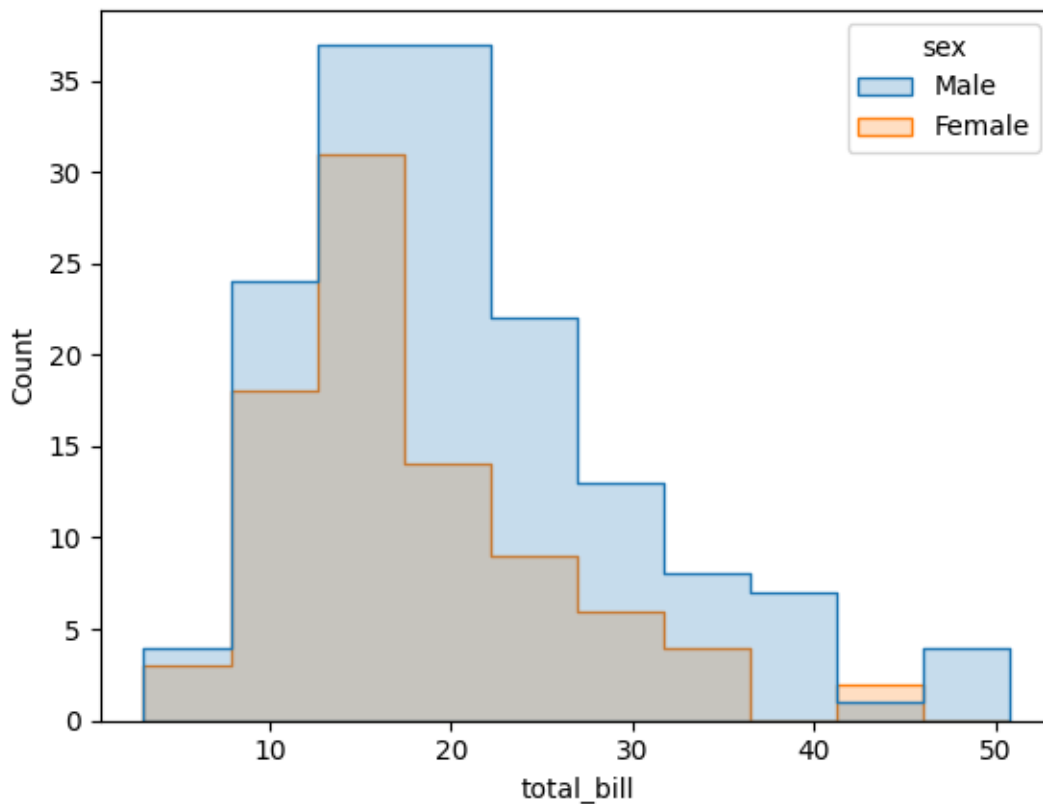
Axis level - 1. `histplot()` 2. `kdeplot()` 3. `rugplot()`

1.2.1 `sns.histplot(data, x, bins, hue, element)`

element is used to designing

```
[29]: sns.histplot(data=tips, x='total_bill', bins=10, hue='sex', element='step')
```

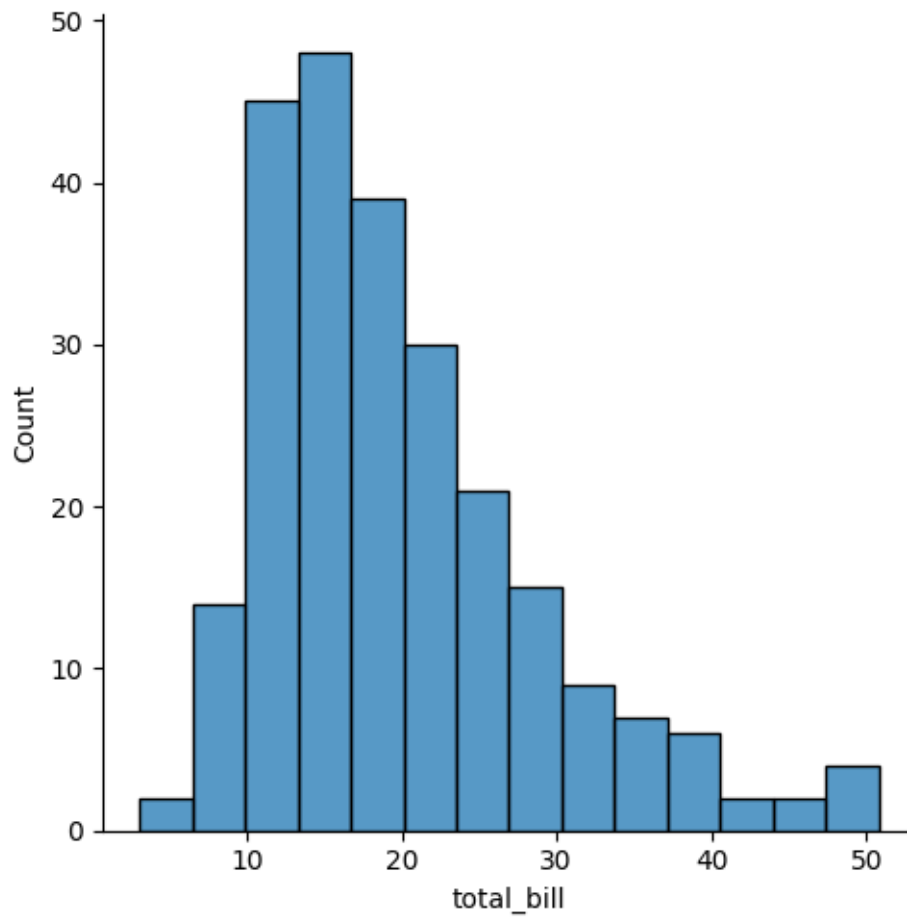
```
[29]: <Axes: xlabel='total_bill', ylabel='Count'>
```



`sns.displot(data, x, kind)`

```
[25]: sns.displot(data=tips, x='total_bill', kind='hist')
```

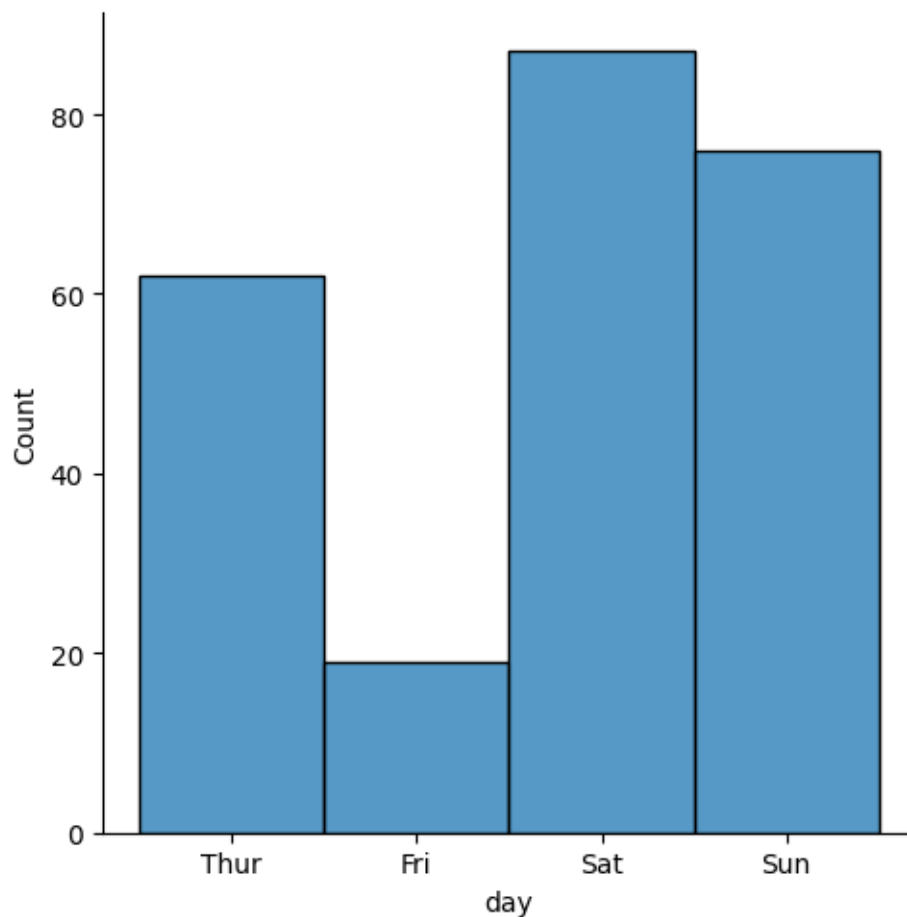
[25]: <seaborn.axisgrid.FacetGrid at 0x2508390cf50>



[27]: *# Categorical variables can also be plotted*

```
sns.displot(data=tips, x='day', kind='hist')
```

[27]: <seaborn.axisgrid.FacetGrid at 0x25085d6e390>



```
[30]: titanic = sns.load_dataset('titanic')
      titanic
```

```
[30]:
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	
..	
886	0	2	male	27.0	0	0	13.0000	S	Second	
887	1	1	female	19.0	0	0	30.0000	S	First	
888	0	3	female	NaN	1	2	23.4500	S	Third	
889	1	1	male	26.0	0	0	30.0000	C	First	
890	0	3	male	32.0	0	0	7.7500	Q	Third	

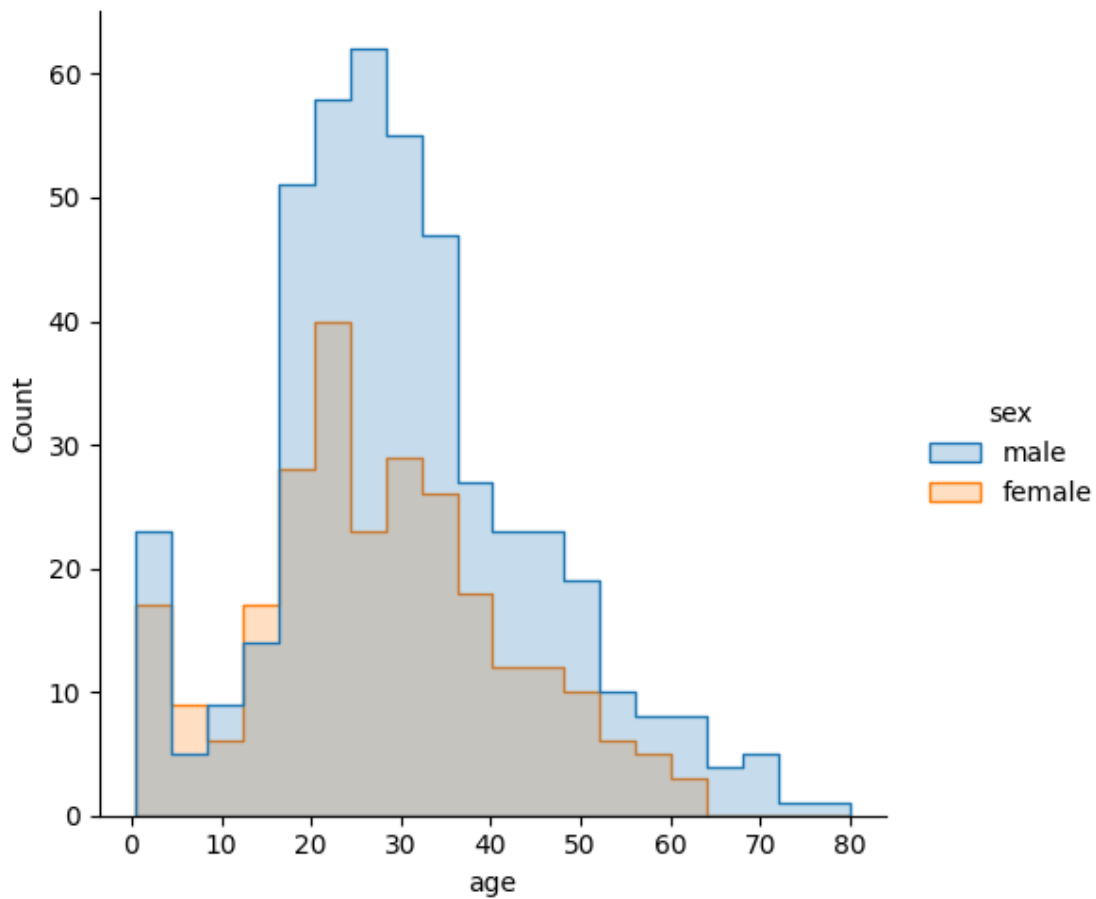
	who	adult_male	deck	embark_town	alive	alone
0	man	True	NaN	Southampton	no	False

1	woman	False	C	Cherbourg	yes	False
2	woman	False	NaN	Southampton	yes	True
3	woman	False	C	Southampton	yes	False
4	man	True	NaN	Southampton	no	True
..
886	man	True	NaN	Southampton	no	True
887	woman	False	B	Southampton	yes	True
888	woman	False	NaN	Southampton	no	False
889	man	True	C	Cherbourg	yes	True
890	man	True	NaN	Queenstown	no	True

[891 rows x 15 columns]

```
[31]: sns.displot(data=titanic, x='age', kind='hist', element='step', hue='sex')
```

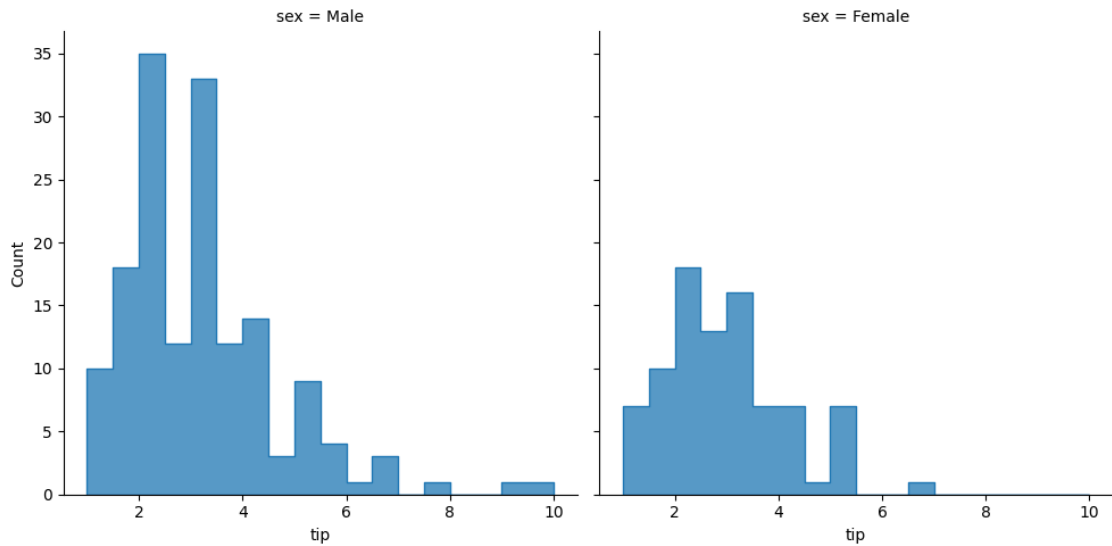
```
[31]: <seaborn.axisgrid.FacetGrid at 0x25082ca02c0>
```



Facet Plot

```
[32]: sns.displot(data=tips, x='tip', kind='hist', col='sex', element='step')
```

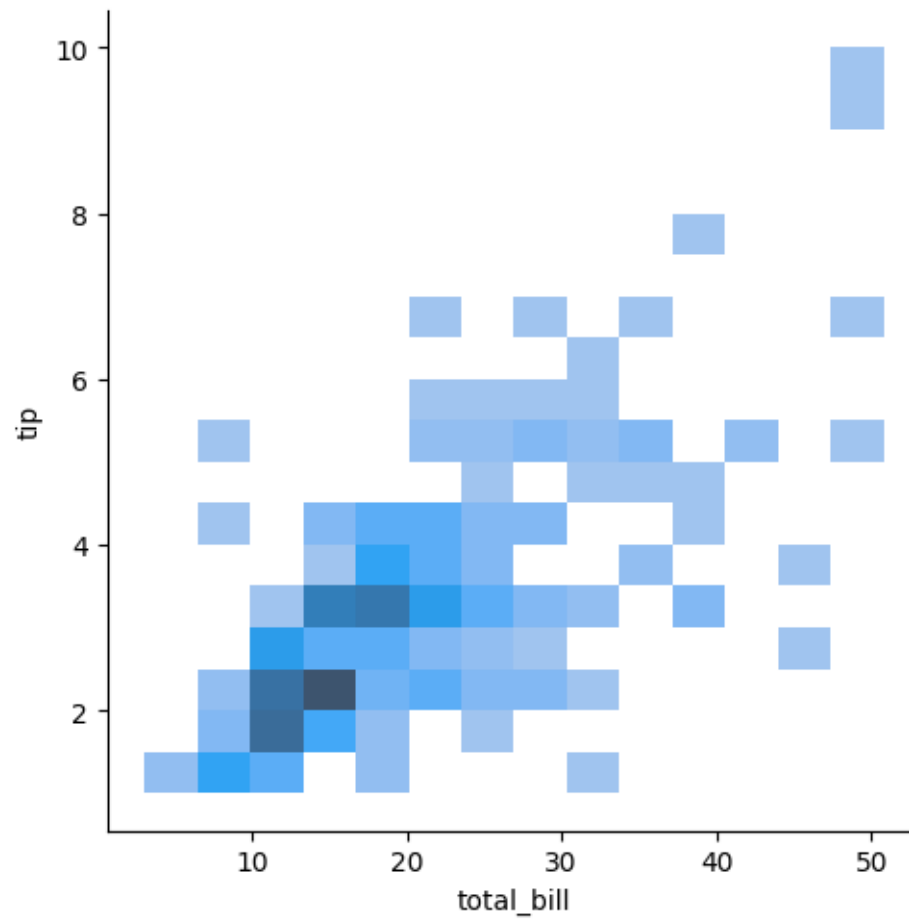
```
[32]: <seaborn.axisgrid.FacetGrid at 0x25087a84a70>
```



Bivariate Histogram A bivariate histogram bins the data within rectangles that tile the plot and then shows the count of observations within each rectangle with the fill color

```
[42]: # sns.histplot(data=tips, x='total_bill', y='tip')
sns.displot(data=tips, x='total_bill', y='tip', kind='hist')
```

```
[42]: <seaborn.axisgrid.FacetGrid at 0x25086a75580>
```

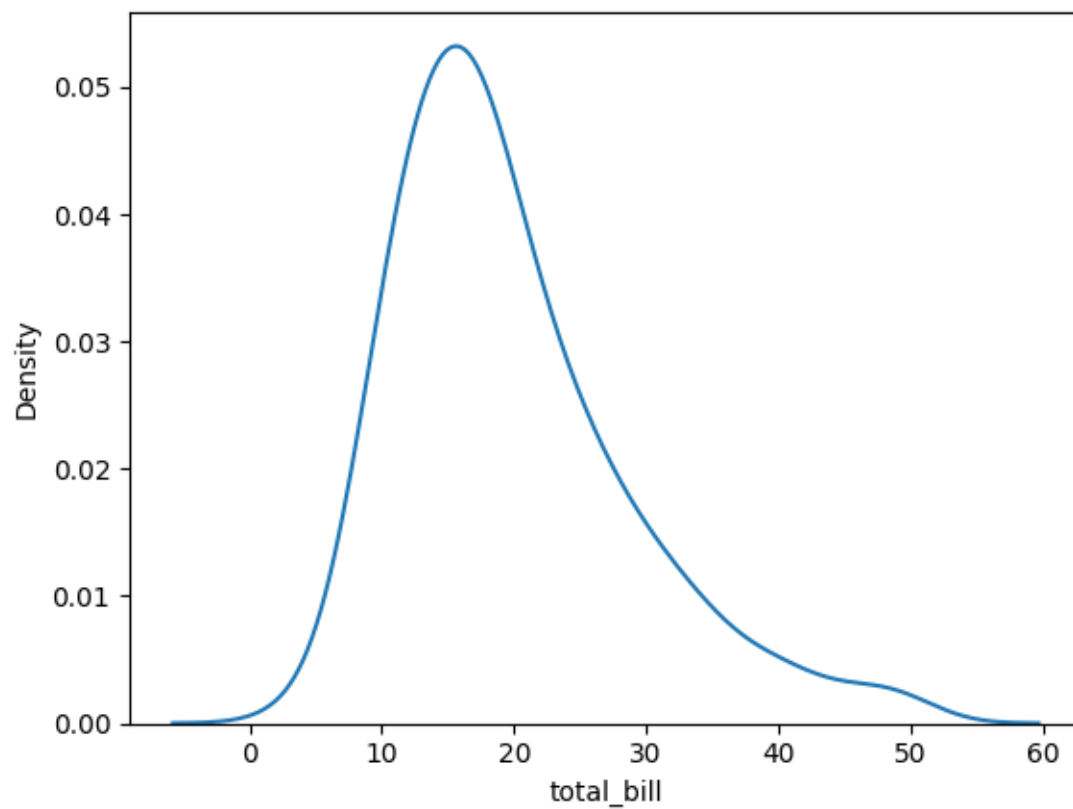



1.2.2 sns.kdeplot(data, x, hue, fill)

Rather than using discrete bins, a KDE plot smooths the observations with a Gaussian kernel, producing a continuous density estimate

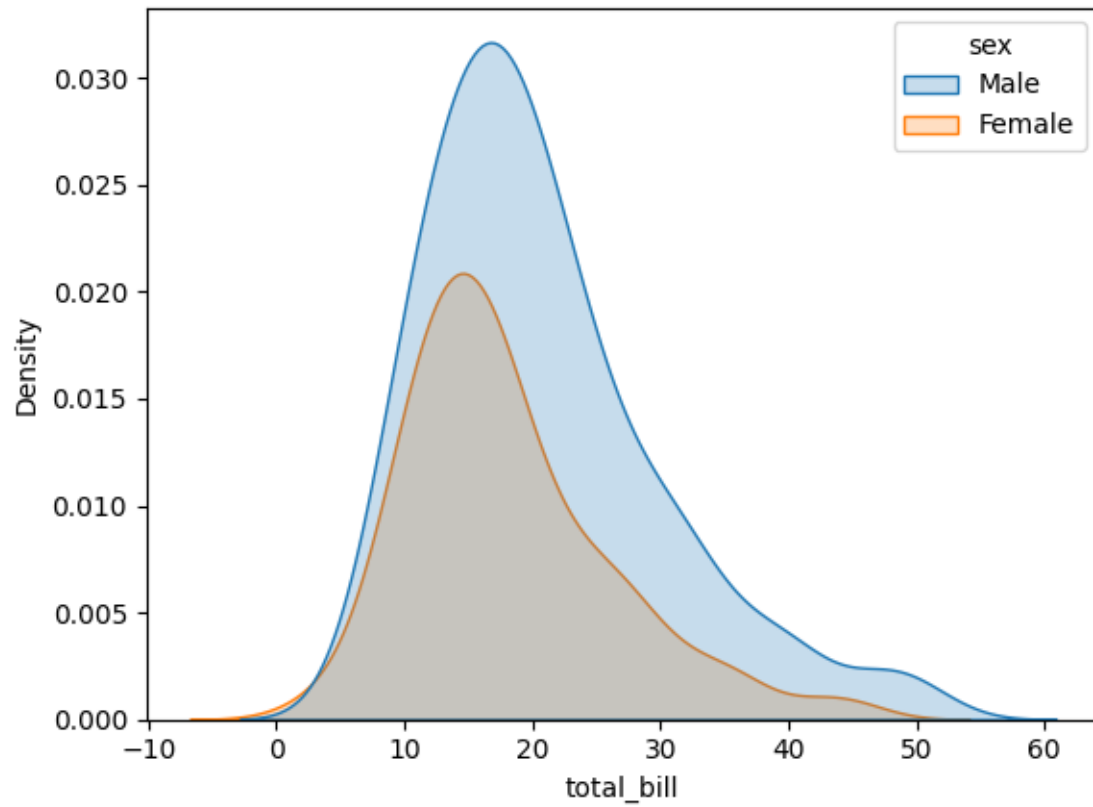
```
[33]: sns.kdeplot(data=tips, x='total_bill')
```

```
[33]: <Axes: xlabel='total_bill', ylabel='Density'>
```



```
[39]: sns.kdeplot(data=tips, x='total_bill', hue='sex', fill=True)
```

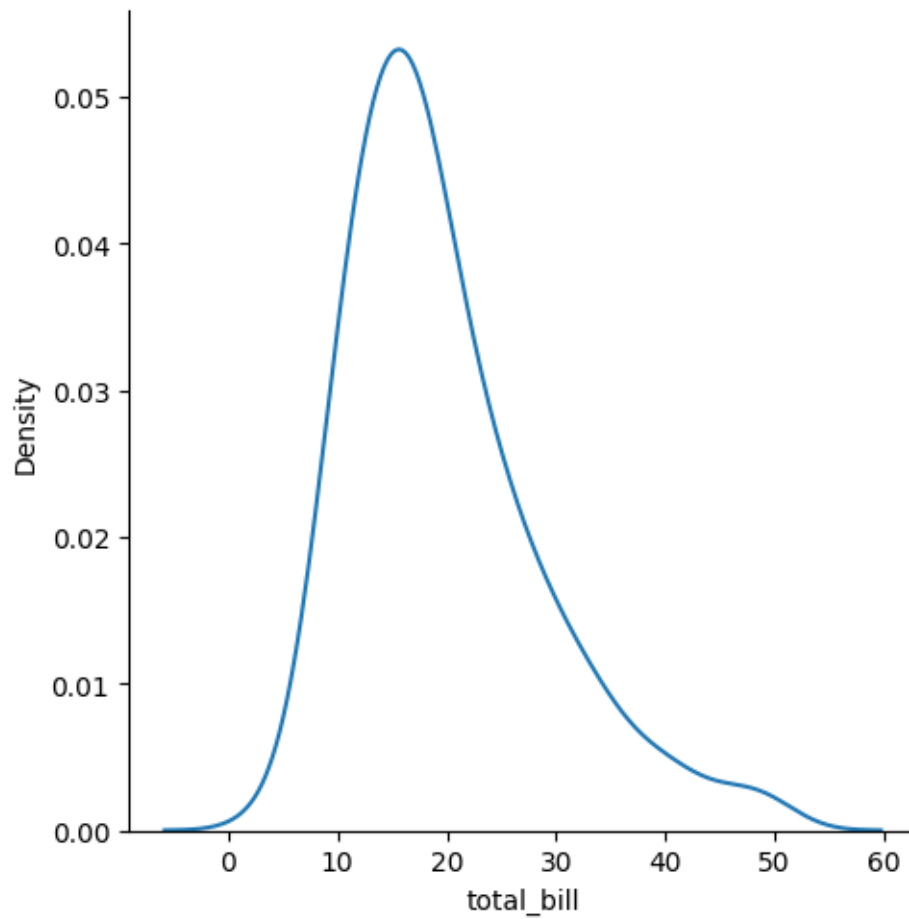
```
[39]: <Axes: xlabel='total_bill', ylabel='Density'>
```



```
sns.displot(data, x, kind)
```

```
[34]: sns.displot(data=tips, x='total_bill', kind='kde')
```

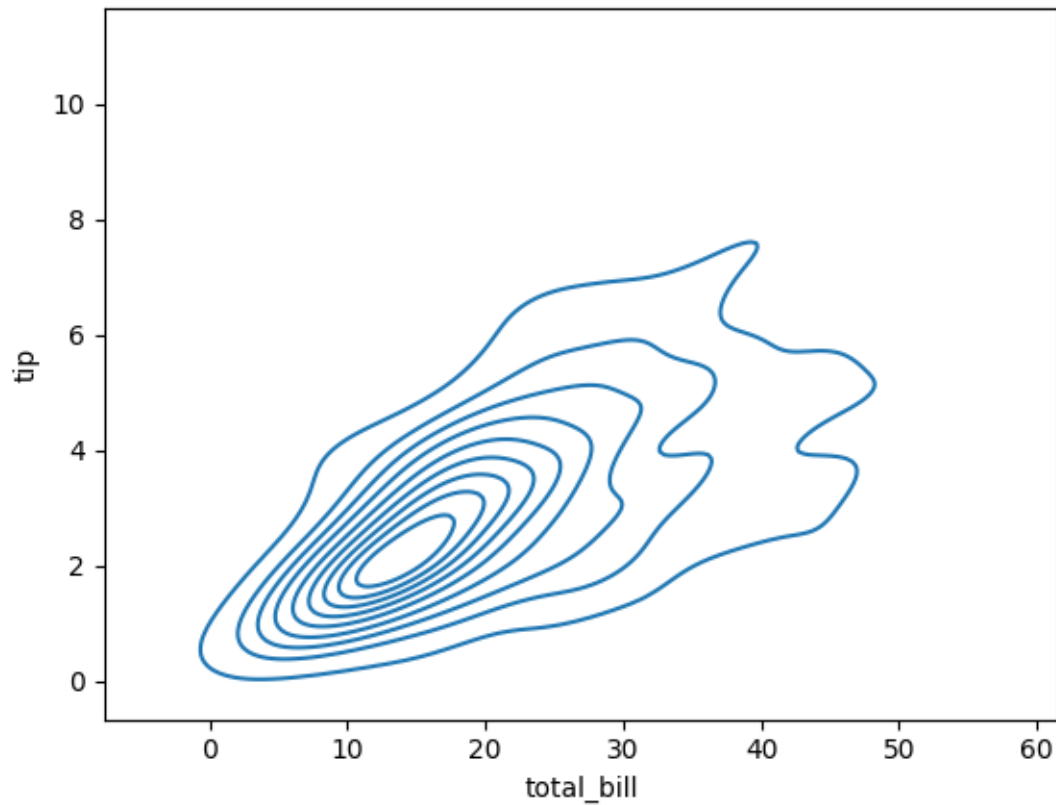
```
[34]: <seaborn.axisgrid.FacetGrid at 0x25087b63ec0>
```



Bivariate Kdeplot A bivariate kde plot smoothes the (x,y) observations with a 2D Gaussian

```
[43]: sns.kdeplot(data=tips, x='total_bill', y='tip')
```

```
[43]: <Axes: xlabel='total_bill', ylabel='tip'>
```



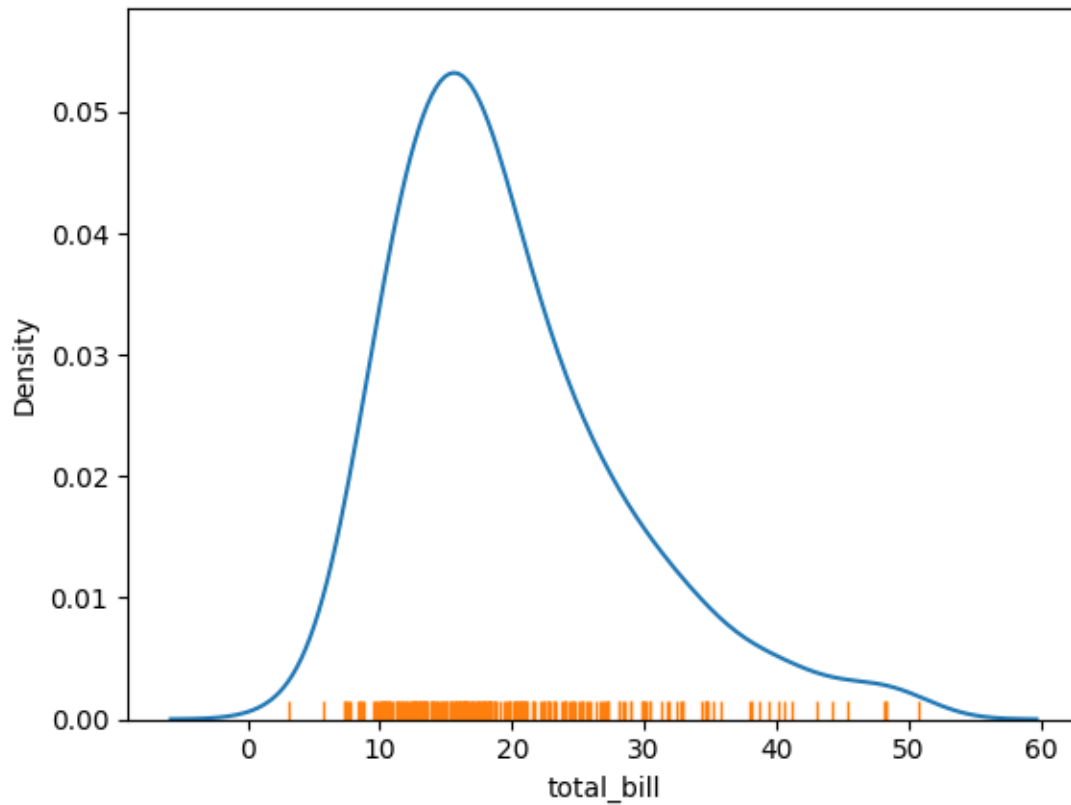
1.2.3 `sns.rugplot(data, x)`

Plot marginal distributions by drawing ticks along the axes.

This function is intended to complement other plots by showing the location of individual observations in an unobtrusive way

```
[40]: sns.kdeplot(data=tips, x='total_bill')  
      sns.rugplot(data=tips, x='total_bill')
```

```
[40]: <Axes: xlabel='total_bill', ylabel='Density'>
```



1.3 3. Matrix Plots

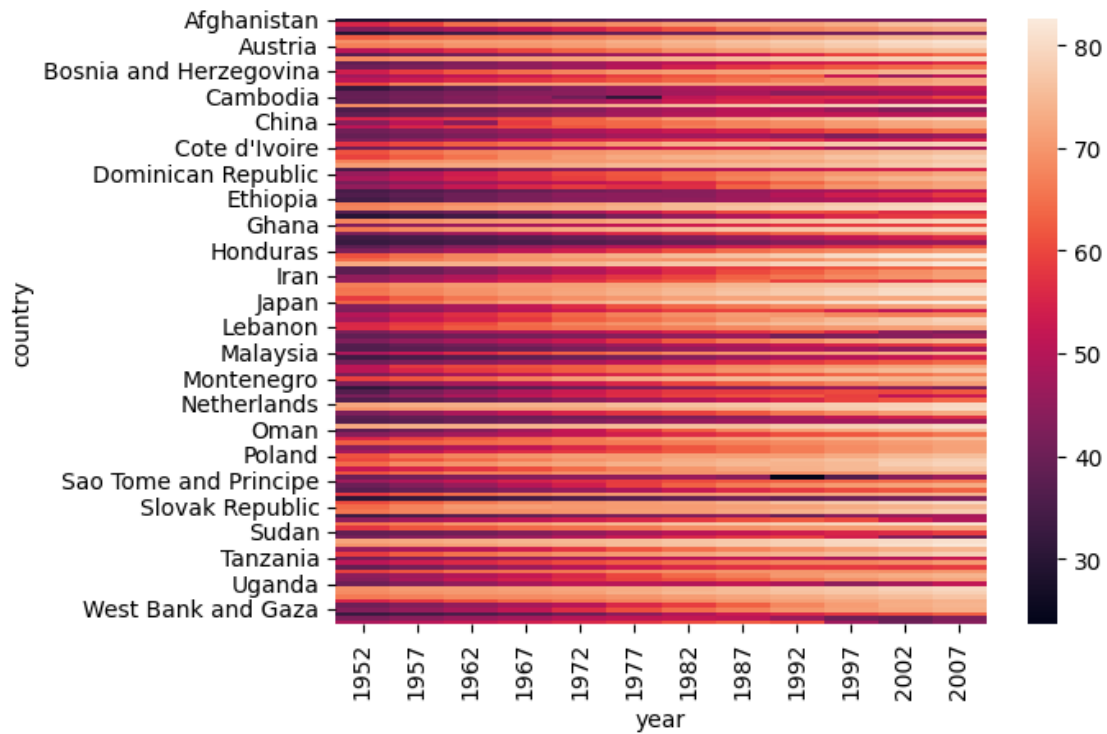
Figure level - `displot()`

Axis level - 1. `heatmap()` 2. `clustermap()`

1.3.1 `sns.heatmap(data, annot=False, linewidth, cmap)`

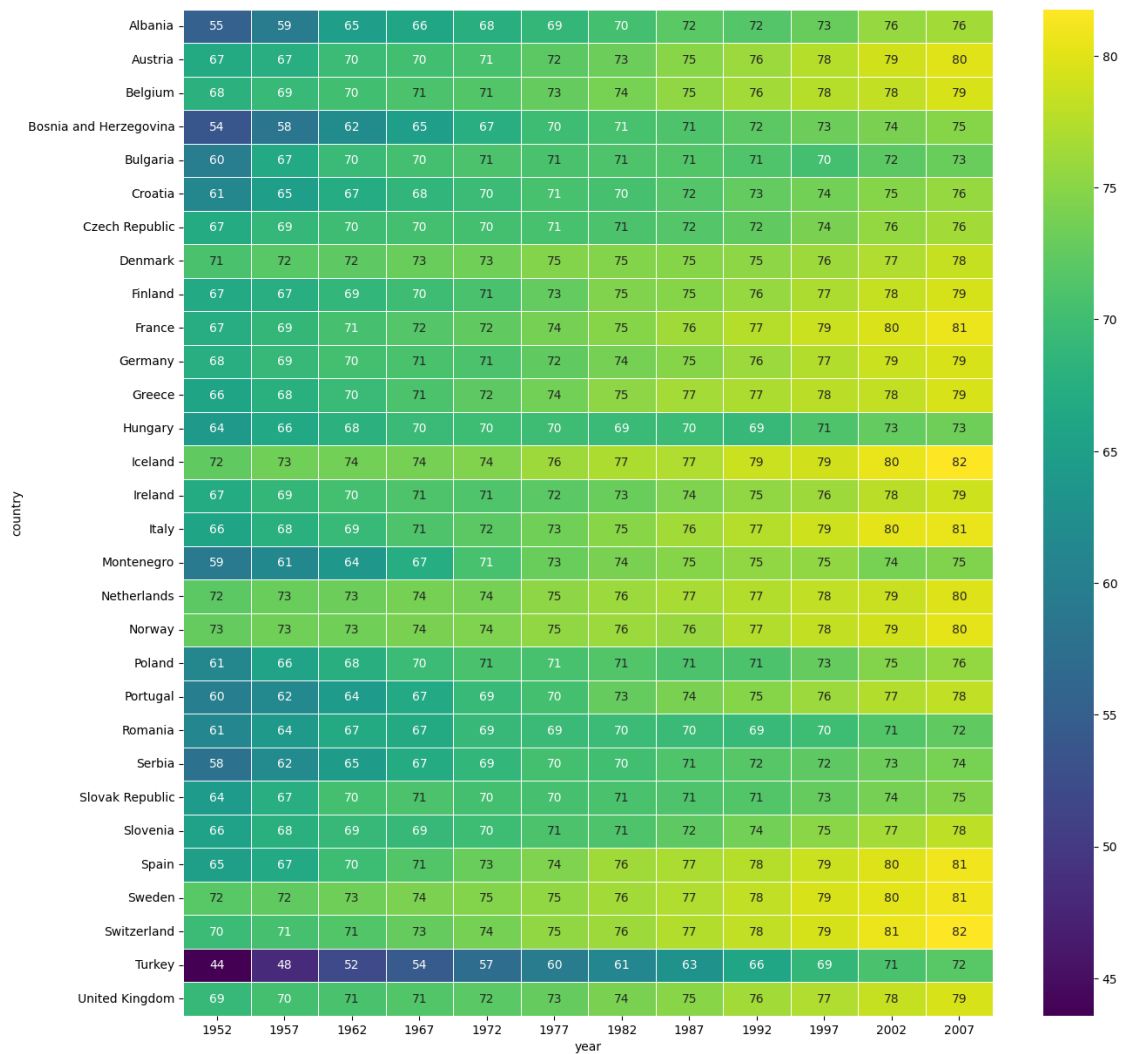
```
[49]: temp = gap.pivot(index='country', columns='year', values='lifeExp')
      sns.heatmap(temp)
```

```
[49]: <Axes: xlabel='year', ylabel='country'>
```



```
[56]: temp = gap[gap['continent'] == 'Europe'].pivot(index='country', columns='year',
↪values='lifeExp')
import matplotlib.pyplot as plt
plt.figure(figsize=(15,15))
sns.heatmap(temp, annot=True, linewidth=0.5, cmap='viridis')
```

```
[56]: <Axes: xlabel='year', ylabel='country'>
```



1.3.2 sns.clustermap(data, annot=False, linewidth, cmap)

Plot a matrix dataset as a hierarchically-clustered heatmap.
This function requires scipy to be available.

```
[58]: import plotly.express as px
iris = px.data.iris()
iris
```

```
[58]:      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

```

```

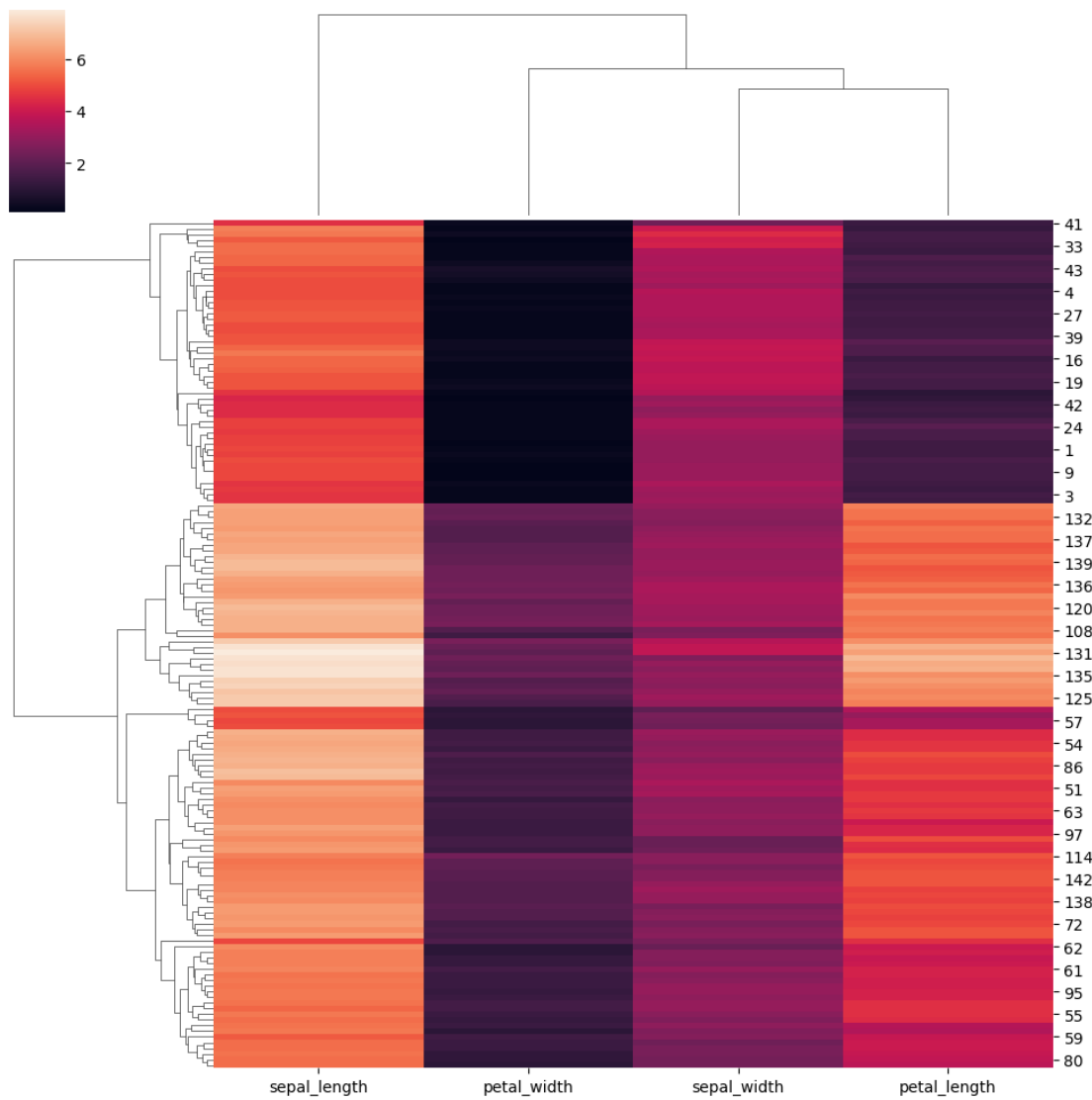
      species_id
0             1
1             1
2             1
3             1
4             1
..          ...
145          3
146          3
147          3
148          3
149          3

```

[150 rows x 6 columns]

```
[59]: sns.clustermap(iris.iloc[:, [0,1,2,3]])
```

```
[59]: <seaborn.matrix.ClusterGrid at 0x2508f22d160>
```



1.4 4. Categorical Plots

1.4.1 Categorical Scatter Plot

- Stripplot
- Swarmplot

1.4.2 Categorical Distribution Plots

- Boxplot
- Violinplot

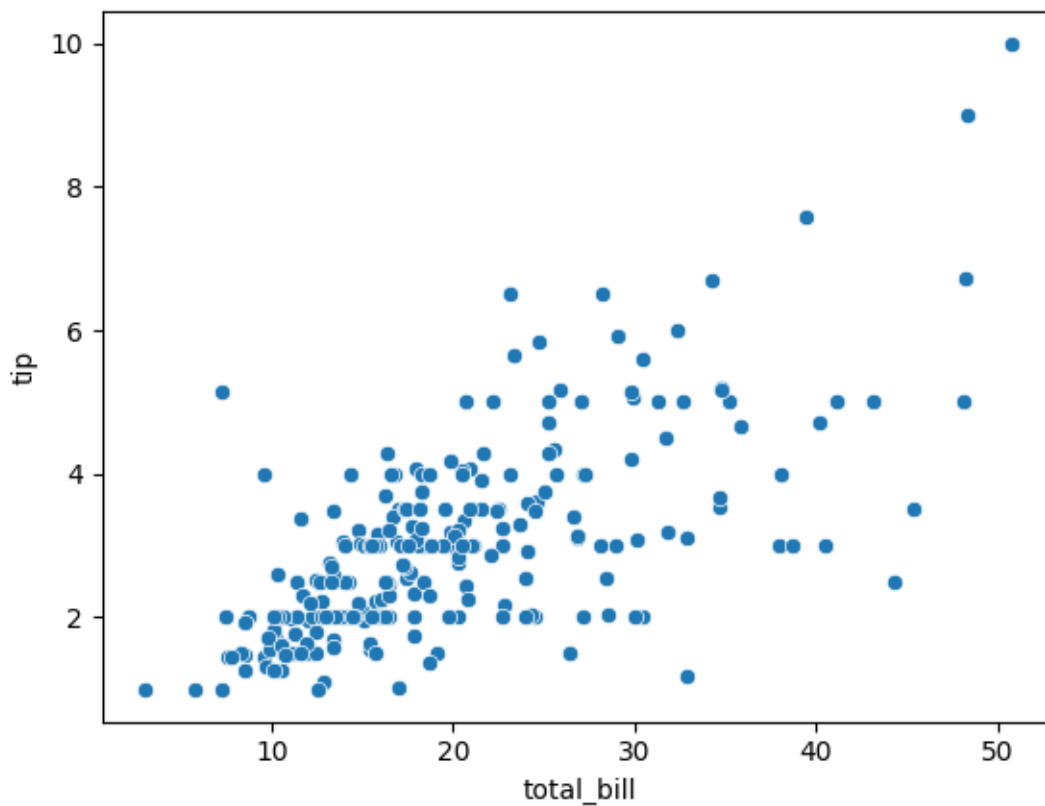
1.4.3 Categorical Estimate Plot -> for central tendency

- Barplot
- Pointplot
- Countplot

1.4.4 Figure level function -> catplot

```
[60]: sns.scatterplot(data=tips, x='total_bill', y='tip')
```

```
[60]: <Axes: xlabel='total_bill', ylabel='tip'>
```

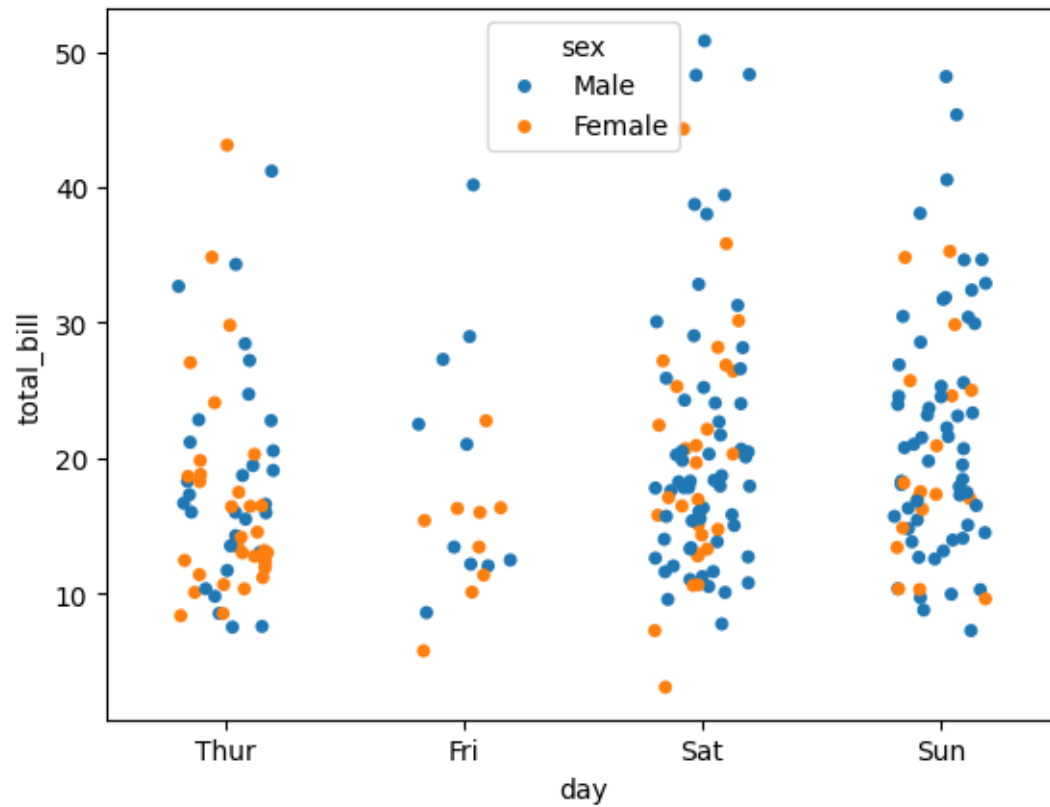


1.4.5 Categorical Scatter Plot

`stripplot(data, x, y, jitter, hue)` Axes level function

```
[63]: sns.stripplot(data=tips, x='day', y='total_bill', jitter=0.2, hue='sex')
```

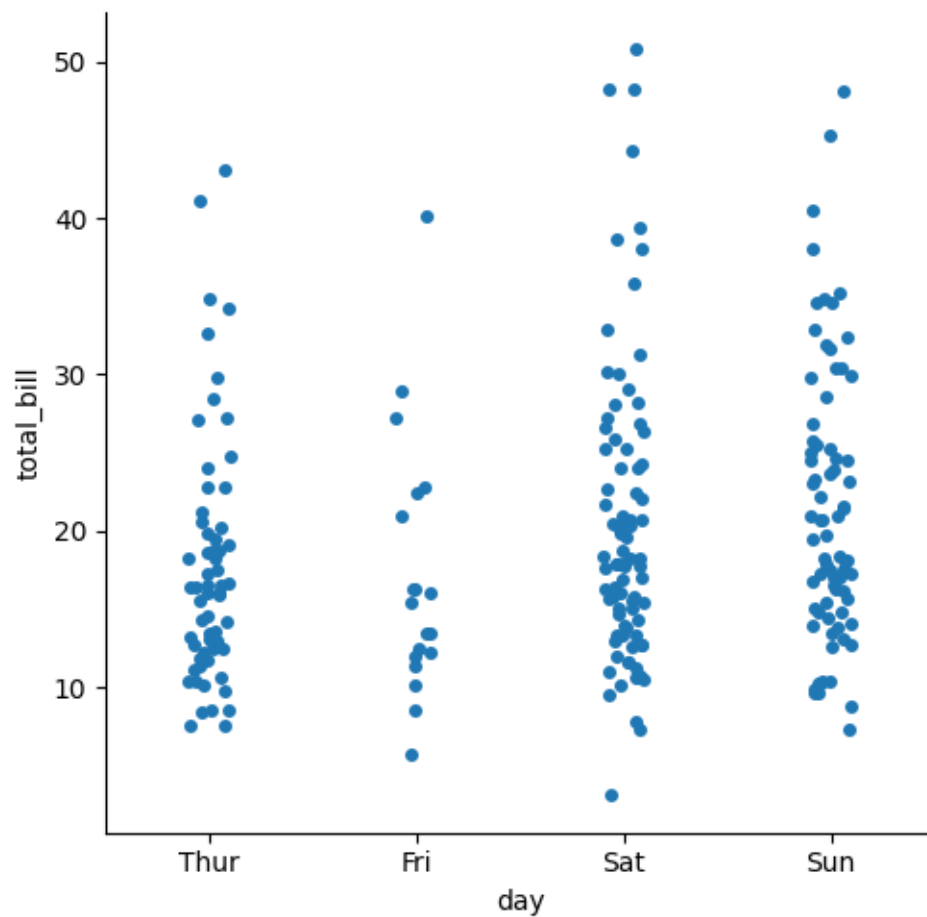
```
[63]: <Axes: xlabel='day', ylabel='total_bill'>
```



`sns.catplot(data, x, y, kind)` Figure level function

```
[62]: sns.catplot(data=tips, x='day', y='total_bill', kind='strip')
```

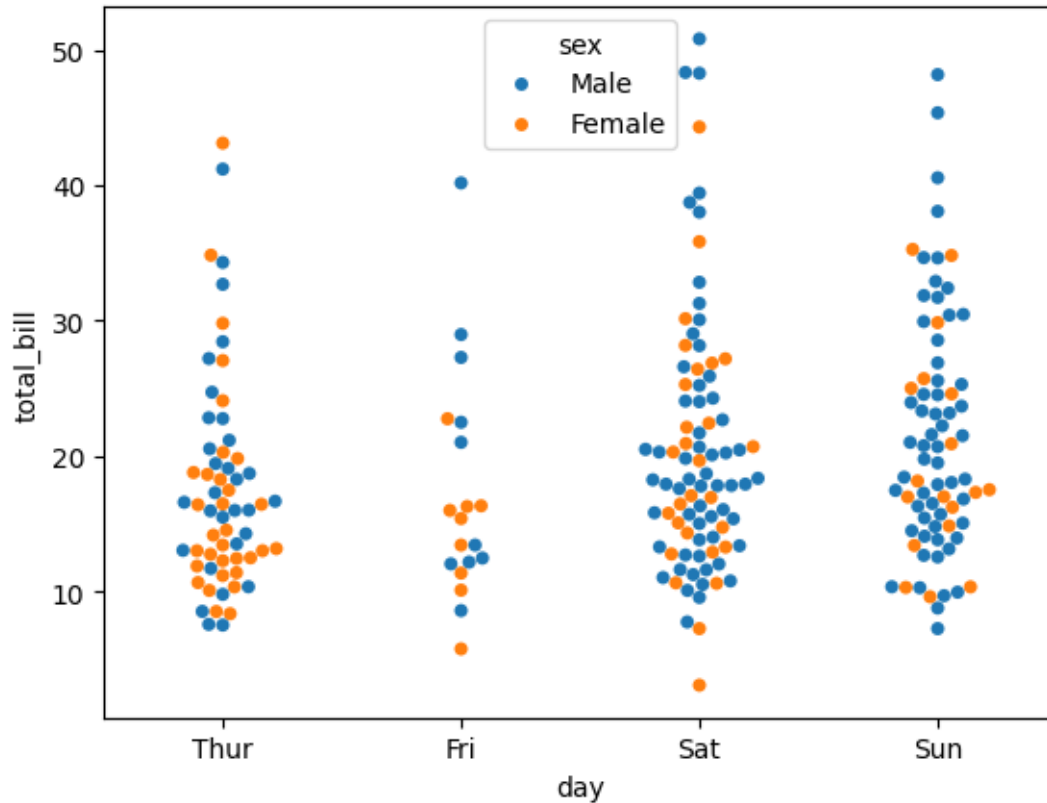
```
[62]: <seaborn.axisgrid.FacetGrid at 0x2508ea45490>
```



```
sns.swarmplot(data, x, y, hue)
```

```
[65]: sns.swarmplot(data=tips, x='day', y='total_bill', hue='sex')
```

```
[65]: <Axes: xlabel='day', ylabel='total_bill'>
```



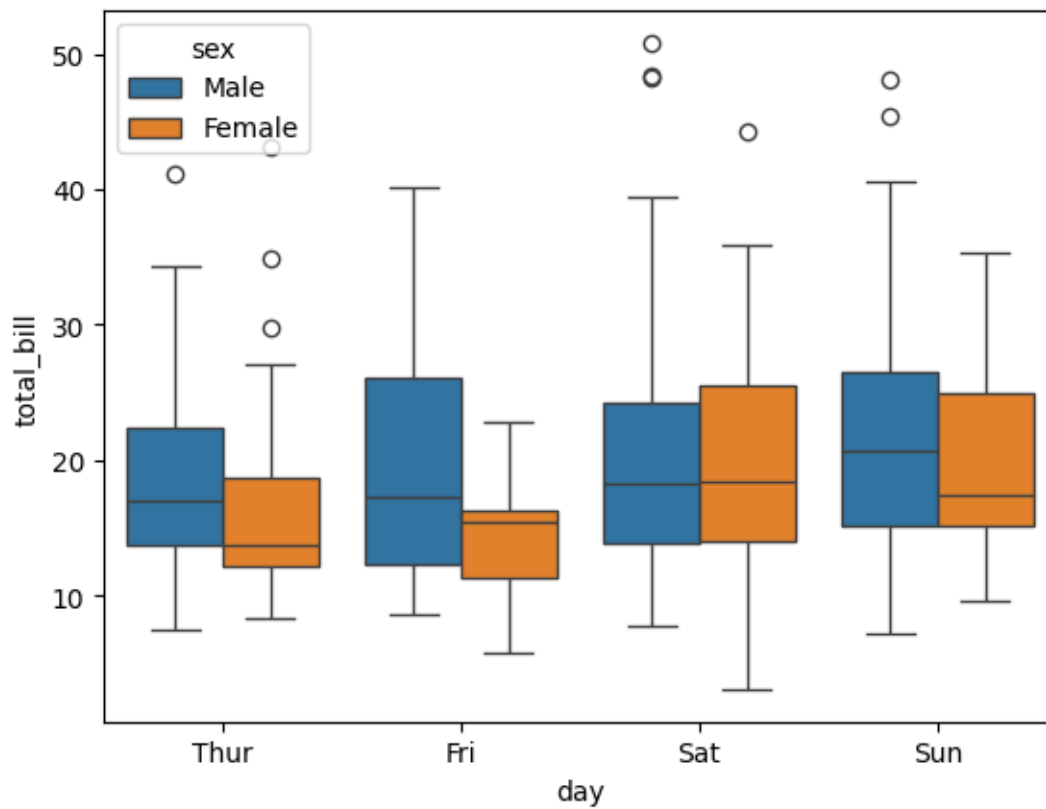
1.4.6 Boxplot

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile [Q1], median, third quartile [Q3] and “maximum”). It can tell you about your outliers and what their values are. Boxplots can also tell you if your data is symmetrical, how tightly your data is grouped and if and how your data is skewed.

1.4.7 `sns.boxplot(data, x, y, hue)`

```
[68]: sns.boxplot(data=tips, x='day', y='total_bill', hue='sex')
```

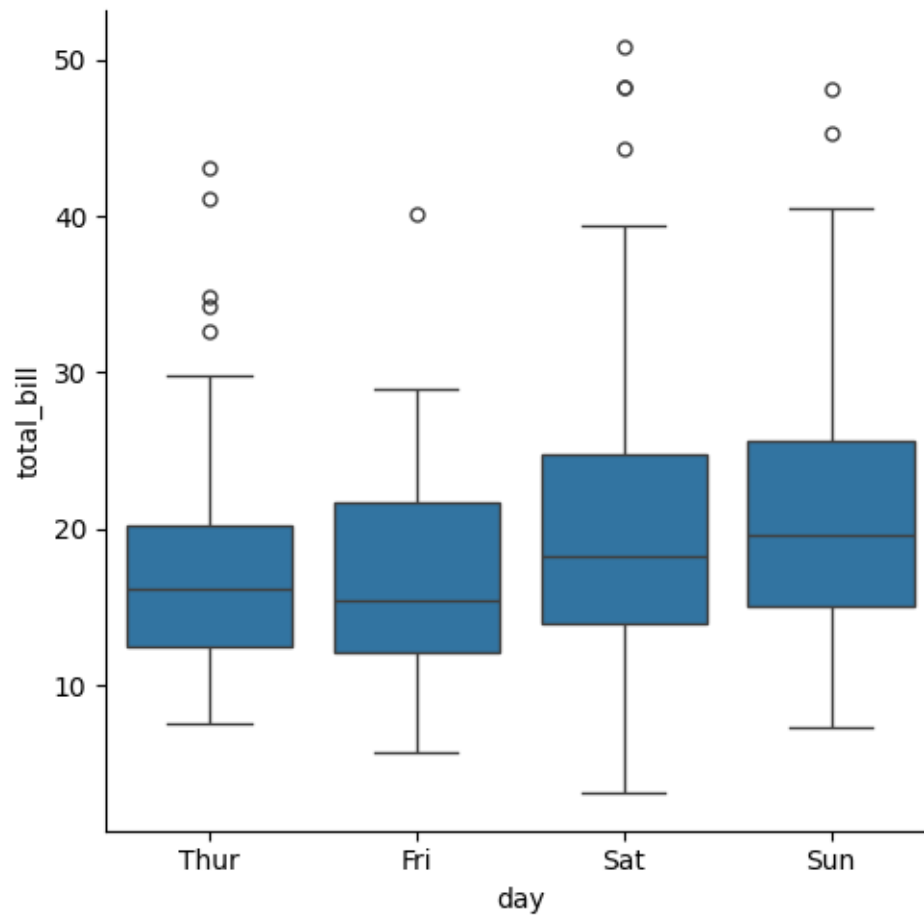
```
[68]: <Axes: xlabel='day', ylabel='total_bill'>
```



1.4.8 sns.catplot(data, x, y, kind)

```
[67]: sns.catplot(data=tips, x='day', y='total_bill', kind='box')
```

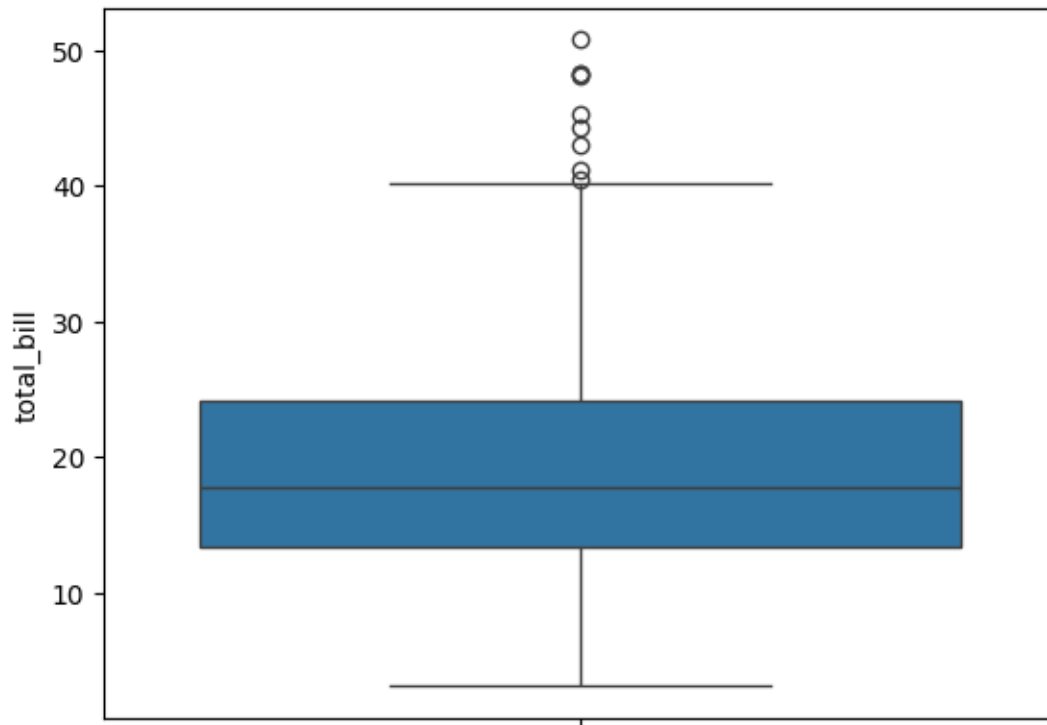
```
[67]: <seaborn.axisgrid.FacetGrid at 0x2508f8e6870>
```



Single Boxplot -> numerical column

```
[69]: sns.boxplot(data=tips, y='total_bill')
```

```
[69]: <Axes: ylabel='total_bill'>
```

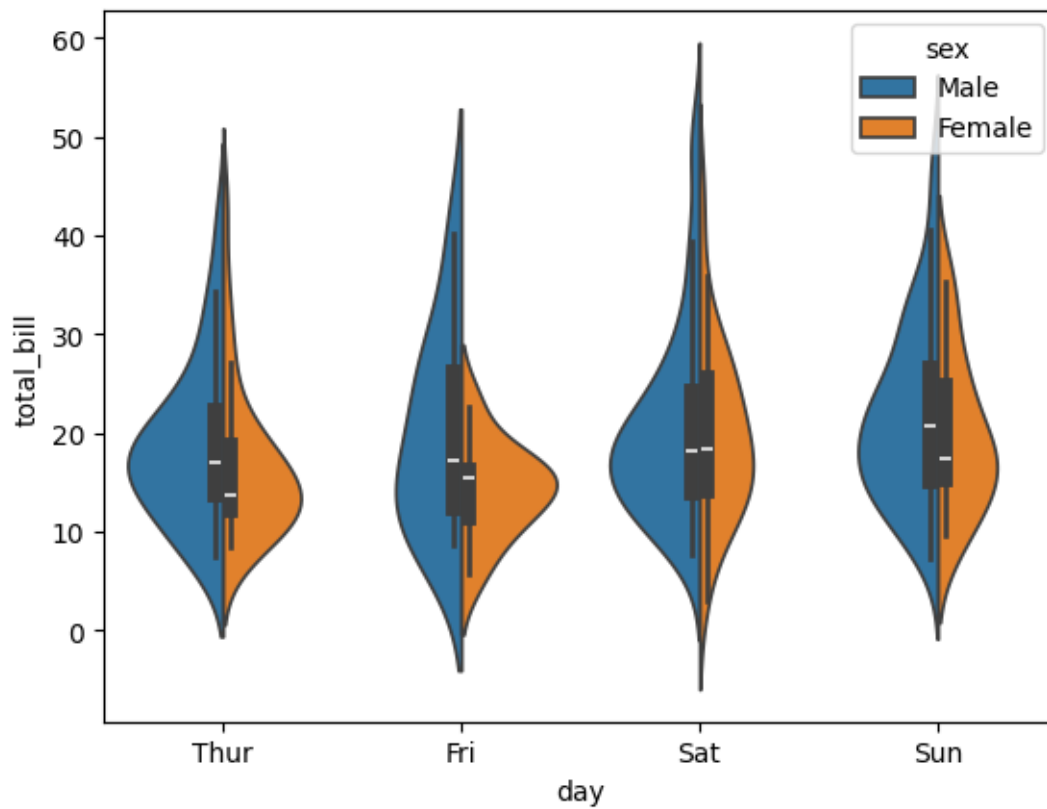



1.4.9 sns.violinplot(data, x, y, hue, split=False)

By default split is False which means for every category in hue there is two violin plots, but if split is True then in single violin plot both category is merged.

```
[73]: sns.violinplot(data=tips, x='day', y='total_bill', hue='sex', split=True)
```

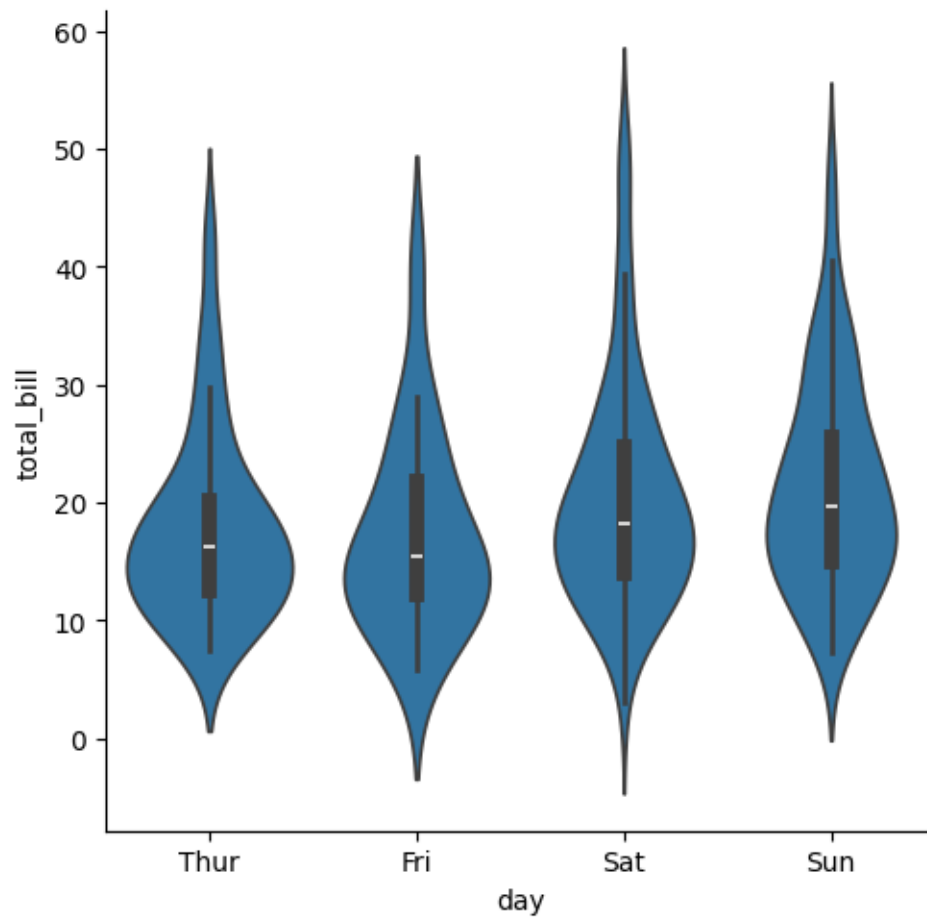
```
[73]: <Axes: xlabel='day', ylabel='total_bill'>
```



```
sns.catplot(data, x, y, kind)
```

```
[71]: sns.catplot(data=tips, x='day', y='total_bill', kind='violin')
```

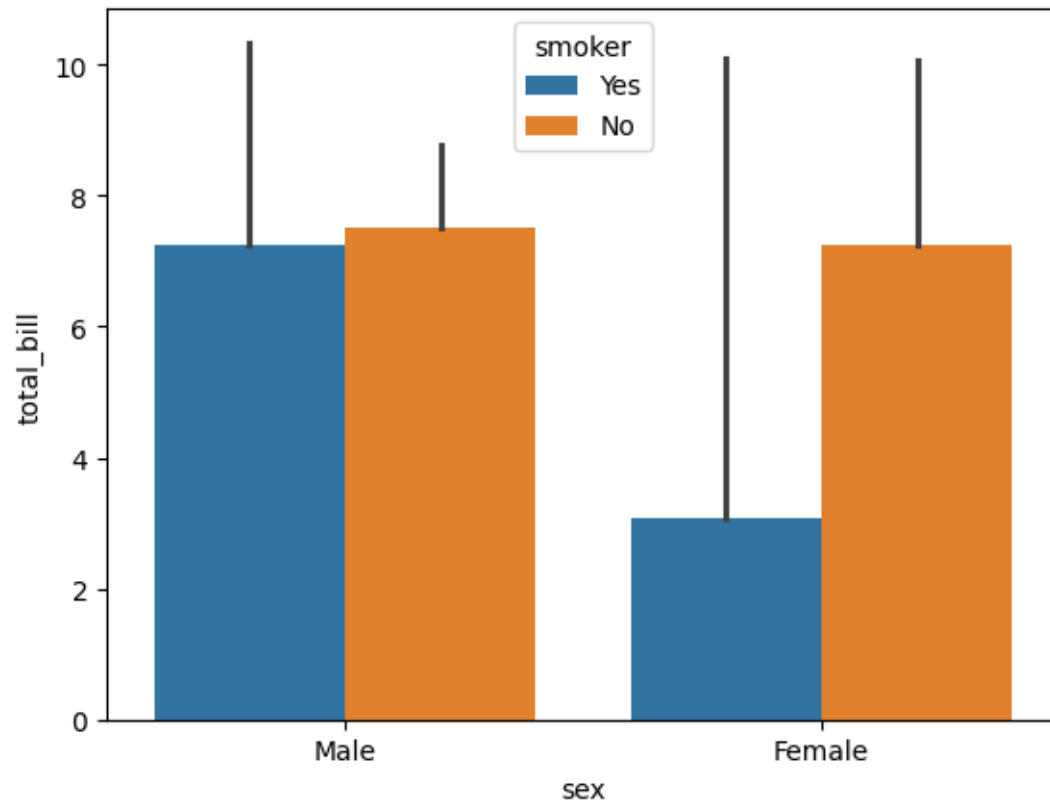
```
[71]: <seaborn.axisgrid.FacetGrid at 0x25086b73050>
```



1.4.10 sns.barplot(data, x, y, hue, estimator)

```
[79]: sns.barplot(data=tips, x='sex', y='total_bill', hue='smoker', estimator=np.min)
```

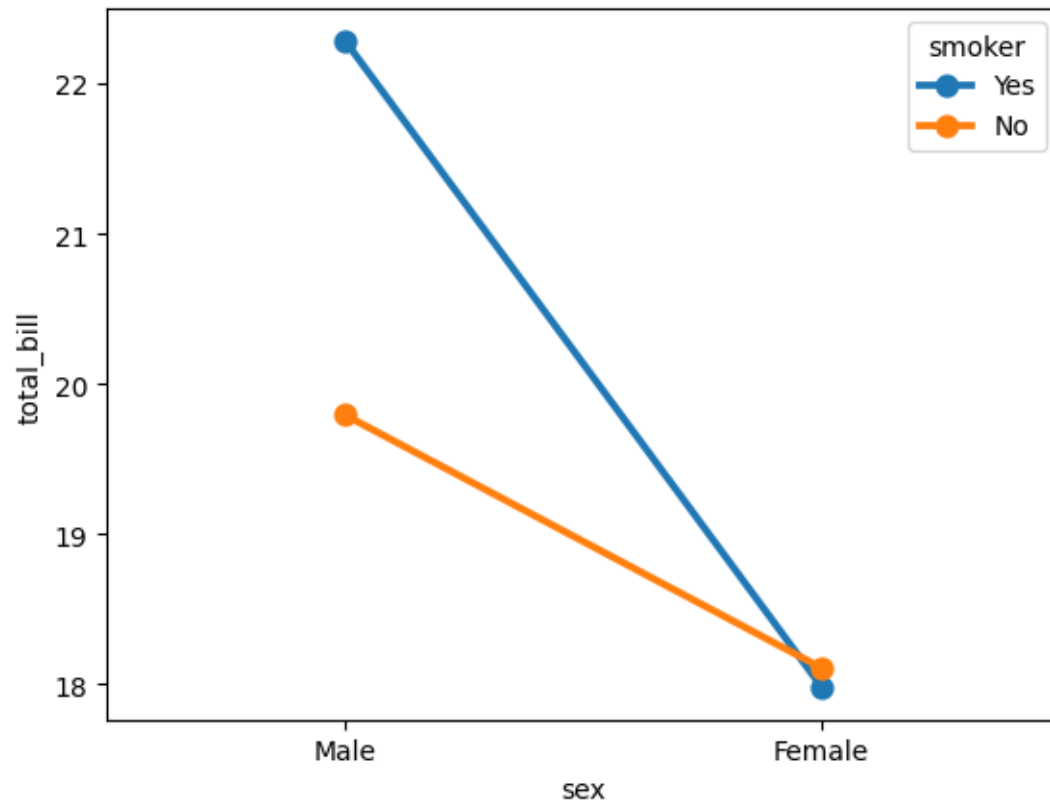
```
[79]: <Axes: xlabel='sex', ylabel='total_bill'>
```



1.4.11 sns.pointplot(data, x, y, hue, errorbar)

```
[83]: sns.pointplot(data=tips, x='sex', y='total_bill', hue='smoker', errorbar=None)
```

```
[83]: <Axes: xlabel='sex', ylabel='total_bill'>
```

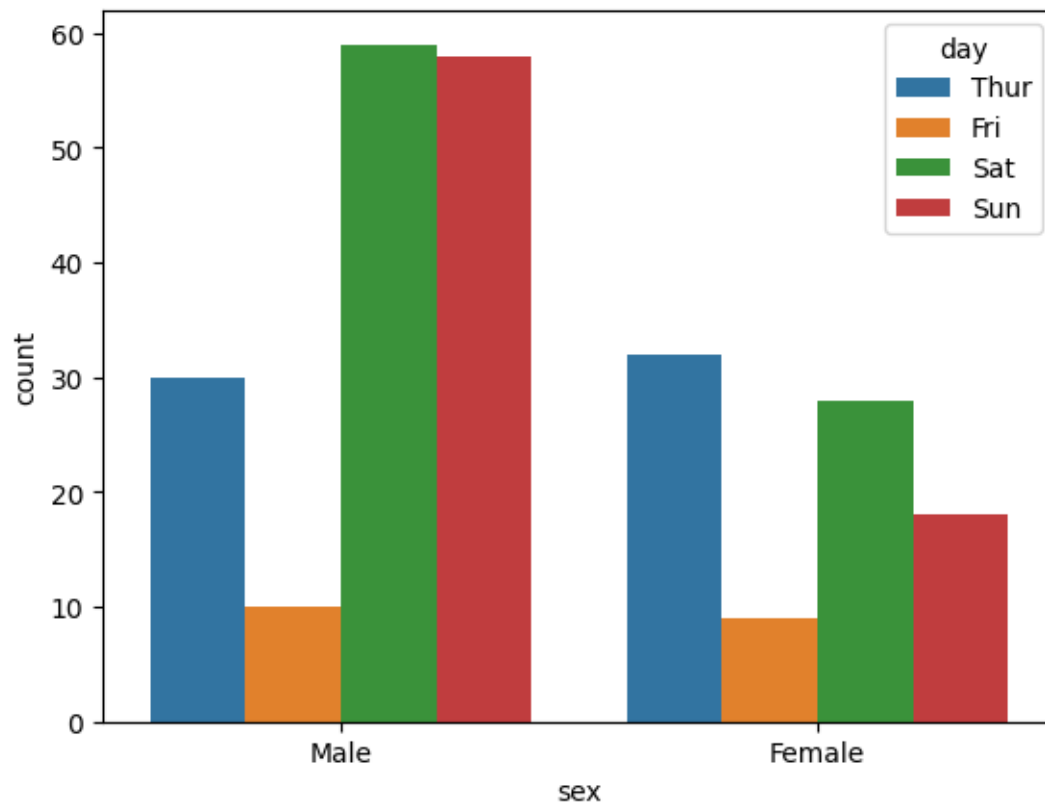


1.4.12 sns.countplot(data, x, hue)

When there are multiple observations in each category, it also uses bootstrapping to compute a confidence interval around the estimate, which is plotted using error bars.

```
[84]: sns.countplot(data=tips, x='sex', hue='day')
```

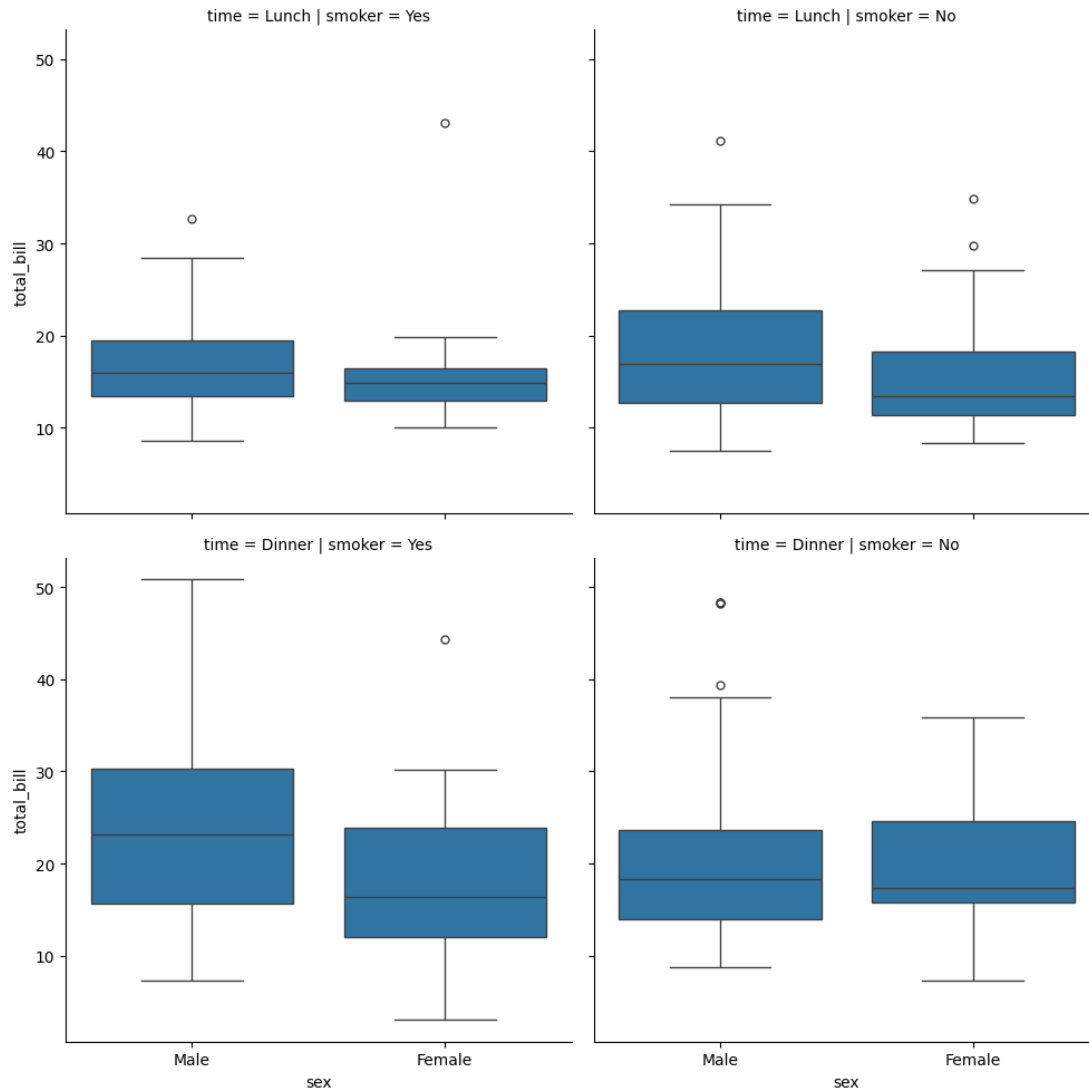
```
[84]: <Axes: xlabel='sex', ylabel='count'>
```



Facet plots

```
[85]: sns.catplot(data=tips,   
                ↪x='sex',y='total_bill',col='smoker',kind='box',row='time')
```

```
[85]: <seaborn.axisgrid.FacetGrid at 0x25092bbd520>
```



1.5 4. Regression Plots

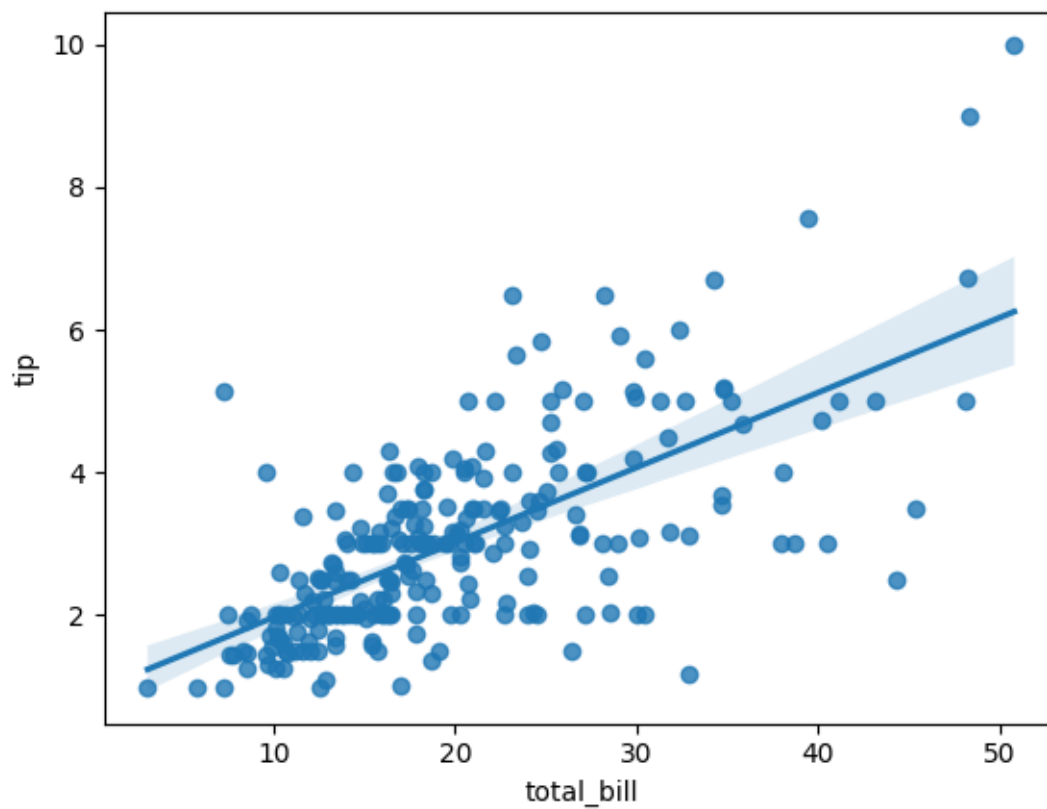
- regplot
- lmpplot

In the simplest invocation, both functions draw a scatterplot of two variables, x and y , and then fit the regression model $y \sim x$ and plot the resulting regression line and a 95% confidence interval for that regression.

Hue parameter is not available.

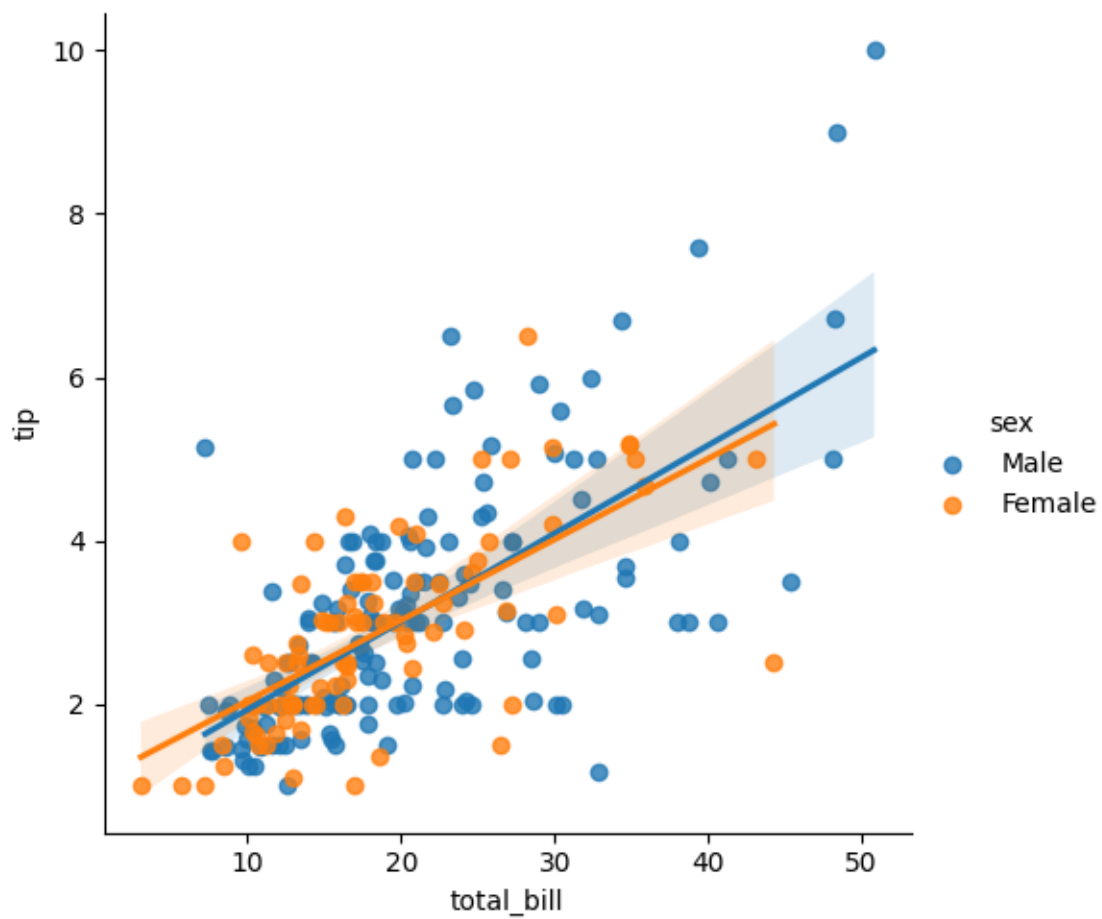
```
[86]: sns.regplot(data=tips, x='total_bill', y='tip')
```

```
[86]: <Axes: xlabel='total_bill', ylabel='tip'>
```



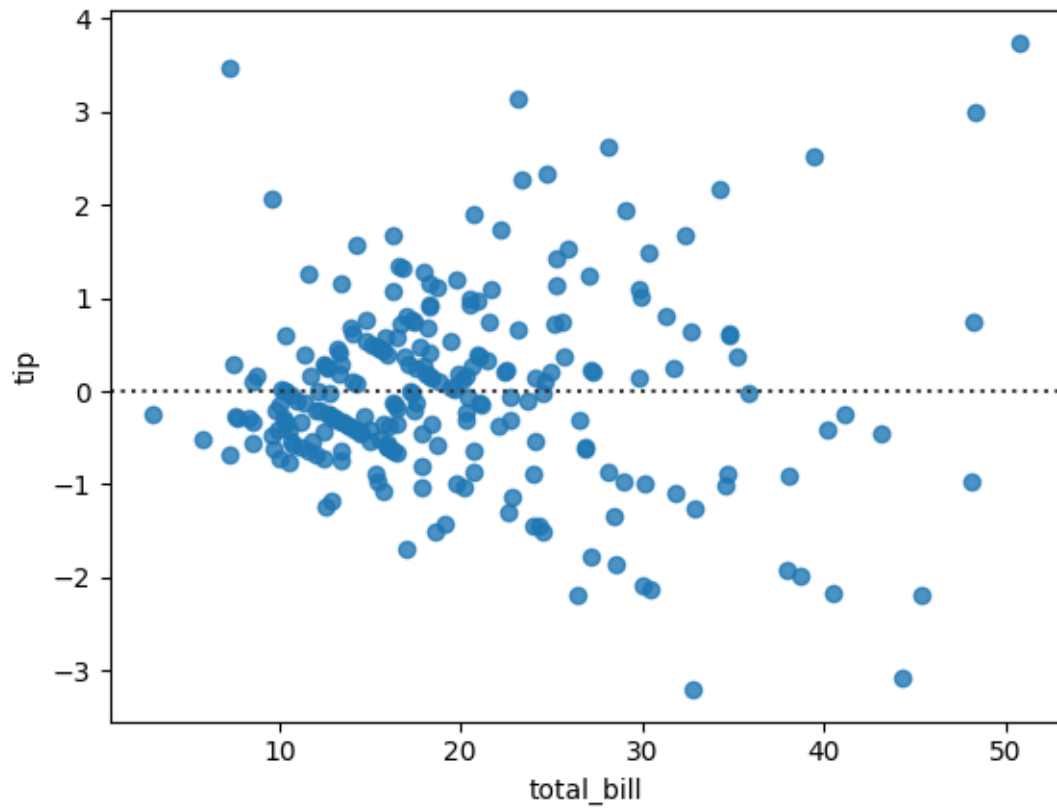
```
[87]: sns.lmplot(data=tips, x='total_bill', y='tip', hue='sex')
```

```
[87]: <seaborn.axisgrid.FacetGrid at 0x25092e10bc0>
```

```
[88]: sns.residplot(data=tips, x='total_bill', y='tip')
```

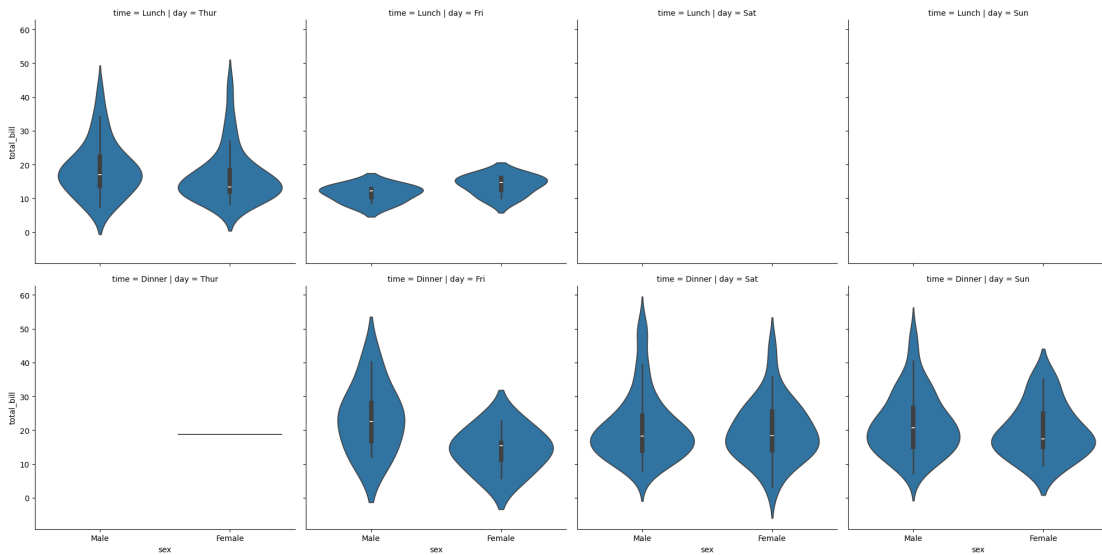
```
[88]: <Axes: xlabel='total_bill', ylabel='tip'>
```



1.5.1 Facet Grid

```
[89]: sns.catplot(data=tips, x='sex', y='total_bill', kind='violin', col='day',  
    ↪row='time')
```

```
[89]: <seaborn.axisgrid.FacetGrid at 0x25092e6fd10>
```



```
[92]: g = sns.FacetGrid(data=tips, col='day', row='time', hue='smoker')
g.map(sns.boxplot, 'sex', 'total_bill')
g.add_legend()
```

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:718:
UserWarning:

Using the boxplot function without specifying `order` is likely to produce an incorrect plot.

```
-----
TypeError                                Traceback (most recent call last)
Cell In[92], line 2
      1 g = sns.FacetGrid(data=tips, col='day', row='time', hue='smoker')
----> 2 g.map(sns.boxplot, 'sex', 'total_bill')
      3 g.add_legend()

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:758, in FacetGrid.map(self, func, *args, **kwargs)
    755     plot_args = [v.values for v in plot_args]
    757     # Draw the plot
--> 758     self._facet_plot(func, ax, plot_args, kwargs)
    760 # Finalize the annotations and layout
    761 self._finalize_grid(args[:2])

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:854, in FacetGrid._facet_plot(self, func, ax, plot_args, plot_kwargs)
    852     plot_args = []
```

```

853     plot_kwargs["ax"] = ax
--> 854 func(*plot_args, **plot_kwargs)
856 # Sort out the supporting information
857 self._update_legend_data(ax)

```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\categorical.py:1634, in
    boxplot(data, x, y, hue, order, hue_order, orient, color, palette,
    saturation, fill, dodge, width, gap, whis, linecolor, linewidth, fliersize,
    hue_norm, native_scale, log_scale, formatter, legend, ax, **kwargs)
    1627 color = _default_color(
    1628     ax.fill_between, hue, color,
    1629     {k: v for k, v in kwargs.items() if k in ["c", "color", "fc",
    "facecolor"]},
    1630     saturation=saturation,
    1631 )
    1632 linecolor = p._complement_color(linecolor, color, p._hue_map)
-> 1634 p.plot_boxes(
    1635     width=width,
    1636     dodge=dodge,
    1637     gap=gap,
    1638     fill=fill,
    1639     whis=whis,
    1640     color=color,
    1641     linecolor=linecolor,
    1642     linewidth=linewidth,
    1643     fliersize=fliersize,
    1644     plot_kws=kwargs,
    1645 )
    1647 p._add_axis_labels(ax)
    1648 p._adjust_cat_axis(ax, axis=p.orient)

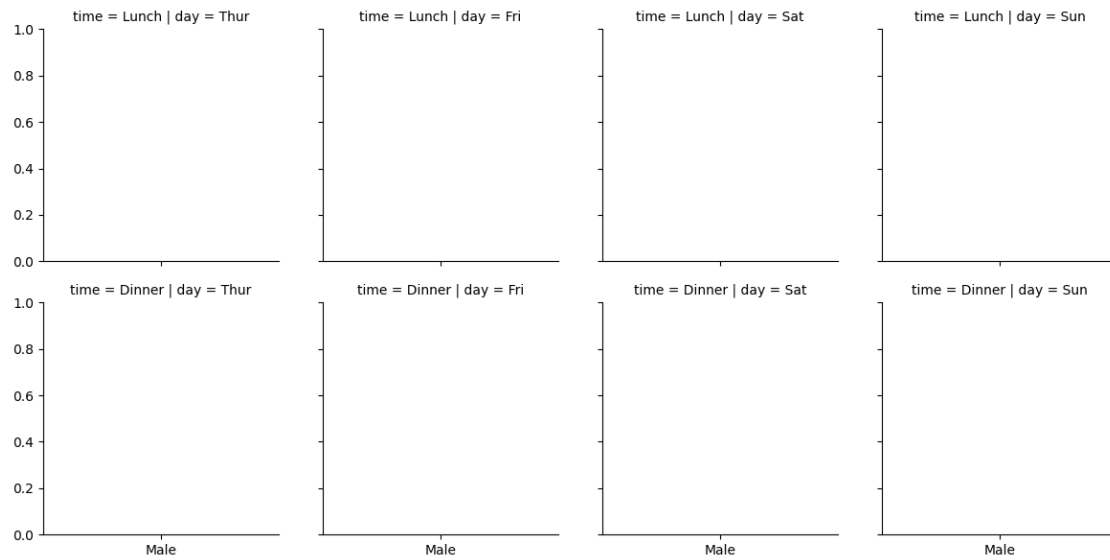
```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\categorical.py:700, in
    _CategoricalPlotter.plot_boxes(self, width, dodge, gap, fill, whis, color,
    linecolor, linewidth, fliersize, plot_kws)
    679 default_kws = dict(
    680     bxpstats=stats.to_dict("records"),
    681     positions=data[self.orient],
    (...)
    697 )
    698 )
    699 boxplot_kws = {**default_kws, **plot_kws}
--> 700 artists = ax.bxp(**boxplot_kws)
    702 # Reset artist widths after adding so everything stays positive
    703 ori_idx = ["x", "y"].index(self.orient)

```

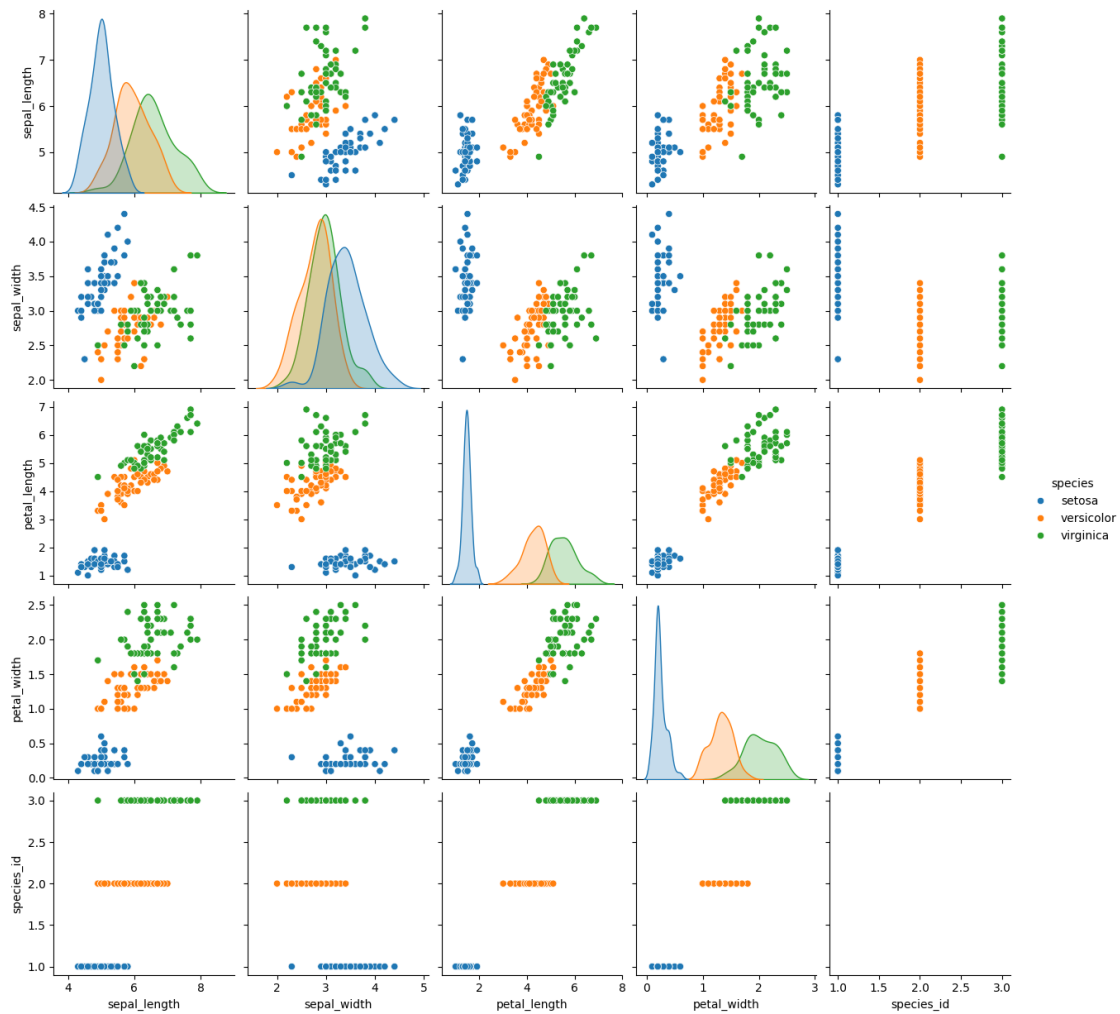
TypeError: Axes.bxp() got an unexpected keyword argument 'label'



1.6 Plotting Pairwise Relationship (PairGrid vs Pairplot)

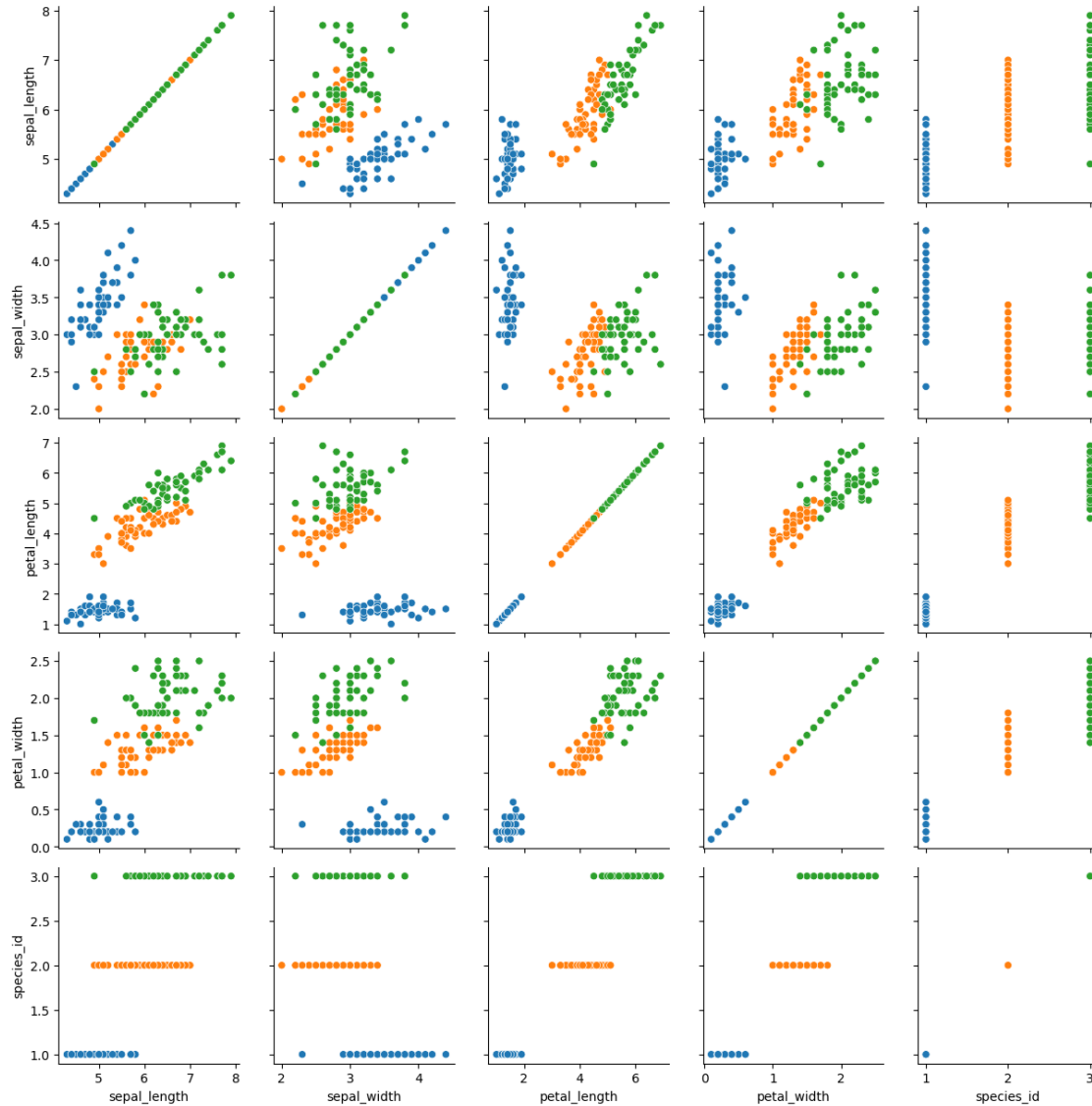
```
[93]: sns.pairplot(iris, hue='species')
```

```
[93]: <seaborn.axisgrid.PairGrid at 0x25095b760f0>
```



```
[95]: # pair grid
g = sns.PairGrid(data=iris, hue='species')
# g.map
g.map(sns.scatterplot)
```

```
[95]: <seaborn.axisgrid.PairGrid at 0x25099c51f10>
```



```
[96]: # map_diag -> map_offdiag
```

```
g = sns.PairGrid(data=iris, hue='species')
g.map_diag(sns.boxplot)
g.map_offdiag(sns.kdeplot)
```

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:

UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

```
-----  
IndexError                                Traceback (most recent call last)  
Cell In[96], line 5  
      3 g = sns.PairGrid(data=iris, hue='species')  
      4 g.map_diag(sns.boxplot)  
----> 5 g.map_offdiag(sns.kdeplot)  
  
File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1425, in PairGrid.map_offdiag(self, func, **kwargs)  
    1414 """Plot with a bivariate function on the off-diagonal subplots.  
    1415  
    1416 Parameters  
    (...)  
    1422  
    1423 """  
    1424 if self.square_grid:  
-> 1425     self.map_lower(func, **kwargs)  
    1426     if not self._corner:  
    1427         self.map_upper(func, **kwargs)  
  
File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1395, in PairGrid.map_lower(self, func, **kwargs)  
    1384 """Plot with a bivariate function on the lower diagonal subplots.  
    1385  
    1386 Parameters  
    (...)  
    1392  
    1393 """  
    1394 indices = zip(*np.tril_indices_from(self.axes, -1))  
-> 1395 self._map_bivariate(func, indices, **kwargs)  
    1396 return self  
  
File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1574, in PairGrid._map_bivariate(self, func, indices, **kwargs)  
    1572     if ax is None: # i.e. we are in corner mode
```



```

1573         continue
-> 1574     self._plot_bivariate(x_var, y_var, ax, func, **kws)
1575     self._add_axis_labels()
1577     if "hue" in signature(func).parameters:

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615, in
PairGrid._plot_bivariate(self, x_var, y_var, ax, func, **kwargs)
    1611     if "hue" not in kwargs:
    1612         kwargs.update({
    1613             "hue": hue, "hue_order": self._hue_order, "palette": self.
-> _orig_palette,
    1614         })
-> 1615     func(x=x, y=y, **kwargs)
    1617     self._update_legend_data(ax)

```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:1715
-> in kdeplot(data, x, y, hue, weights, palette, hue_order, hue_norm, color,
fill, multiple, common_norm, common_grid, cumulative, bw_method, bw_adjust,
warn_singular, log_scale, levels, thresh, gridsize, cut, clip, legend, cbar,
cbar_ax, cbar_kws, ax, **kwargs)
    1701     p.plot_univariate_density(
    1702         multiple=multiple,
    1703         common_norm=common_norm,
    (...)
    1710         **plot_kws,
    1711     )
    1713 else:
-> 1715     p.plot_bivariate_density(
    1716         common_norm=common_norm,
    1717         fill=fill,
    1718         levels=levels,
    1719         thresh=thresh,
    1720         legend=legend,
    1721         color=color,
    1722         warn_singular=warn_singular,
    1723         cbar=cbar,
    1724         cbar_ax=cbar_ax,
    1725         cbar_kws=cbar_kws,
    1726         estimate_kws=estimate_kws,
    1727         **kwargs,
    1728     )
    1730     return ax

```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:1113
-> in DistributionPlotter.plot_bivariate_density(self, common_norm, fill,
levels, thresh, color, legend, cbar, warn_singular, cbar_ax, cbar_kws,
estimate_kws, **contour_kws)
    1111     # Transform from iso-proportions to iso-densities
    1112     if common_norm:

```

```

-> 1113     common_levels = self._quantile_to_level(
1114         list(densities.values()), levels,
1115     )
1116     draw_levels = {k: common_levels for k in densities}
1117 else:

```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:200,
  in _DistributionPlotter._quantile_to_level(self, data, quantile)
    198 normalized_values = np.cumsum(sorted_values) / values.sum()
    199 idx = np.searchsorted(normalized_values, 1 - isoprop)
--> 200 levels = np.take(sorted_values, idx, mode="clip")
    201 return levels

```

```

File c:\Program Files\Python312\Lib\site-packages\numpy\core\fromnumeric.py:192
  in take(a, indices, axis, out, mode)
    95 @array_function_dispatch(_take_dispatcher)
    96 def take(a, indices, axis=None, out=None, mode='raise'):
    97     """
    98     Take elements from an array along an axis.
    99     (...)
    190         [5, 7]])
    191     """
--> 192     return _wrapfunc(a, 'take', indices, axis=axis, out=out, mode=mode)

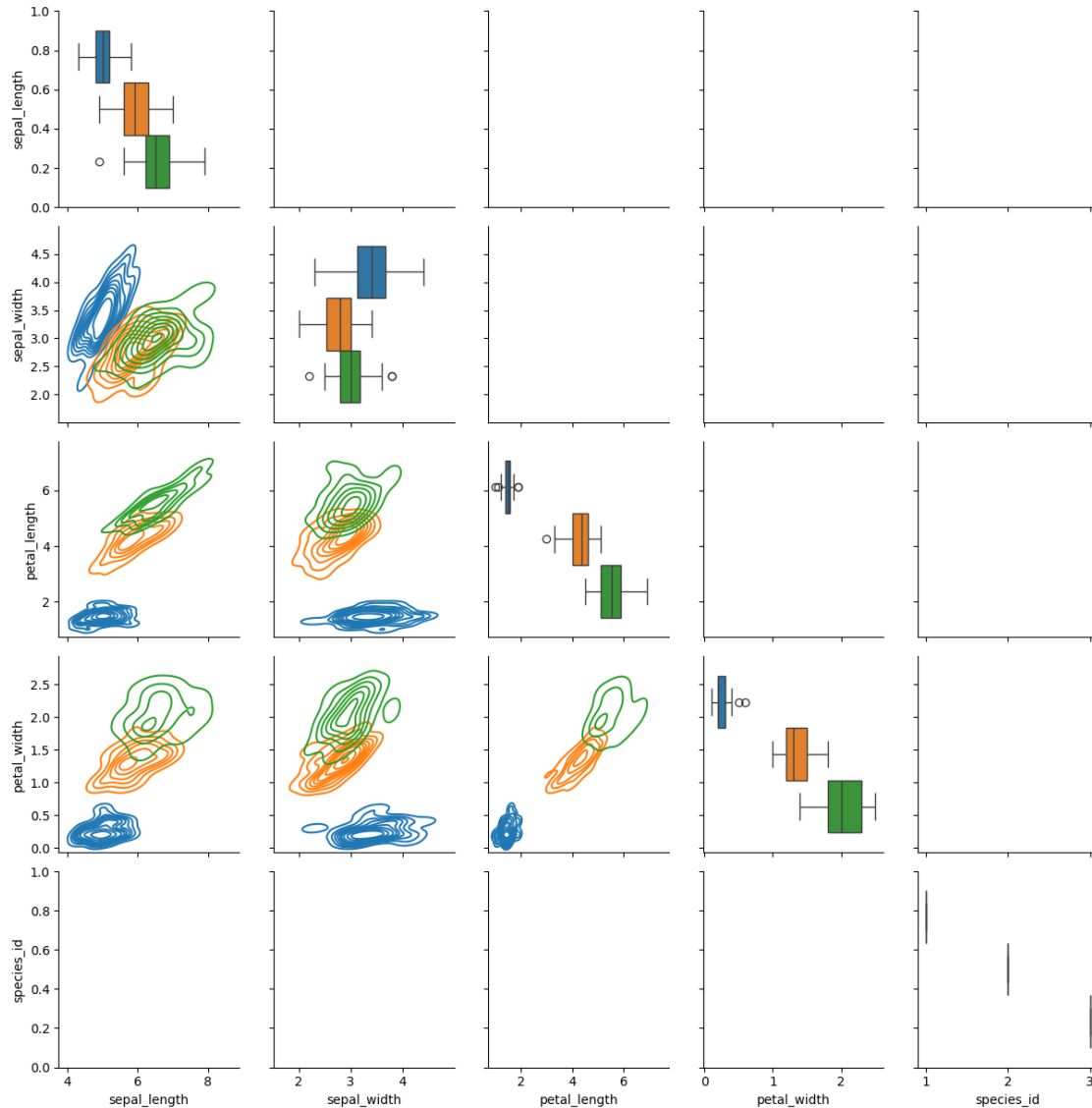
```

```

File c:\Program Files\Python312\Lib\site-packages\numpy\core\fromnumeric.py:59,
  in _wrapfunc(obj, method, *args, **kwds)
    56     return _wrapit(obj, method, *args, **kwds)
    58 try:
--> 59     return bound(*args, **kwds)
    60 except TypeError:
    61     # A TypeError occurs if the object does have such a method in its
    62     # class, but its signature is not identical to that of NumPy's. Thi
    (...)
    66     # Call _wrapit from within the except clause to ensure a potential
    67     # exception has a traceback chain.
    68     return _wrapit(obj, method, *args, **kwds)

```

IndexError: cannot do a non-empty take from an empty axes.



```
[97]: # map_diag -> map_upper -> map_lower
g = sns.PairGrid(data=iris, hue='species')
g.map_diag(sns.histplot)
g.map_upper(sns.kdeplot)
g.map_lower(sns.scatterplot)
```

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:

UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615:
UserWarning:

KDE cannot be estimated (0 variance or perfect covariance). Pass
`warn_singular=False` to disable this warning.

```
-----
IndexError                                Traceback (most recent call last)
Cell In[97], line 4
      2 g = sns.PairGrid(data=iris, hue='species')
      3 g.map_diag(sns.histplot)
----> 4 g.map_upper(sns.kdeplot)
      5 g.map_lower(sns.scatterplot)

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1410, in PairGrid.map_upper(self, func, **kwargs)
    1399 """Plot with a bivariate function on the upper diagonal subplots.
    1400
    1401 Parameters
    (...)
    1407
    1408 """
    1409 indices = zip(*np.triu_indices_from(self.axes, 1))
-> 1410 self._map_bivariate(func, indices, **kwargs)
    1411 return self

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1574, in PairGrid._map_bivariate(self, func, indices, **kwargs)
    1572     if ax is None: # i.e. we are in corner mode
    1573         continue
-> 1574     self._plot_bivariate(x_var, y_var, ax, func, **kws)
    1575 self._add_axis_labels()
    1577 if "hue" in signature(func).parameters:

File c:\Program Files\Python312\Lib\site-packages\seaborn\axisgrid.py:1615, in PairGrid._plot_bivariate(self, x_var, y_var, ax, func, **kwargs)
    1611 if "hue" not in kwargs:
    1612     kwargs.update({
    1613         "hue": hue, "hue_order": self._hue_order, "palette": self.
    ↪ _orig_palette,
    1614     })
```

```
-> 1615 func(x=x, y=y, **kwargs)
    1617 self._update_legend_data(ax)
```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:1715

```
↳ in kdeplot(data, x, y, hue, weights, palette, hue_order, hue_norm, color,
↳ fill, multiple, common_norm, common_grid, cumulative, bw_method, bw_adjust,
↳ warn_singular, log_scale, levels, thresh, gridsize, cut, clip, legend, cbar,
↳ cbar_ax, cbar_kws, ax, **kwargs)
    1701     p.plot_univariate_density(
    1702         multiple=multiple,
    1703         common_norm=common_norm,
    (...)
    1710         **plot_kws,
    1711     )
    1713 else:
-> 1715     p.plot_bivariate_density(
    1716         common_norm=common_norm,
    1717         fill=fill,
    1718         levels=levels,
    1719         thresh=thresh,
    1720         legend=legend,
    1721         color=color,
    1722         warn_singular=warn_singular,
    1723         cbar=cbar,
    1724         cbar_ax=cbar_ax,
    1725         cbar_kws=cbar_kws,
    1726         estimate_kws=estimate_kws,
    1727         **kwargs,
    1728     )
    1730 return ax
```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:1113

```
↳ in _DistributionPlotter.plot_bivariate_density(self, common_norm, fill,
↳ levels, thresh, color, legend, cbar, warn_singular, cbar_ax, cbar_kws,
↳ estimate_kws, **contour_kws)
    1111 # Transform from iso-proportions to iso-densities
    1112 if common_norm:
-> 1113     common_levels = self._quantile_to_level(
    1114         list(densities.values()), levels,
    1115     )
    1116     draw_levels = {k: common_levels for k in densities}
    1117 else:
```

File c:\Program Files\Python312\Lib\site-packages\seaborn\distributions.py:200,

```
↳ in _DistributionPlotter._quantile_to_level(self, data, quantile)
    198 normalized_values = np.cumsum(sorted_values) / values.sum()
    199 idx = np.searchsorted(normalized_values, 1 - isoprop)
--> 200 levels = np.take(sorted_values, idx, mode="clip")
    201 return levels
```

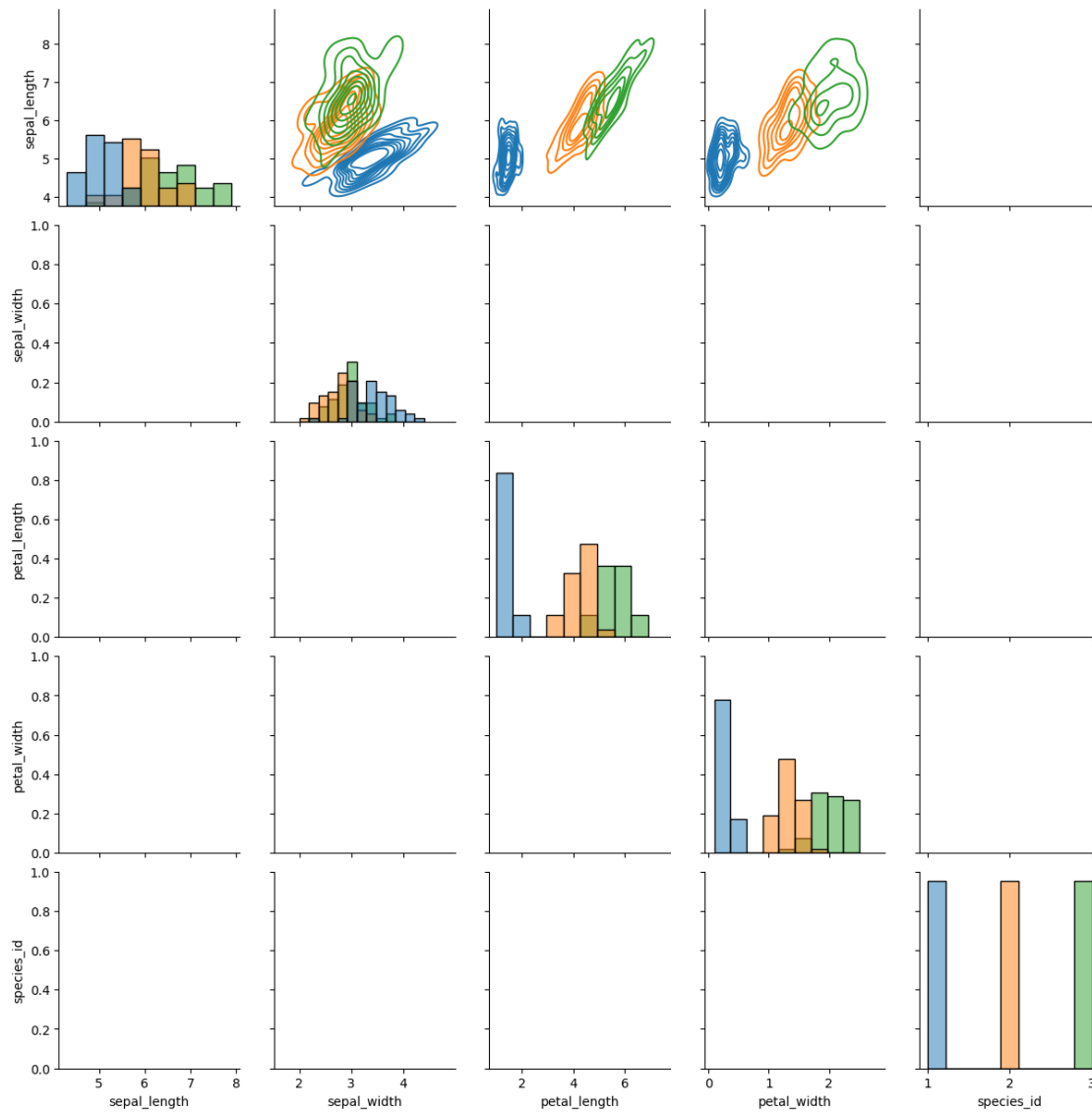
```

File c:\Program Files\Python312\Lib\site-packages\numpy\core\fromnumeric.py:192
↳ in take(a, indices, axis, out, mode)
    95 @array_function_dispatch(_take_dispatcher)
    96 def take(a, indices, axis=None, out=None, mode='raise'):
    97     """
    98     Take elements from an array along an axis.
    99
   (...
   190         [5, 7]])
   191     """
--> 192     return _wrapfunc(a, 'take', indices, axis=axis, out=out, mode=mode)

File c:\Program Files\Python312\Lib\site-packages\numpy\core\fromnumeric.py:59,
↳ in _wrapfunc(obj, method, *args, **kws)
    56     return _wrapit(obj, method, *args, **kws)
    58 try:
--> 59     return bound(*args, **kws)
    60 except TypeError:
    61     # A TypeError occurs if the object does have such a method in its
    62     # class, but its signature is not identical to that of NumPy's. Thi
   (...
    66     # Call _wrapit from within the except clause to ensure a potential
    67     # exception has a traceback chain.
    68     return _wrapit(obj, method, *args, **kws)

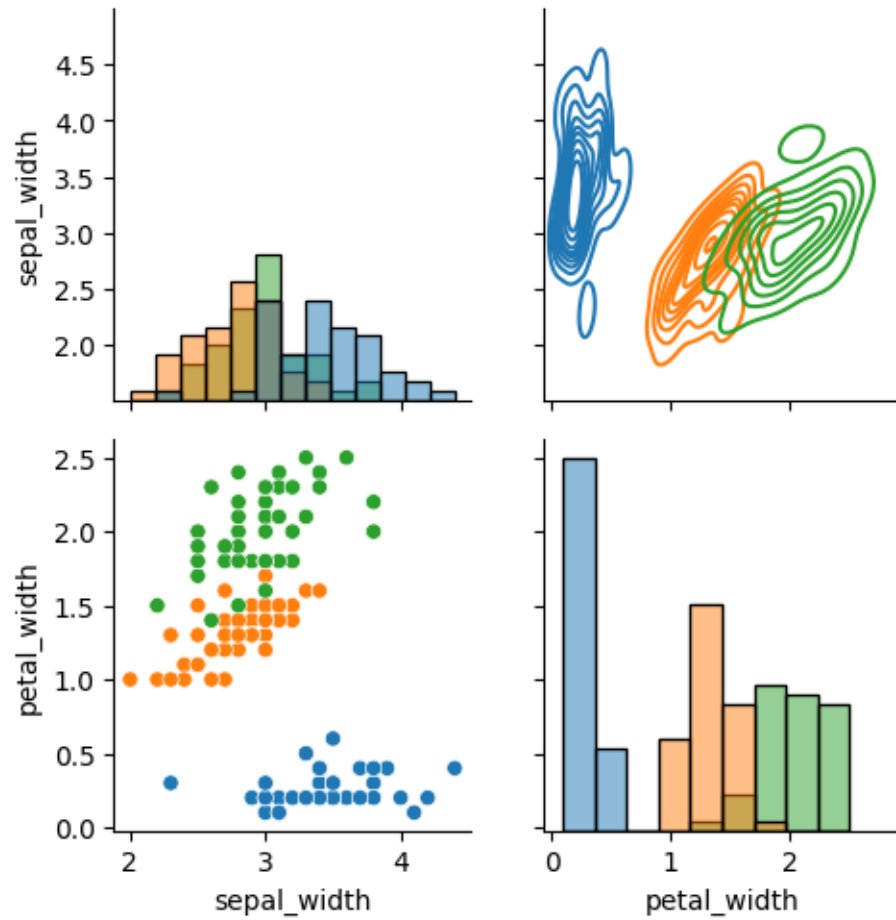
IndexError: cannot do a non-empty take from an empty axes.

```



```
[98]: # vars
g = sns.PairGrid(data=iris, hue='species', vars=['sepal_width', 'petal_width'])
g.map_diag(sns.histplot)
g.map_upper(sns.kdeplot)
g.map_lower(sns.scatterplot)
```

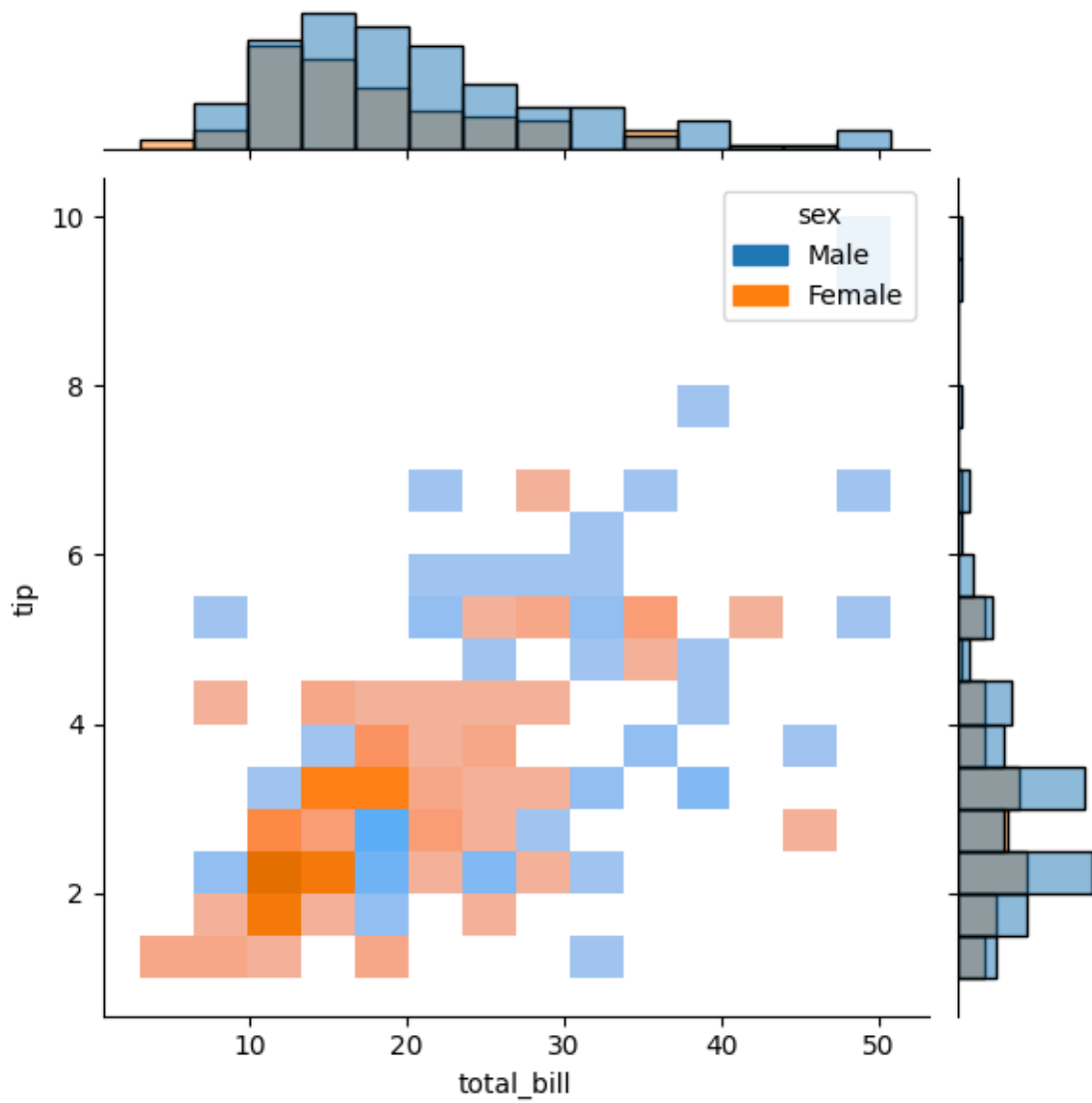
```
[98]: <seaborn.axisgrid.PairGrid at 0x250a056e690>
```



1.7 JointGrid Vs Jointplot

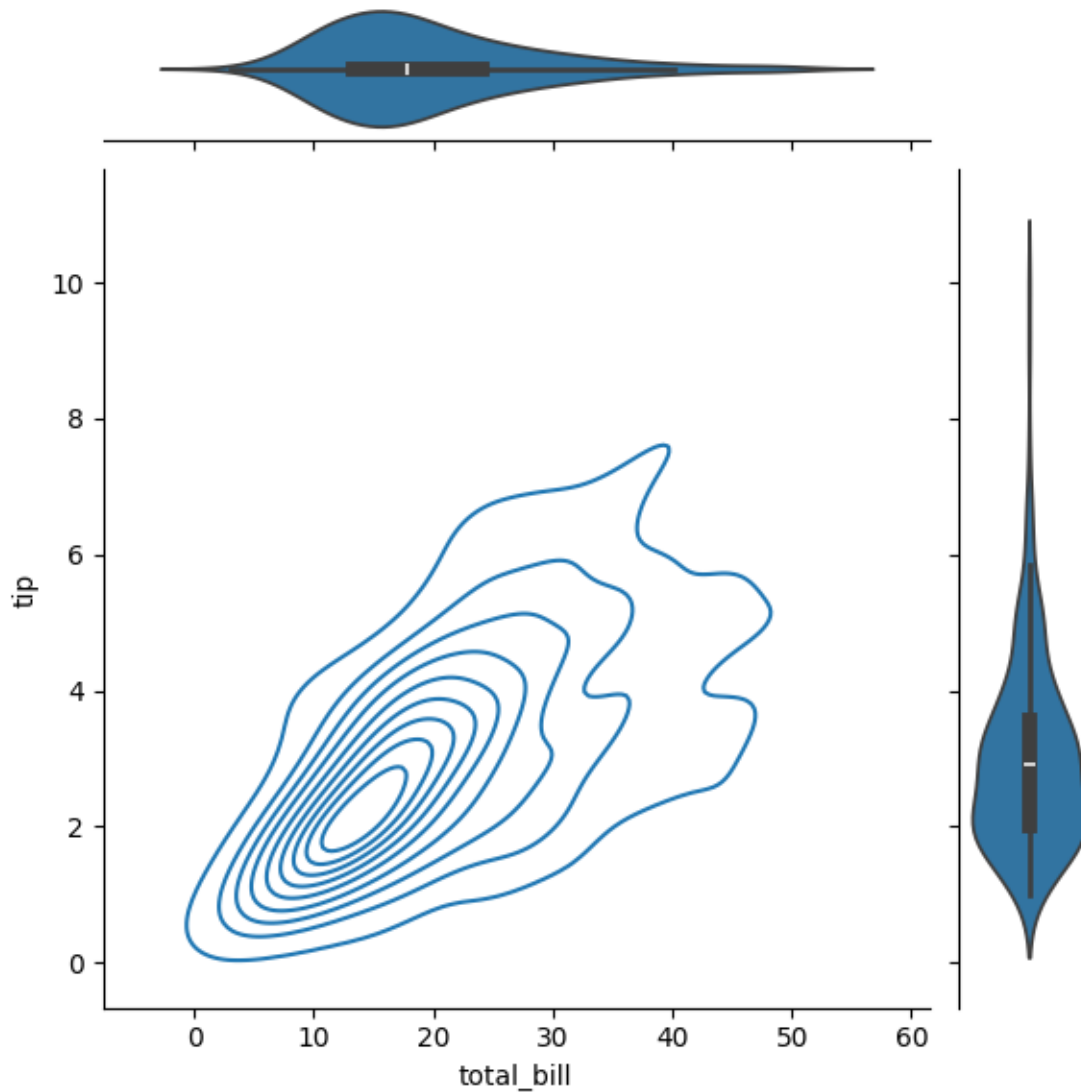
```
[99]: sns.jointplot(data=tips, x='total_bill', y='tip', kind='hist', hue='sex')
```

```
[99]: <seaborn.axisgrid.JointGrid at 0x250a0feca70>
```

```
[101]: g = sns.JointGrid(data=tips, x='total_bill', y='tip')
g.plot(sns.kdeplot, sns.violinplot)
```

```
[101]: <seaborn.axisgrid.JointGrid at 0x250a2aa20f0>
```



1.7.1 Utility Functions

```
[104]: sns.get_dataset_names()
```

```
-----
gaierror                                Traceback (most recent call last)
File c:\Program Files\Python312\Lib\urllib\request.py:1344, in_
  ↳ AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_args)
    1343 try:
-> 1344     h.request(req.get_method(), req.selector, req.data, headers,
    1345                  encode_chunked=req.has_header('Transfer-encoding'))
    1346 except OSError as err: # timeout error
```

```

File c:\Program Files\Python312\Lib\http\client.py:1336, in HTTPConnection.
    ↪ request(self, method, url, body, headers, encode_chunked)
    1335 """Send a complete request to the server."""
-> 1336 self._send_request(method, url, body, headers, encode_chunked)

```

```

File c:\Program Files\Python312\Lib\http\client.py:1382, in HTTPConnection.
    ↪ _send_request(self, method, url, body, headers, encode_chunked)
    1381     body = _encode(body, 'body')
-> 1382 self.endheaders(body, encode_chunked=encode_chunked)

```

```

File c:\Program Files\Python312\Lib\http\client.py:1331, in HTTPConnection.
    ↪ endheaders(self, message_body, encode_chunked)
    1330     raise CannotSendHeader()
-> 1331 self._send_output(message_body, encode_chunked=encode_chunked)

```

```

File c:\Program Files\Python312\Lib\http\client.py:1091, in HTTPConnection.
    ↪ _send_output(self, message_body, encode_chunked)
    1090 del self._buffer[:]
-> 1091 self.send(msg)
    1093 if message_body is not None:
    1094
    1095     # create a consistent interface to message_body

```

```

File c:\Program Files\Python312\Lib\http\client.py:1035, in HTTPConnection.
    ↪ send(self, data)
    1034 if self.auto_open:
-> 1035     self.connect()
    1036 else:

```

```

File c:\Program Files\Python312\Lib\http\client.py:1470, in HTTPSConnection.
    ↪ connect(self)
    1468 "Connect to a host on a given (SSL) port."
-> 1470 super().connect()
    1472 if self._tunnel_host:

```

```

File c:\Program Files\Python312\Lib\http\client.py:1001, in HTTPConnection.
    ↪ connect(self)
    1000 sys.audit("http.client.connect", self, self.host, self.port)
-> 1001 self.sock = self._create_connection(
    1002     (self.host,self.port), self.timeout, self.source_address)
    1003 # Might fail in OSs that don't implement TCP_NODELAY

```

```

File c:\Program Files\Python312\Lib\socket.py:829, in create_connection(address
    ↪ timeout, source_address, all_errors)
    828 exceptions = []
--> 829 for res in getaddrinfo(host, port, 0, SOCK_STREAM):
    830     af, socktype, proto, canonname, sa = res

```

```

File c:\Program Files\Python312\Lib\socket.py:964, in getaddrinfo(host, port,
↳family, type, proto, flags)
    963 addrlist = []
--> 964 for res in _socket.getaddrinfo(host, port, family, type, proto, flags):
    965     af, socktype, proto, canonname, sa = res

```

gaierror: [Errno 11001] getaddrinfo failed

During handling of the above exception, another exception occurred:

```

URLError                                Traceback (most recent call last)
Cell In[104], line 1
----> 1 sns.get_dataset_names()

```

```

File c:\Program Files\Python312\Lib\site-packages\seaborn\utils.py:499, in
↳get_dataset_names()
    493 def get_dataset_names():
    494     """Report available example datasets, useful for reporting issues.
    495
    496     Requires an internet connection.
    497
    498     """
--> 499     with urlopen(DATASET_NAMES_URL) as resp:
    500         txt = resp.read()
    502     dataset_names = [name.strip() for name in txt.decode().split("\n")]

```

```

File c:\Program Files\Python312\Lib\urllib\request.py:215, in urlopen(url, data,
↳timeout, cafile, capath, cadefault, context)
    213 else:
    214     opener = _opener
--> 215 return opener.open(url, data, timeout)

```

```

File c:\Program Files\Python312\Lib\urllib\request.py:515, in OpenerDirector.
↳open(self, fullurl, data, timeout)
    512     req = meth(req)
    514     sys.audit('urllib.Request', req.full_url, req.data, req.headers, req.
↳get_method())
--> 515 response = self._open(req, data)
    517 # post-process response
    518 meth_name = protocol+"_response"

```

```

File c:\Program Files\Python312\Lib\urllib\request.py:532, in OpenerDirector.
↳_open(self, req, data)
    529     return result
    531     protocol = req.type
--> 532 result = self._call_chain(self.handle_open, protocol, protocol +
    533                             '_open', req)

```

```
534 if result:
535     return result
```

File c:\Program Files\Python312\Lib\urllib\request.py:492, in OpenerDirector.

```
↪_call_chain(self, chain, kind, meth_name, *args)
    490 for handler in handlers:
    491     func = getattr(handler, meth_name)
--> 492     result = func(*args)
    493     if result is not None:
    494         return result
```

File c:\Program Files\Python312\Lib\urllib\request.py:1392, in HTTPSHandler.

```
↪https_open(self, req)
    1391 def https_open(self, req):
-> 1392     return self.do_open(http.client.HTTPSConnection, req,
    1393                          context=self._context)
```

File c:\Program Files\Python312\Lib\urllib\request.py:1347, in

```
↪AbstractHTTPHandler.do_open(self, http_class, req, **http_conn_args)
    1344         h.request(req.get_method(), req.selector, req.data, headers,
    1345                  encode_chunked=req.has_header('Transfer-encoding'))
    1346     except OSError as err: # timeout error
-> 1347         raise URLError(err)
    1348     r = h.getresponse()
    1349 except:
```

URLError: <urlopen error [Errno 11001] getaddrinfo failed>