Data Visualisation

Using seaborn & Panel

Content

Part 1

The basics

Design

1. Import the libraries

```
import matplotlib.pyplot as plt
import seaborn as sns
#for using seaborn you need to import
matplotlib as well
```

2. Define the figure size

```
fig = plt.figure(figsize=(8, 6))
```

3. Plot the graph

```
plt.plot(x, y)#matplotlib
sns.lineplot(x=x, y=y)#seaborn
```

4. Add the legend

```
Add at the plot label='Data' ->
sns.lineplot(x=x, y=y, label='Data')
plt.legend()
```

5. Specify color

```
Add at the plot color='red' ->
sns.lineplot(x=x, y=y, color='red')
```

6. Add the titles

```
plt.title('My Plot')
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
```

A complete list of color palettes.

Color - palettes

1. **Default**: The default color palette used by Seaborn is called colorblind, which is designed to be easily distinguishable for people with color vision deficiencies.
2. **Categorical**: Seaborn provides several categorical color palettes that are suitable for categorical data. These include deep, pastel, bright, dark, and muted.
3. **Sequential**: Sequential color palettes are useful for representing continuous data with a gradient of colors. Seaborn provides several sequential color palettes, such as rocket, magma, inferno, plasma, viridis, and cividis.
4. **Diverging**: Diverging color palettes are useful for representing data that has two distinct endpoints with a neutral middle point. Seaborn provides several diverging color palettes, such as coolwarm, vlag, PuOr, BrBG, and RdBu.

Set a custom palette

```
# Define a custom color palette
my_palette = sns.color_palette(['#FF0000',
'#00FF00', '#0000FF'])

# Set the custom color palette
sns.set_palette(my_palette)
```

Set a default palette

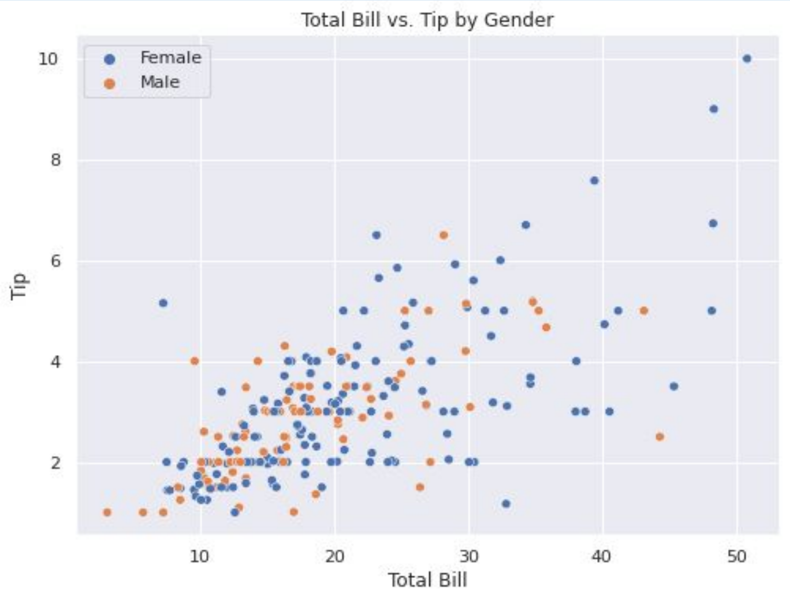
```
sns.set_palette('rocket')
```

Scatter plots

```python
import seaborn as sns
import matplotlib.pyplot as plt
# Load the tips dataset from Seaborn
tips = sns.load_dataset('tips')
# Set the figure size
sns.set(rc={'figure.figsize':(8,6)})
# Set the color palette
sns.set_palette('deep')
# Create the scatter plot
ax = sns.scatterplot(x='total_bill', y='tip', hue='sex',
data=tips)
# Set the title and axes labels
ax.set_title('Total Bill vs. Tip by Gender')
ax.set_xlabel('Total Bill')
ax.set_ylabel('Tip')
# Add a legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, ['Female', 'Male'])
# Show the plot
plt.show()
```

It's used to show the relationship between two continuous variables

```
seaborn.scatterplot(data=None, *, x=None, y=None,
hue=None, size=None, style=None, palette=None,
hue_order=None, hue_norm=None, sizes=None,
size_order=None, size_norm=None, markers=True,
style_order=None, legend='auto', ax=None,
**kwargs)
```

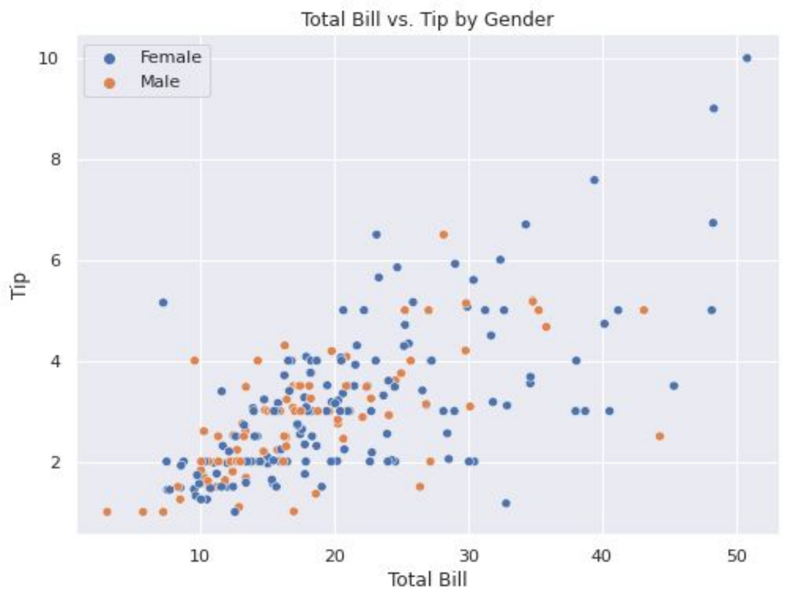


https://seaborn.pydata.org/generated/seaborn.scatterplot.html

It's used to show the relationship between two continuous variables

Scatter plots

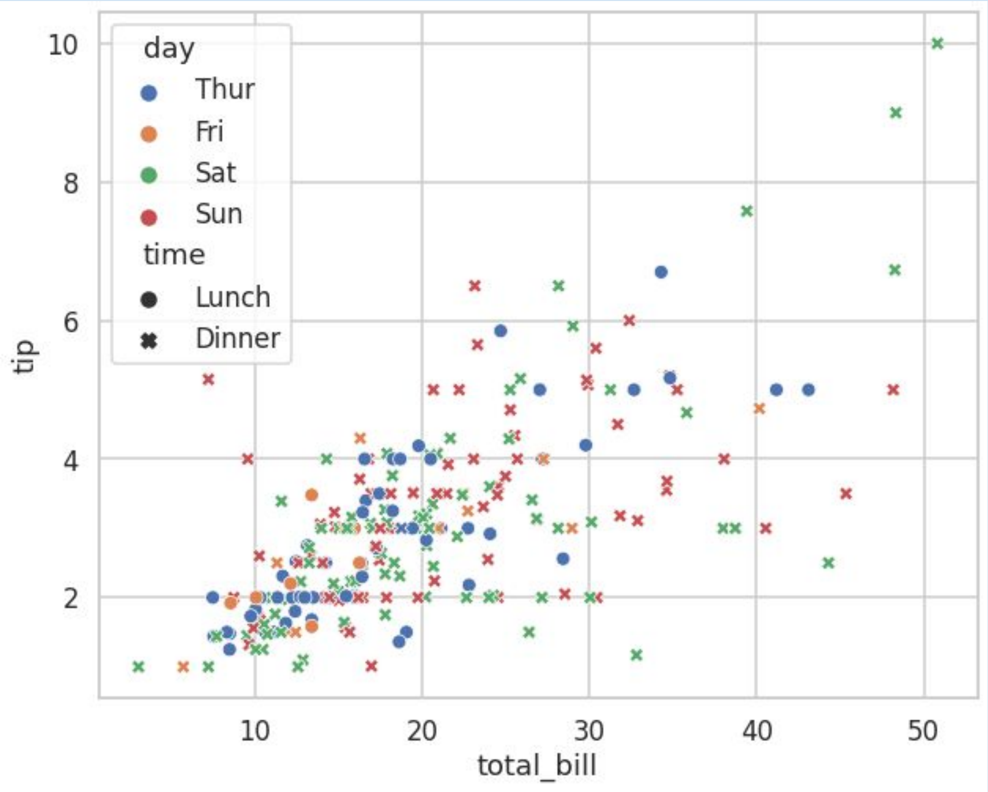
```
import seaborn as sns
import matplotlib.pyplot as plt
# Load the tips dataset from Seaborn
tips = sns.load_dataset('tips')
# Set the figure size
sns.set(rc={'figure.figsize':(8,6)})
# Set the color palette
sns.set_palette('deep')
# Create the scatter plot
ax = sns.scatterplot(x='total_bill', y='tip', hue='sex',
data=tips)
# Set the title and axes labels
ax.set_title('Total Bill vs. Tip by Gender')
ax.set_xlabel('Total Bill')
ax.set_ylabel('Tip')
# Add a legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, ['Female', 'Male'])
# Show the plot
plt.show()
```



It's used to show the relationship between two continuous variables

Scatter plots

```
tips = sns.load_dataset("tips")
fig = plt.figure(figsize=(10, 8))
sns.scatterplot(data=tips, x="total_bill",
y="tip", hue="day", style="time");
```



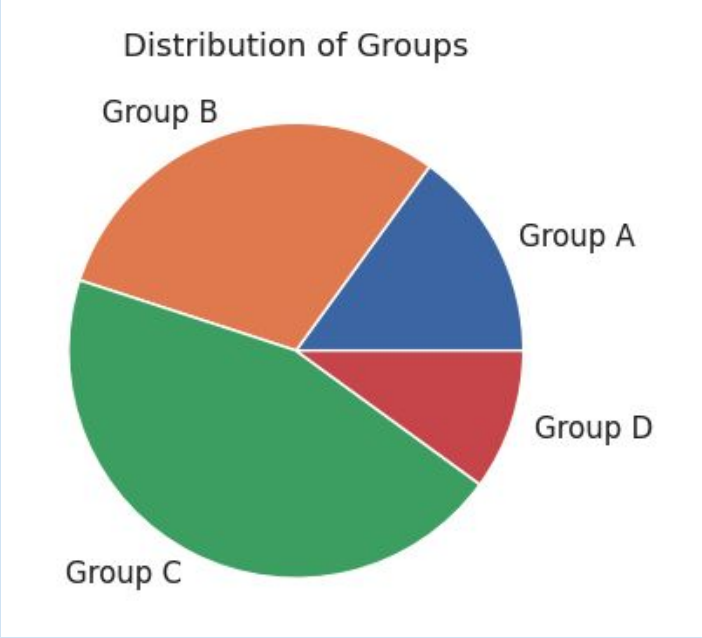
The relationship between x and y can be shown for different subsets of the data using the hue, size, and style parameters. These parameters control what visual semantics are used to identify the different subsets. It is possible to show up to three dimensions independently by using all three semantic types, but this style of plot can be hard to interpret and is often ineffective. Using redundant semantics (i.e. both hue and style for the same variable) can be helpful for making graphics more accessible.

Pie plots

It's often used to show the proportion of each category in the total data set.

Pie plots are done with matplotlib, but you could use palettes and more sophisticated features by seaborn

```python
import matplotlib.pyplot as plt
import seaborn as sns
# Create some data for the pie chart
sizes = [15, 30, 45, 10]
labels = ['Group A', 'Group B', 'Group C',
'Group D']
# Create the pie chart using Matplotlib
fig, ax = plt.subplots()
ax.pie(sizes, labels=labels)
# Add a title
ax.set_title('Distribution of Groups')
# Use Seaborn to style the plot
sns.set_style('white')
# Show the plot
plt.show()
```

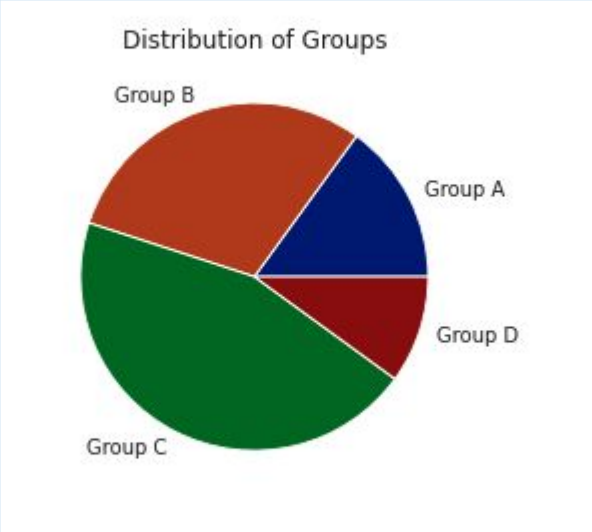


Pie plots

It's often used to show the proportion of each category in the total data set.

Pie plots are done with matplotlib, but you could use palettes and more sophisticated features by seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns
# Create some data for the pie chart
sizes = [15, 30, 45, 10]
labels = ['Group A', 'Group B', 'Group C', 'Group D']
# Create the pie chart using Matplotlib
fig, ax = plt.subplots()
palette_color = sns.color_palette('dark')
ax.pie(sizes,colors=palette_color, labels=labels)
# Add a title
ax.set_title('Distribution of Groups')
# Use Seaborn to style the plot
sns.set_style('white')
# Show the plot
plt.show()
```

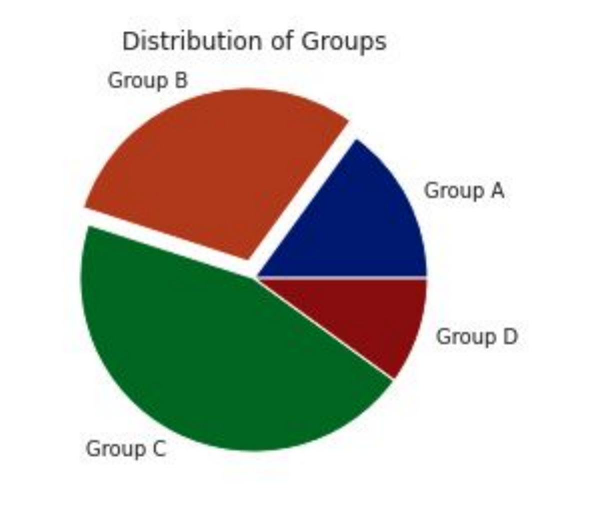


Pie plots

It's often used to show the proportion of each category in the total data set.

Pie plots are done with matplotlib, but you could use palettes and more sophisticated features by seaborn

```
import matplotlib.pyplot as plt
import seaborn as sns
# Create some data for the pie chart
sizes = [15, 30, 45, 10]
labels = ['Group A', 'Group B', 'Group C', 'Group D']
# Create the pie chart using Matplotlib
fig, ax = plt.subplots()
explode = [0, 0.1, 0, 0]
palette_color = sns.color_palette(dark')
ax.pie(sizes,colors=palette_color,explode=explode
,labels=labels)
# Add a title
ax.set_title('Distribution of Groups')
# Use Seaborn to style the plot
sns.set_style('white')
# Show the plot
plt.show()
```

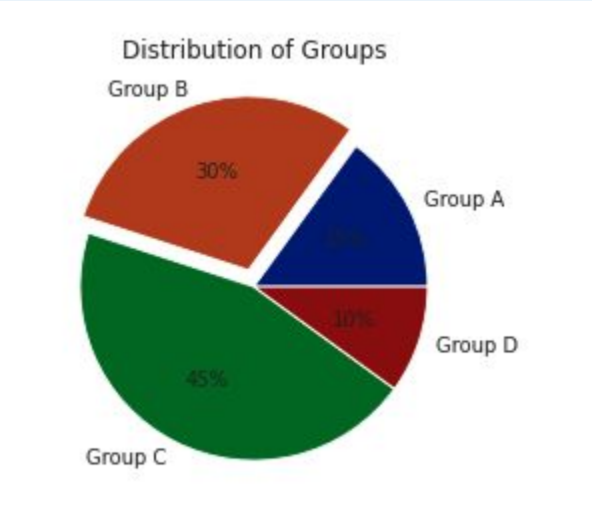


It's often used to show the proportion of each category in the total data set.

Pie plots

Pie plots are done with matplotlib, but you could use palettes and more sophisticated features by seaborn

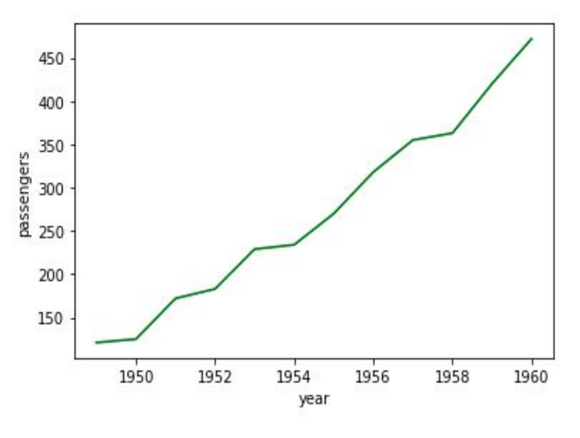
```
import matplotlib.pyplot as plt
import seaborn as sns
# Create some data for the pie chart
sizes = [15, 30, 45, 10]
labels = ['Group A', 'Group B', 'Group C', 'Group D']
# Create the pie chart using Matplotlib
fig, ax = plt.subplots()
explode = [0, 0.1, 0, 0]
palette_color = sns.color_palette('dark')
ax.pie(sizes,colors=palette_color,explode=explode
,labels=labels, autopct='%.0f%%')
# Add a title
ax.set_title('Distribution of Groups')
# Use Seaborn to style the plot
sns.set_style('white')
# Show the plot
plt.show()
```



It's often used to show trends over time (e.g. time series data)

Line plots

```
flights = sns.load_dataset('flights")
may_flights = flights.query('month == 'May'")
sns.lineplot(data=may_flights, x="year",
y="passengers",color='green');
```

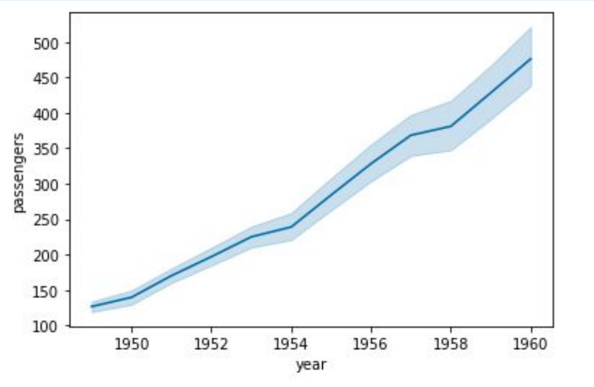


https://seaborn.pydata.org/generated/seaborn.lineplot.html

```
seaborn.lineplot(data=None, *, x=None, y=None,
hue=None, size=None, style=None, units=None,
palette=None, hue_order=None, hue_norm=None,
sizes=None, size_order=None, size_norm=None,
dashes=True, markers=None, style_order=None,
estimator='mean', errorbar=('ci', 95),
n_boot=1000, seed=None, orient='x', sort=True,
err_style='band', err_kws=None, legend='auto',
ci='deprecated', ax=None, **kwargs)
```

Passing the entire dataset in long-form mode will aggregate over repeated values (each year) to show the mean and 95% confidence interval:

```
sns.lineplot(data=flights, x="year",
y="passengers");
```



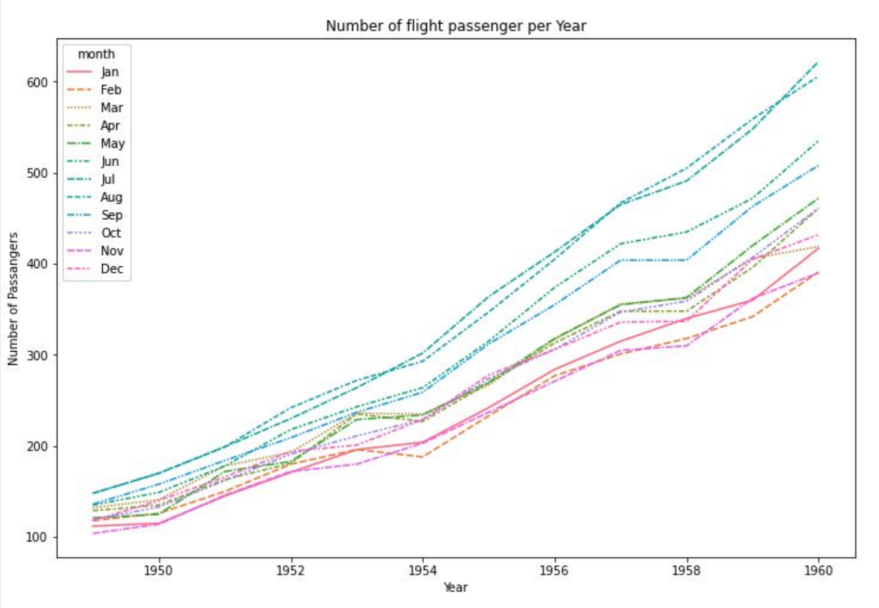
It's often used to show trends over time (e.g. time series data)

Line plots

```
plt.figure(figsize=(12,8))
flights_wide = flights.pivot("year", "month",
"passengers")
sns.lineplot(data=flights_wide);

# set title and axis labels
plt.title("Number of flight passenger per Year")
plt.xlabel("Year")
plt.ylabel("Number of Passangers")

# show the plot
plt.show()
```



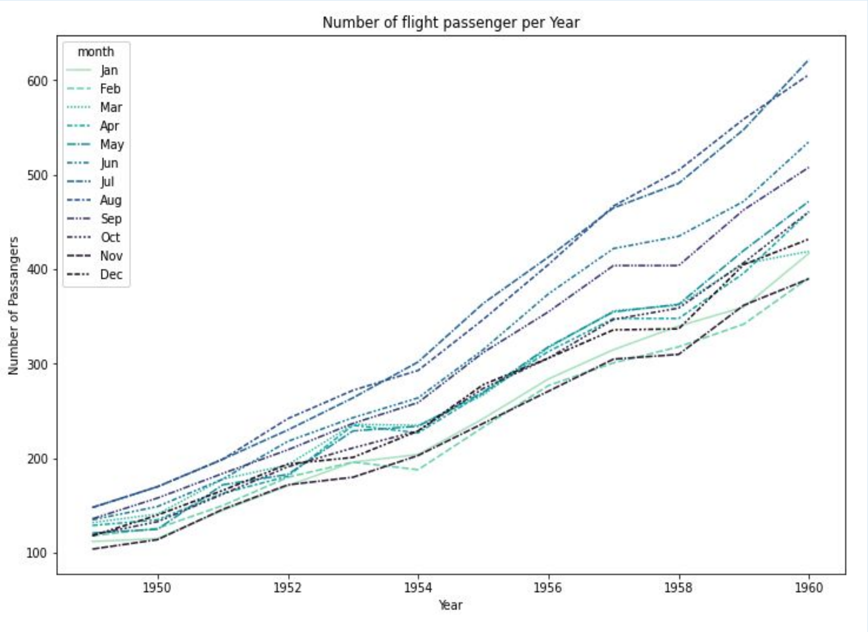
It's often used to show trends over time (e.g. time series data)

Line plots

```
palette = sns.color_palette('mako_r", 12)
plt.figure(figsize=(12,8))
sns.lineplot(data=flights_wide,palette=palette);

# set title and axis labels
plt.title("Number of flight passenger per Year")
plt.xlabel("Year")
plt.ylabel("Number of Passangers")

# show the plot
plt.show()
```



https://seaborn.pydata.org/generated/seaborn.histplot.html

visualizing the distribution of a single continuous variable. (e.g. identifying outliers)

Histograms

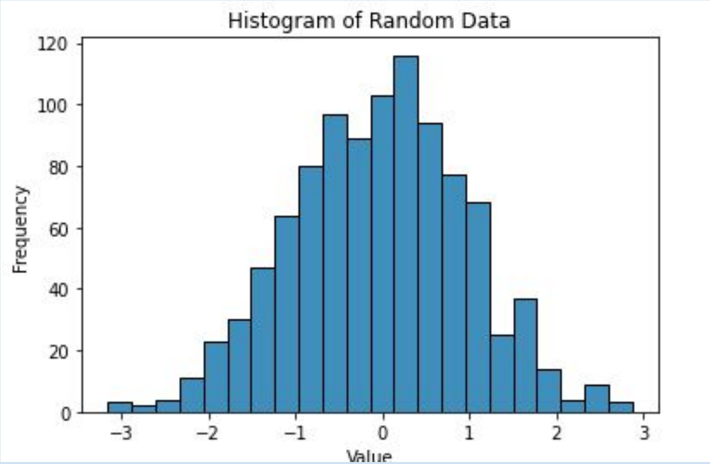
```
seaborn.histplot(data=None, *, x=None, y=None, hue=None, weights=None, stat='count', bins='auto', binwidth=None, binrange=None, discrete=None, cumulative=False, common_bins=True, common_norm=True, multiple='layer', element='bars', fill=True, shrink=1, kde=False, kde_kws=None, line_kws=None, thresh=0, pthresh=None, pmax=None, cbar=False, cbar_ax=None, cbar_kws=None, palette=None, hue_order=None, hue_norm=None, color=None, log_scale=None, legend=True, ax=None, **kwargs)
```

```
import numpy as np
# generate some random data
data = np.random.randn(1000)

# create histogram using Seaborn
sns.histplot(data=data)

# set title and axis labels
plt.title("Histogram of Random Data")
plt.xlabel("Value")
plt.ylabel("Frequency")

# show the plot
plt.show()
```



Histograms

visualizing the distribution of a single continuous variable. (e.g. identifying outliers)

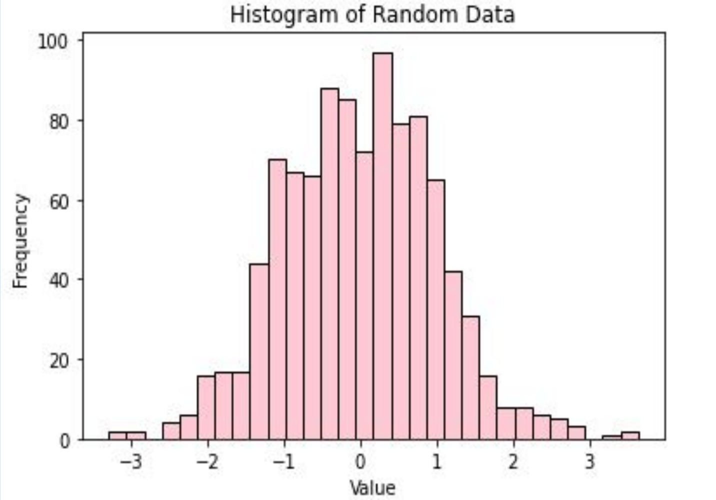
```
seaborn.histplot(data=None, *, x=None, y=None, hue=None,
weights=None, stat='count', bins='auto', binwidth=None,
binrange=None, discrete=None, cumulative=False,
common_bins=True, common_norm=True, multiple='layer',
element='bars', fill=True, shrink=1, kde=False,
kde_kws=None, line_kws=None, thresh=0, pthresh=None,
pmax=None, cbar=False, cbar_ax=None, cbar_kws=None,
palette=None, hue_order=None, hue_norm=None, color=None,
log_scale=None, legend=True, ax=None, **kwargs)
```

```python
import numpy as np
# generate some random data
data = np.random.randn(1000)

# create histogram using Seaborn
sns.histplot(data=data,color='pink',
bins=30)

# set title and axis labels
plt.title("Histogram of Random
Data")
plt.xlabel("Value")
plt.ylabel("Frequency")

# show the plot
plt.show()
```

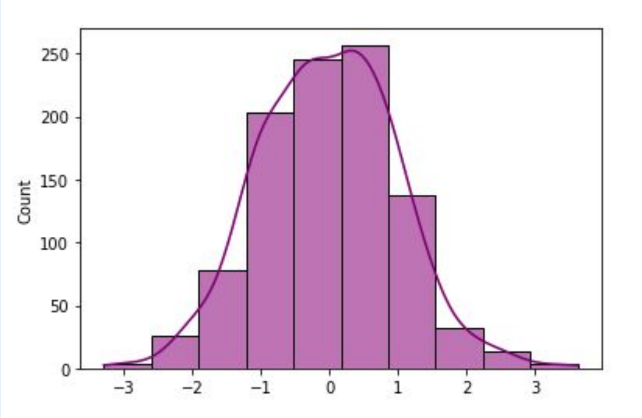


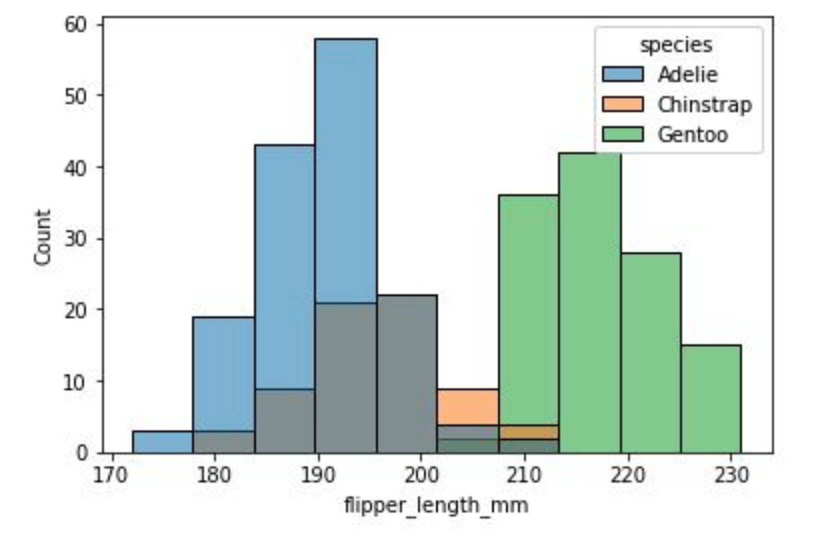
Histograms

visualizing the distribution of a single continuous variable. (e.g. identifying outliers)

```
#Add a kernel density estimate to smooth the histogram, providing complementary information about the shape of the distribution:

sns.histplot(data=data,color='purple',bins=10, kde=True);
```



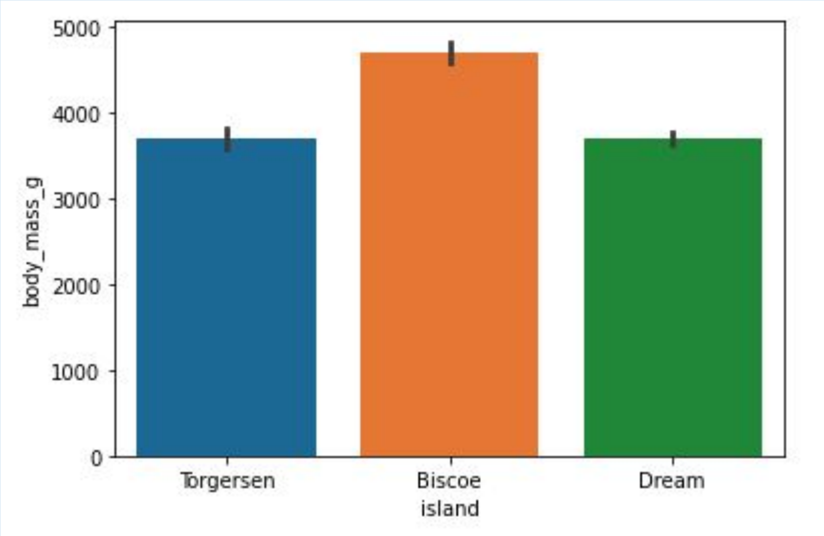



```
penguins = sns.load_dataset("penguins")
sns.histplot(data=penguins,
x="flipper_length_mm", hue="species");
```

Bar plots

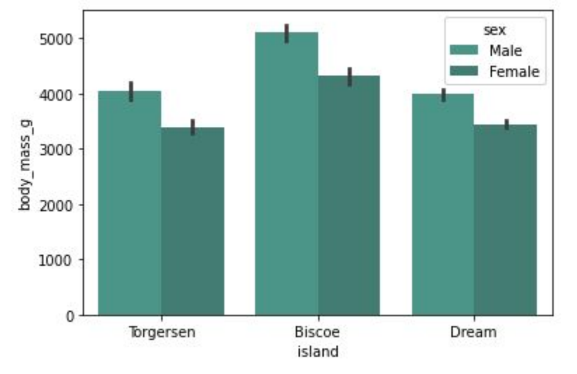
It's often used to compare the values of categorical variable across different groups or categories.

```
df = sns.load_dataset("penguins")
sns.barplot(data=df, x="island", y="body_mass_g");
```



https://seaborn.pydata.org/generated/seaborn.barplot.html

```
seaborn.barplot(data=None, *, x=None,
y=None, hue=None, order=None,
hue_order=None, estimator='mean',
errorbar=('ci', 95), n_boot=1000,
units=None, seed=None, orient=None,
color=None, palette=None, saturation=0.75,
width=0.8, errcolor='.26', errwidth=None,
capsize=None, dodge=True, ci='deprecated',
ax=None, **kwargs)
```




```
color=sns.color_palette("dark:#5A9_r")
sns.barplot(data=df, x="island", y="body_mass_g",
hue="sex",palette=color);
```

Box plots

It's often used to compare the values of continuous variable across different groups or categories.

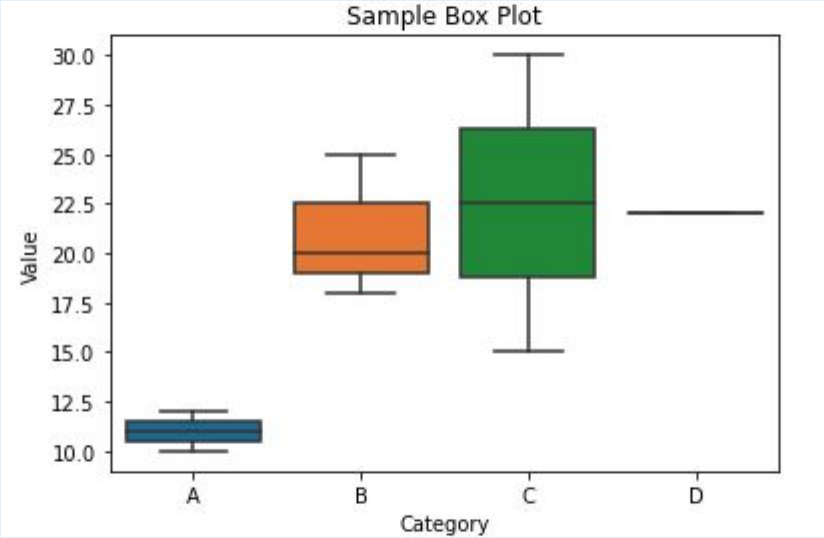
```
seaborn.boxplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, orient=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fliersize=5, linewidth=None, whis=1.5, ax=None, **kwargs)
```

```
# create some sample data
x = ['A', 'A', 'B', 'B', 'B', 'C', 'C', 'D']
y = [10, 12, 20, 18, 25, 15, 30, 22]

# create a box plot using Seaborn
sns.boxplot(x=x, y=y)

# set title and axis labels
plt.title("Sample Box Plot")
plt.xlabel("Category")
plt.ylabel("Value")

# show the plot
plt.show()
```

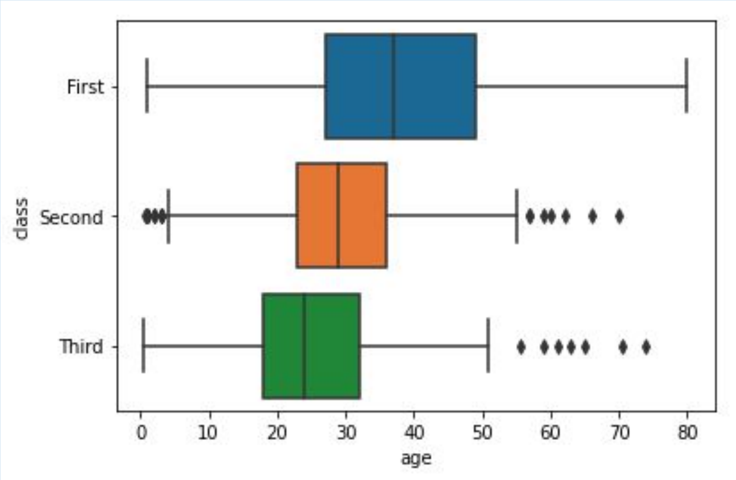


https://seaborn.pydata.org/generated/seaborn.boxplot.html

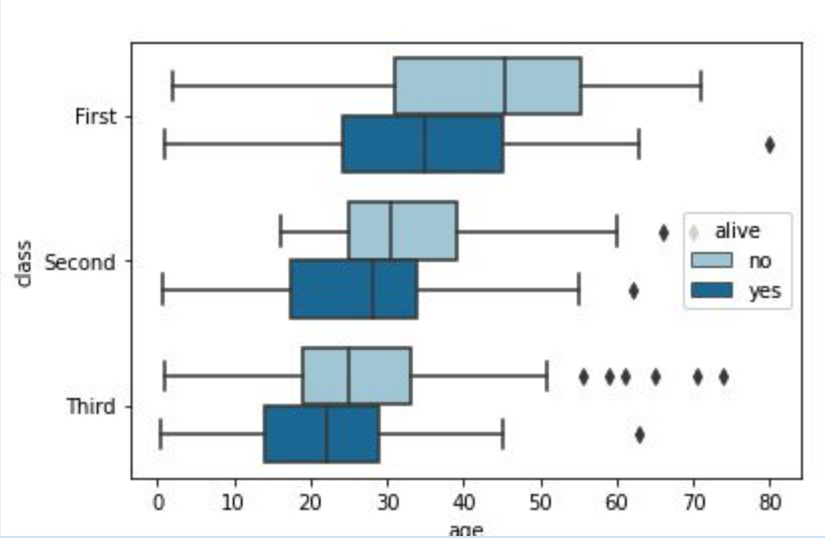
Box plots

It's often used to compare the values of continuous variable across different groups or categories.

```
df = sns.load_dataset("titanic")
sns.boxplot(data=df, x="age", y="class");
```




```
sns.boxplot(data=df, x="age", y="class",
hue="alive",palette="Paired");
```



https://seaborn.pydata.org/generated/seaborn.boxplot.html

Heatmaps

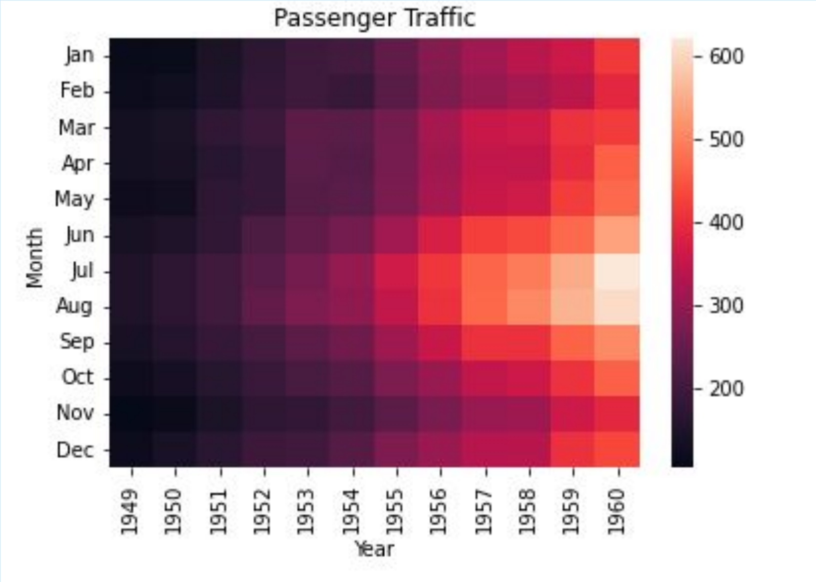
It's often used to show the distribution of a numerical variable across two dimension.

```
seaborn.heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, robust=False, annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, ax=None, **kwargs)
```

```
flights = sns.load_dataset('flights')
# reshape the data into a pivot table
flights_pivot = flights.pivot('month', 'year',
'passengers')

# create a heatmap using Seaborn
sns.heatmap(flights_pivot)

# set title and axis labels
plt.title("Passenger Traffic")
plt.xlabel("Year")
plt.ylabel("Month")
plt.show()
```



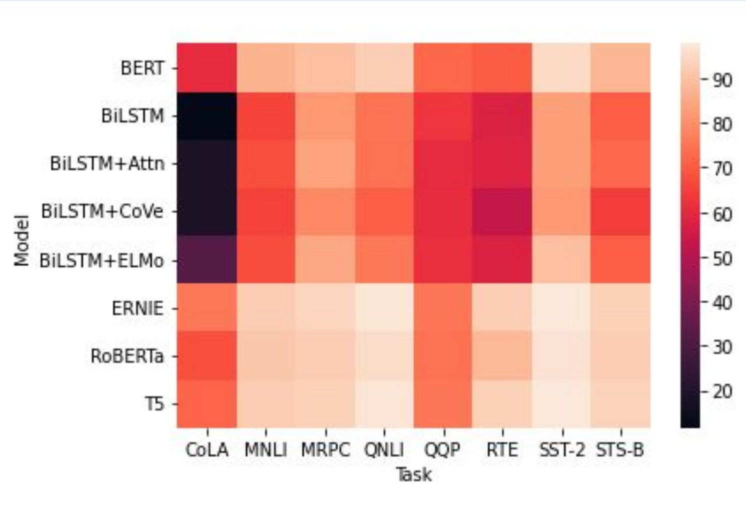
https://seaborn.pydata.org/generated/seaborn.heatmap.html

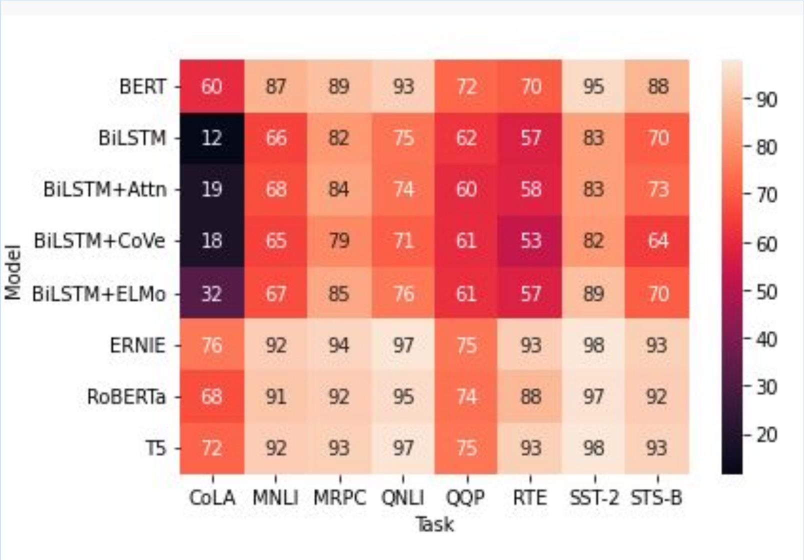
It's often used to show the distribution of a numerical variable across two dimension.

Heatmaps

```python
glue = sns.load_dataset("glue").pivot("Model", "Task", "Score")
sns.heatmap(glue);
```

```python
sns.heatmap(glue, annot=True);
```





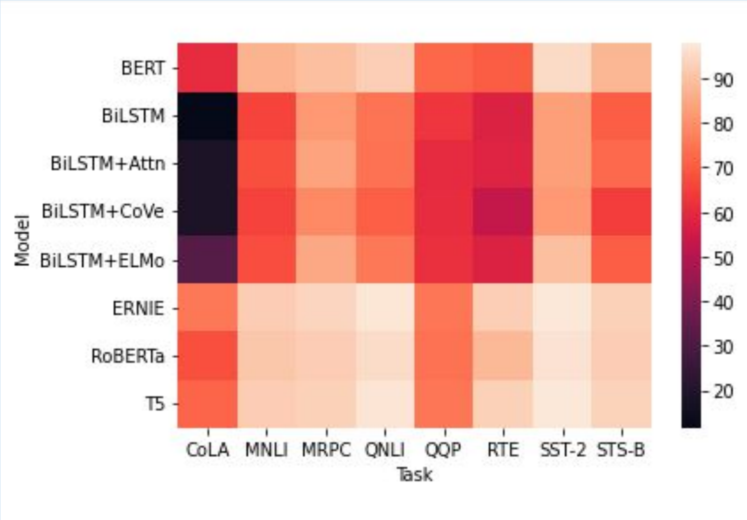
https://seaborn.pydata.org/generated/seaborn.heatmap.html

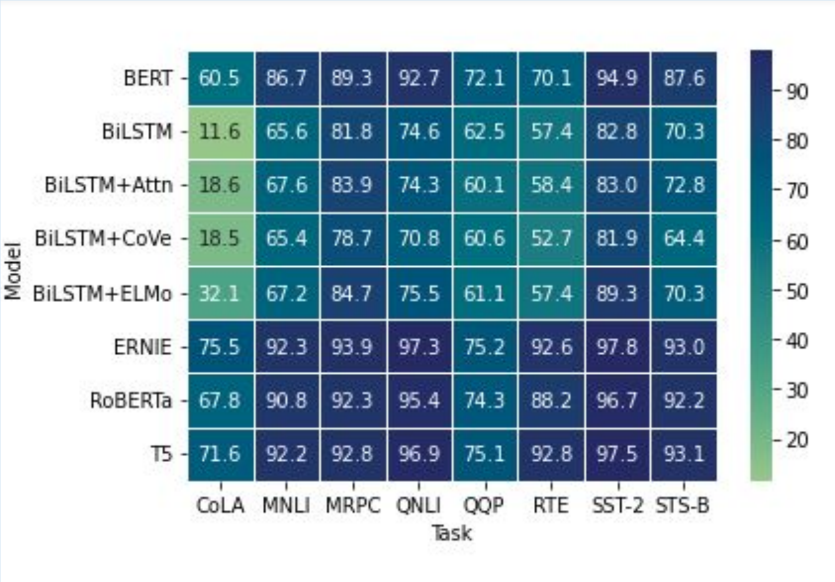
It's often used to show the distribution of a numerical variable across two dimension.

Heatmaps

```
glue = sns.load_dataset("glue").pivot("Model", "Task", "Score")
sns.heatmap(glue);
```

```
sns.heatmap(glue, annot=True, fmt=".1f",cmap="crest", linewidth=.5);
```





https://seaborn.pydata.org/generated/seaborn.heatmap.html

Part 2

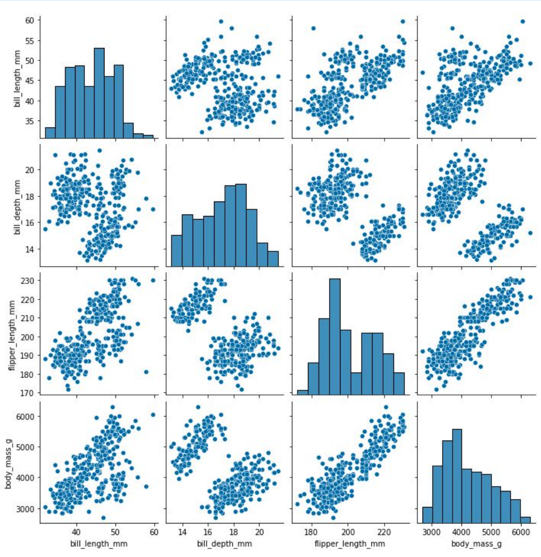
Off the beaten track

It's often used to visualize the pairwise relationship between multiple variable in a dataset.

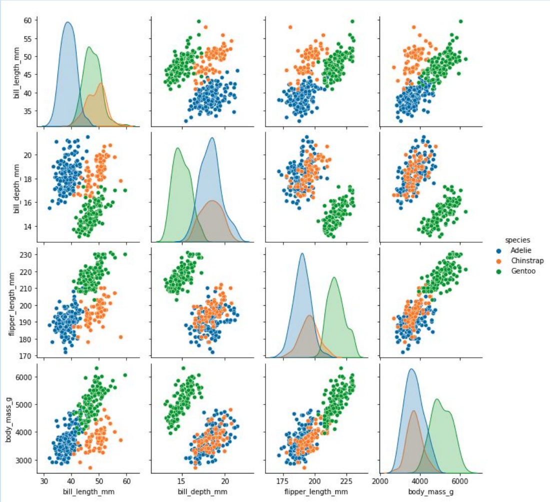
Pair Plots

```
seaborn.pairplot(data, *, hue=None, hue_order=None, palette=None, vars=None, x_vars=None, y_vars=None, kind='scatter', diag_kind='auto', markers=None, height=2.5, aspect=1, corner=False, dropna=False, plot_kws=None, diag_kws=None, grid_kws=None, size=None)
```

```
penguins = sns.load_dataset("penguins")
sns.pairplot(penguins);
```




```
sns.pairplot(penguins, hue="species");
```



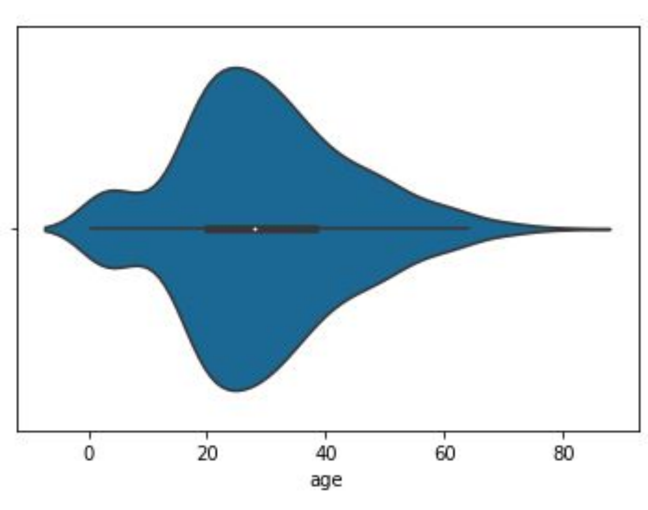
https://seaborn.pydata.org/generated/seaborn.pairplot.html

It's often used to visualize the distribution of numerical values, similar to a box plot but with a kernel density estimate

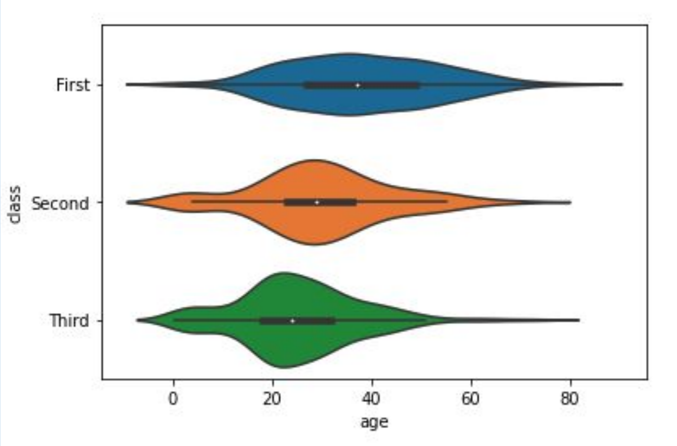
Violin Plots

```
df = sns.load_dataset("titanic")
sns.violinplot(x=df["age"]);
```

```
seaborn.violinplot(data=None, *, x=None, y=None,
hue=None, order=None, hue_order=None, bw='scott',
cut=2, scale='area', scale_hue=True, gridsize=100,
width=0.8, inner='box', split=False, dodge=True,
orient=None, linewidth=None, color=None, palette=None,
saturation=0.75, ax=None, **kwargs)
```




```
sns.violinplot(data=df, x="age", y="class");
```

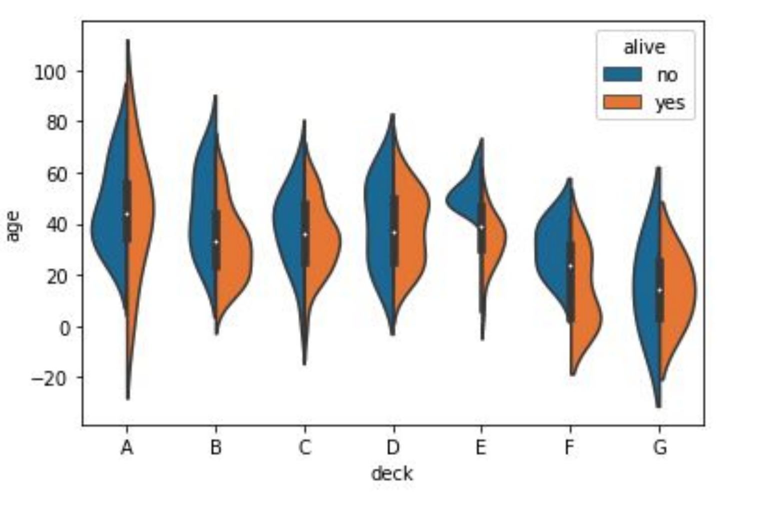


https://seaborn.pydata.org/generated/seaborn.violinplot.html

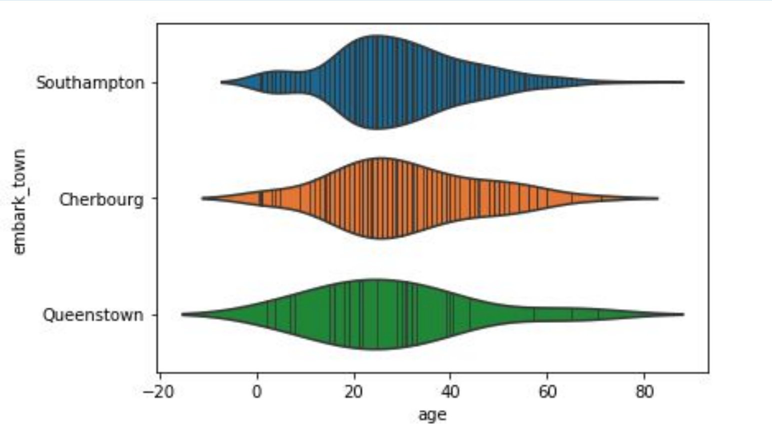
Violin Plots

It's often used to visualize the distribution of numerical values, similar to a box plot but with a kernel density estimate

```
sns.violinplot(data=df, x="deck", y="age",
hue="alive", split=True);
```




```
sns.violinplot(data=df, x="age", y="embark_town", inner="stick");
```



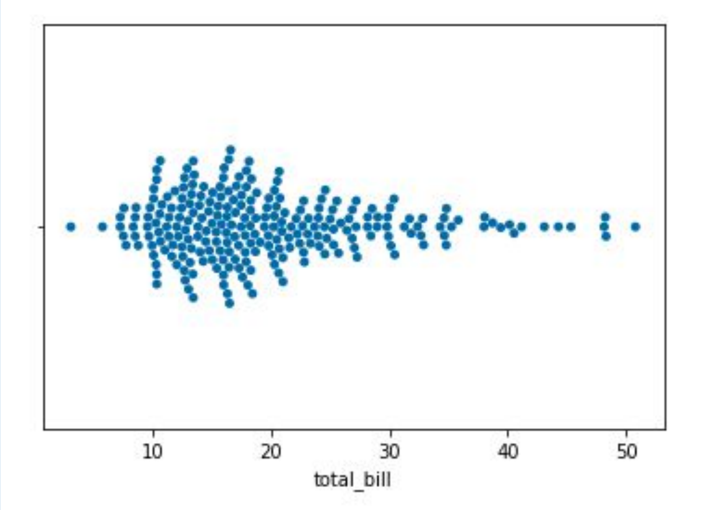
https://seaborn.pydata.org/generated/seaborn.violinplot.html

It's often used to visualize the distribution of numerical variable across categories, as scatter plot but with points adjusted to avoid overlap.

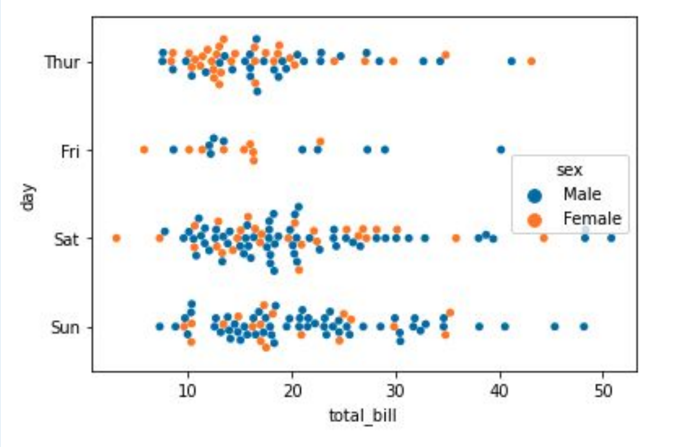
Swarm Plots

```
seaborn.swarmplot(data=None, *, x=None, y=None, hue=None, order=None, hue_order=None, dodge=False, orient=None, color=None, palette=None, size=5, edgecolor='gray', linewidth=0, hue_norm=None, native_scale=False, formatter=None, legend='auto', warn_thresh=0.05, ax=None, **kwargs)
```

```
tips = sns.load_dataset("tips")
sns.swarmplot(data=tips, x="total_bill");
```




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



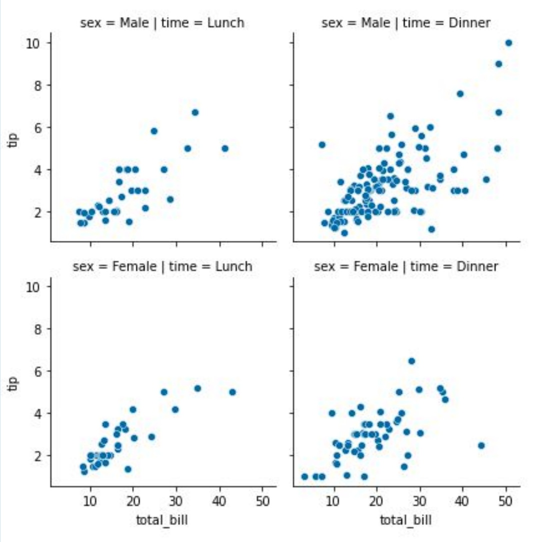
https://seaborn.pydata.org/generated/seaborn.swarmplot.html

Facet Grids

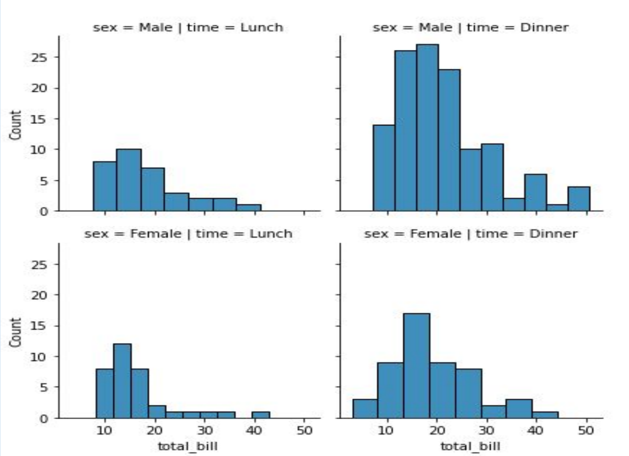
It's often used to create multiple plots for subsets of a dataset, based on one or more categorical variables.

```
class seaborn.FacetGrid(data, *, row=None, col=None,
hue=None, col_wrap=None, sharex=True, sharey=True,
height=3, aspect=1, palette=None, row_order=None,
col_order=None, hue_order=None, hue_kws=None,
dropna=False, legend_out=True, despine=True,
margin_titles=False, xlim=None, ylim=None,
subplot_kws=None, gridspec_kws=None)
```

```
tips = sns.load_dataset("tips")
g = sns.FacetGrid(tips, col="time",  row="sex")
g.map(sns.scatterplot, "total_bill", "tip");
```




```
g = sns.FacetGrid(tips, col="time",  row="sex")
g.map_dataframe(sns.histplot, x="total_bill");
```



https://seaborn.pydata.org/generated/seaborn.FacetGrid.html

Part 3

Build dashboards

Interactive dashboards with Panel

Panel is an open-source Python library that lets you create custom interactive web apps and dashboards by connecting user-defined widgets to plots, images, tables, or text.

```
import panel as pn
pn.extension('tabulator')
import hvplot.pandas
import seaborn as sns
import numpy as np
import pandas as pd
tips = sns.load_dataset("tips")
```

1

```
# Define panel widgets
bill_slider=pn.widgets.IntSlid
er(name='total_bill',
start=0,end=int(max(tips.total
_bill)),step=1, value=10)
bill_slider
```

2

```
tips=tips.interactive()
```

3

Make the data interactive.
Super Important!!!

Panel is an open-source Python library that lets you create custom interactive web apps and dashboards by connecting user-defined widgets to plots, images, tables, or text.

Interactive dashboards with Panel

```
# Create radio buttons
y_axis_tip=pn.widgets.RadioButtonGroup(
    name='Y axis',
    options=['tip'],
    button_type='success')
sex=['Female','Male']
```

4

```
#create the pipeline
tip_pipeline=(
    tips[(tips.total_bill<=bill_slider) &
(tips.sex.isin(sex))]

.groupby(['sex','total_bill'])[y_axis_tip].mean()
    .to_frame()
    .reset_index()
    .sort_values(by='total_bill')
    .reset_index(drop=True)
)
```

5

```
tip_plot=tip_pipeline.hvplot(x='total_bill',by='sex',y=y_axis_tip, title='bill vs tips')
```

6

```python
#Layout using Template
template = pn.template.FastListTemplate(
    title="Waiter's tips dashboard",
    sidebar=[pn.pane.Markdown("# Tips in restaurants"),
             pn.pane.Markdown("#### Food servers ….."),
             pn.pane.PNG('tips.png', sizing_mode='scale_both'),
             pn.pane.Markdown("## Settings"),
             bill_slider],
    main=[pn.Row(pn.Column(y_axis_tip,
                          tip_plot.panel(width=700), margin=(0,25)),
                 tip_table.panel(width=500)),
          pn.Row(pn.Column(tip_scatterplot.panel(width=600), margin=(0,25)),
                 pn.Column(yaxis_tip_source, tip_source_bar_plot.panel(width=600)))],
    accent_base_color="#88d8b0",
    header_background="#88d8b0",
)
#template.show()
template.servable();
```

Waiter's tips dashboard

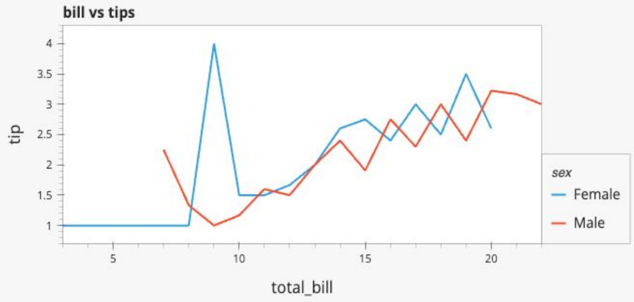
Tips in restaurants

Food servers tips in restaurants may be influenced by many factors, including the nature of the restaurant, size of the party, and table locations in the restaurant. Restaurant managers need to know which factors matter when they assign tables to food servers. For the sake of staff morale, they usually want to avoid either the substance or the appearance of unfair treatment of the servers, for whom tips (at least in restaurants in the United States) are a major component of pay.

Settings

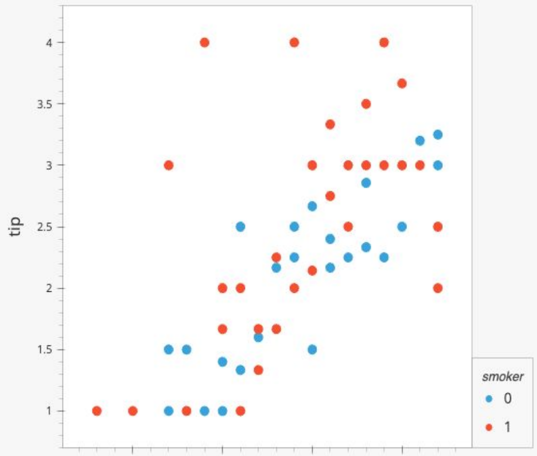
total_bill: 22

tip



index	sex	total_bill	tip
0	Male	3	NaN
1	Female	3	1.0
2	Male	5	NaN
3	Female	5	1.0
4	Male	7	2.25
5	Female	7	1.0
6	Female	8	1.0
7	Male	8	1.333333
8	Female	9	4.0
9	Male	9	1.0

First Prev 1 2 3 4 Next Last



smoker | time | total_bill