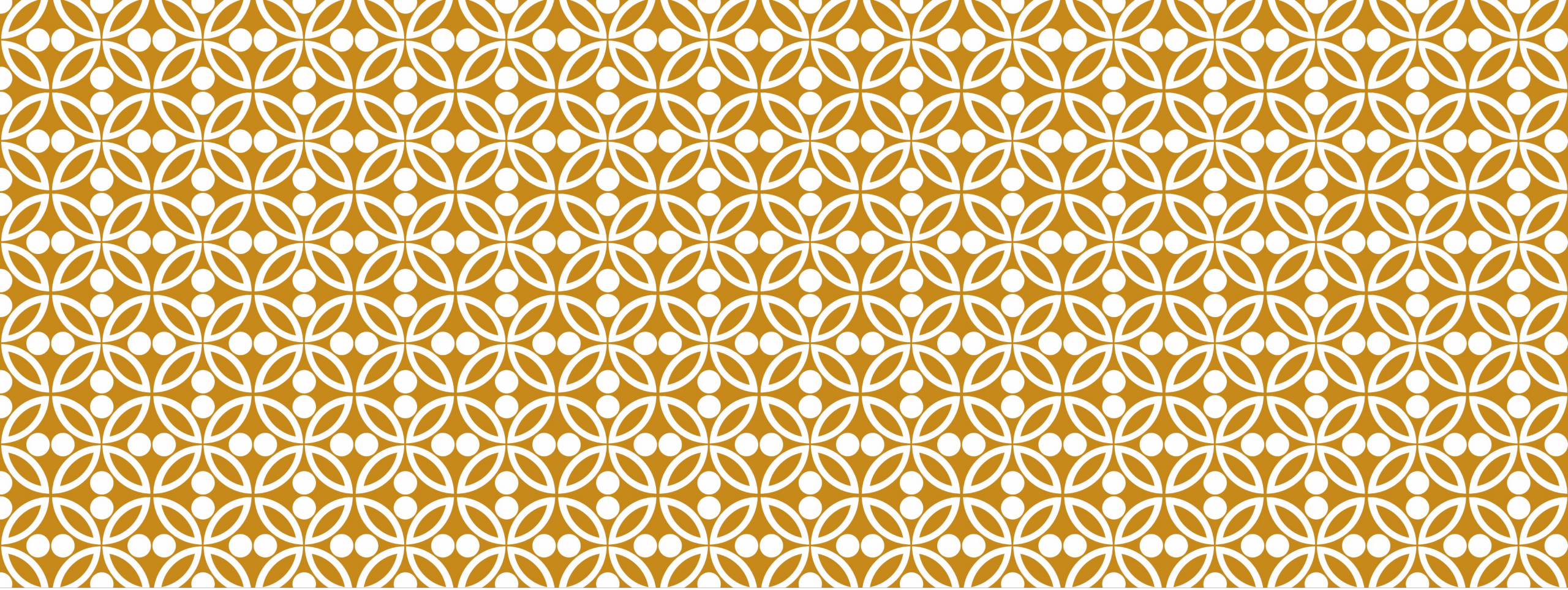


SEABORN (PYTHON) FOR DATA VISUALIZATION

Instructor: Somya Agrawal
Lab: 317

AGENDA

- Data visualization
- Python and data visualization
- Python libraries
  - Seaborn

ESSENTIAL LIBRARIES in Python

- NumPy
- Pandas
- Matplotlib
- Seaborn

PREREQUISITES

- Before directly jumping to the libraries, we need to build a strong core knowledge of what datatypes, lists, dictionaries, mutable or immutable objects are.
- It’s also needed to practice looping, be able to write functions, lambda functions and other basic built in functions.

KEY POINTS

- Python is a case sensitive language.
- Everything in Python is an object and objects have functions we call as methods that we can access using do notation.
- Every character in a string has a numeric representation in programming. You can use the built in function ord. So ord(“b”) is 98. So, 98 is the numeric representation of the letter b.

VSCODE TO IDE

To make VSCode into an IDE install the following extensions:

1. Python
2. Pylint
3. Formatter autopep8

STRUCTURE OF OOP

- Classes
- Objects
- Methods
- Attributes

https://www.youtube.com/watch?v=q2SGW2VgwAM

FOUR PILLARS OF OOP

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Reference video: https://www.youtube.com/watch?v=pTB0EiLXUC8

DATA VISUALIZATION THROUGH PYTHON

Lesson objective: To learn how to visualize data using Python.

Please can you think for a moment?

WHY DO WE NEED DATA VISUALIZATION?

WHY DO WE NEED DATA VISUALIZATION?

- Humans are visual creatures, and data visualization helps us quickly grasp complex datasets by presenting information in graphical or visual formats.
- It allows us to see patterns, trends, and relationships that might not be apparent from raw data alone.
- This helps in making informed decisions, predicting future outcomes, and discovering hidden insights that can drive business strategies or scientific research.

WHY DO WE NEED DATA VISUALIZATION?

- Visualizations provide an effective means of communicating insights and findings to others, whether they are stakeholders, decision-makers, or the general public.
- A well-designed visualization can convey a message or tell a story more clearly and convincingly than a textual or tabular presentation of data.

WHY DO WE NEED DATA VISUALIZATION?

- Data visualization allows us to explore relationships between different variables or dimensions of data.
- By plotting data points on graphs or charts, we can analyze how one variable affects another and gain insights into cause-and-effect relationships or dependencies.

WHY DO WE NEED DATA VISUALIZATION?

- Visualizations make it easier to spot anomalies, outliers, or errors in the data that may require further investigation.
- For example, scatter plots or box plots can highlight data points that deviate significantly from the norm, helping to detect errors or unusual patterns in the data.

WHY DO WE NEED DATA VISUALIZATION?

- Data visualization facilitates data-driven decision-making by providing decision-makers with clear, actionable insights derived from data analysis.
- Visualizations help stakeholders understand complex issues, evaluate different scenarios, and make informed choices based on evidence rather than intuition or guesswork.

WHY DO WE NEED DATA VISUALIZATION?

- Visualizations enhance storytelling by making presentations more engaging, memorable, and persuasive.
- Whether it's in business meetings, academic presentations, or journalistic reports, visualizations help captivate the audience's attention and convey information in a compelling manner.

In summary

“Data visualization is crucial for understanding data, communicating insights, identifying trends, exploring relationships, detecting anomalies, making decisions, and enhancing storytelling.

It plays a vital role in data analysis, decision-making processes, and effective communication of information in various domains, including business, science, academia, and journalism.”

TYPES OF DATA VISUALIZATION

Plotting libraries:

- Matplotlib: low level, provides lots of freedom
- Pandas visualization: easy to use interface, built on Matplotlib
- Seaborn: high level interface, great default styles
- ggplot: based on R’s ggplot2
- Plotly: can create interactive plots

USING SEABORN LIBRARY

In this lesson we will learn how to use Seaborn library in Python to visualize data.

USING SEABORN LIBRARY

- Seaborn is a Python data visualization library based on matplotlib.
- It provides a high-level interface for drawing attractive and informative statistical graphics.
- Seaborn is built on top of matplotlib and closely integrated with pandas data structures.
- It is designed to work well with DataFrame objects in pandas, making it particularly useful for visualizing datasets in exploratory data analysis.

USING SEABORN LIBRARY

- Seaborn was developed by Michael Waskom and was first released in 2012.
- It was created to address the limitations and complexities of Matplotlib, which required significant code to generate visually appealing plots.

https://mwaskom.github.io/

Dependency of Seaborn

NumPy – to work with arrays

Matplotlib – already covered in the class

Pandas – to store, to filter

SciPy – analyzes data first and then visualizes it

INSTALLING SEABORN

- Open the terminal/ command prompt window depending on the OS.
- Use the following statement:

```
pip install seaborn
Conda install seaborn # to install in Anaconda
pip install matplotlib
```

INSTALLING SEABORN

- You need to import this library in Python. Use the following statements.

```
import matplotlib.pyplot as plt
OR
from matplotlib import pyplot as plt
And
import seaborn as sns
```

- Here plt and sns are just aliases. If we don't use alias then we will have to type the whole statement.
- Similar concept to SQL.

Note: Here pyplot is a class which helps you to make graphs

CLASSES IN PYTHON

- In Python, classes are a fundamental concept of object-oriented programming (OOP).
- They serve as a blueprint for creating objects, which are instances of the class.
- Classes encapsulate data (attributes) and behaviors (methods) into a single unit, allowing for modular and organized code.
- Example: Human, characteristics of human beings

CLASSES IN PYTHON

- **Class Definition:** A class is defined using the `class` keyword, followed by the class name and a colon. Inside the class definition, you define attributes and methods.

```
class MyClass:
    # Class attributes and methods are defined here
    pass
```

CLASS ATTRIBUTES IN PYTHON

Attributes: Attributes are variables that hold data associated with a class or its objects. They represent the state of an object. Attributes can be either class attributes (shared among all instances) or instance attributes (unique to each instance).

```
class Person:
    # Class attribute
    species = "Homo sapiens"


    def __init__(self, name, age):
        # Instance attributes
        self.name = name
        self.age = age
```

Dictionaries in Python

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionaries are written with curly brackets, and have keys and values.
- Dictionary items are ordered, changeable, and do not allow duplicates.
- Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Reference: https://www.w3schools.com/python/python_dictionaries.asp

METHODS IN PYTHON

Methods: Methods are functions defined within a class.

They define the behavior of the class and can access and modify its attributes.

Methods can be instance methods, class methods, or static methods.

```python
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

    @classmethod
    def square(cls, side_length):
        return cls(side_length, side_length)

    @staticmethod
    def is_square(rect):
        return rect.width == rect.height
```

JUPYTER NOTEBOOK

- Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.
- It supports various programming languages, but it's particularly popular in the Python community.
- Jupyter Notebook provides an interactive computing environment where you can write, execute, and document code in an interactive and collaborative manner.

JUPYTER NOTEBOOK

- **Support for Multiple Programming Languages:** While commonly used with Python, Jupyter Notebook supports over 40 different programming languages, including R, Julia, and Scala. Each notebook can have a kernel associated with a specific language, allowing you to write code in that language.
- **Visualization Capabilities:** Jupyter Notebook integrates with various visualization libraries, such as Matplotlib, Seaborn, Plotly, and Bokeh, allowing you to create interactive plots, charts, and graphs directly within your notebooks. This makes data exploration and presentation more convenient and effective.

JUPYTER NOTEBOOK

- **Notebook Sharing and Collaboration:** You can share Jupyter Notebooks with others by exporting them as HTML, PDF, or other formats, or by hosting them on platforms like GitHub, JupyterHub, or Jupyter Notebooks Viewer. Additionally, Jupyter supports collaborative editing, allowing multiple users to work on the same notebook simultaneously.

INSTALLING JUPYTER NOTEBOOK

To install Jupyter notebook, we will use Anaconda environment.

Steps:

1. Install Python
2. Check it on terminal if/ which version of Python exists
3. Download Anaconda
4. Launch Jupyter notebook (it will run in your default browser)

Reference video link for how to install Anaconda and then launching Jupyter notebook: https://www.youtube.com/watch?v=WUeBzT43JyY

Reference link for downloading Anaconda: https://www.anaconda.com/download

INTRODUCING ANACONDA

- Anaconda is a popular open-source distribution of the Python and R programming languages for scientific computing, data science, and machine learning.
- It includes a collection of over 1,500 packages and libraries commonly used in these fields, along with tools for managing environments, dependencies, and packages.
- Anaconda simplifies the process of setting up and managing software environments for data analysis, allowing users to focus on their work rather than worrying about software compatibility and installation issues.

INTRODUCING ANACONDA

- Anaconda provides access to a vast collection of packages and libraries commonly used in scientific computing, data analysis, machine learning, and related fields.
- This includes popular libraries such as NumPy, pandas, Matplotlib, SciPy, scikit-learn, TensorFlow, and PyTorch, among others.

INTRODUCING ANACONDA

- **Cross-Platform Support:** Anaconda is available for Windows, macOS, and Linux operating systems, making it suitable for a wide range of users and environments.
- **Integrated Development Environments (IDEs):** Anaconda can be used with various integrated development environments (IDEs) and text editors, including Jupyter Notebook, JupyterLab, Spyder, and VSCode. These environments provide features such as code editing, debugging, and interactive visualization, enhancing the productivity of users.

CHECKLIST:

1. Install Python
2. Check it on terminal if/ which version of Python exists
3. Download Anaconda
4. Launch Jupyter notebook (it will run in your default browser)

Reference video link for how to install Anaconda and then launching Jupyter notebook: https://www.youtube.com/watch?v=WUeBzT43JyY

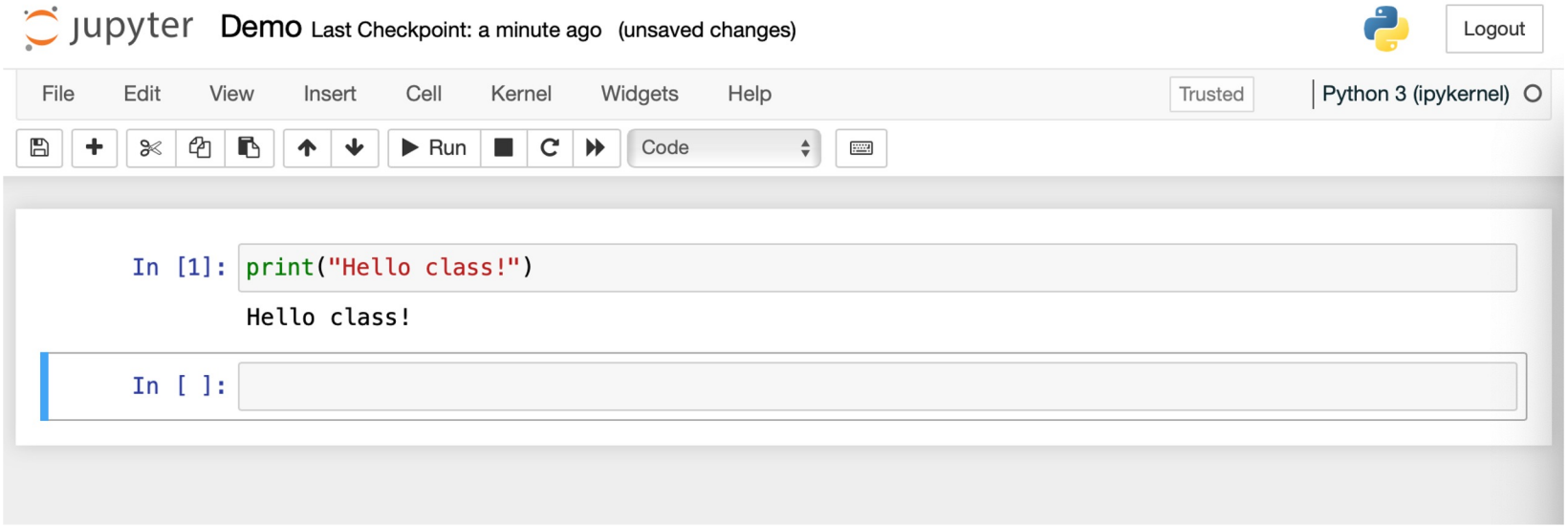
Reference link for downloading Anaconda: https://www.anaconda.com/download

GETTING STARTED WITH JUPYTER NOTEBOOK

- Click on “New” on the right hand side of the Jupyter interface.
- A new notebook will be created where you will write the Python code in the cells.
- Rename the file. Maybe call it “Demo”.
- Explore the interface.
- Please note, to run a Python code, an interpreter is needed.
- The interpreter in Jupyter notebook is called “Kernel”.

GETTING STARTED WITH JUPYTER NOTEBOOK

Write the following piece of code to test if the notebook is working!



GETTING STARTED WITH JUPYTER NOTEBOOK

- Click on the ‘+’ sign to add new cells to write code.
- The number beside **In** in every cell represents the number of times you have run the file.

INSTALLING LIBRARIES

- There are several ways of installing Matplotlib library on the computer.
- We will install Matplotlib library on Jupyter notebook using Anaconda.
- Steps:
  1. Create a new notebook called “install_matplotlib” on Jupyter. Save it.
  2. In the first cell type: **!pip install matplotlib**
  3. Click on run.
  4. If successful, then you will get an interface as follows.

Note: To install any Python library package, we need pip.

INSTALLING MATPLOTLIB LIBRARY

```
In [1]: !pip install matplotlib

Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.11/site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.0.
5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (4.2
5.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.
4.4)
Requirement already satisfied: numpy>=1.20 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (23.1)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/python3.11/site-packages (from matplotlib)
(2.8.2)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.7->mat
plotlib) (1.16.0)
```

INSTALLING LIBRARIES

- There are several ways of installing Matplotlib library on the computer.
- We will install Matplotlib library on Jupyter notebook using Anaconda.
- Steps:
  1. Use the same notebook called “install_matplotlib” on Jupyter. Save it.
  2. In the first cell type: **!pip install seaborn**
  3. Click on run.
  4. If successful, then you will get an interface as follows.

Note: To install any Python library package, we need pip.

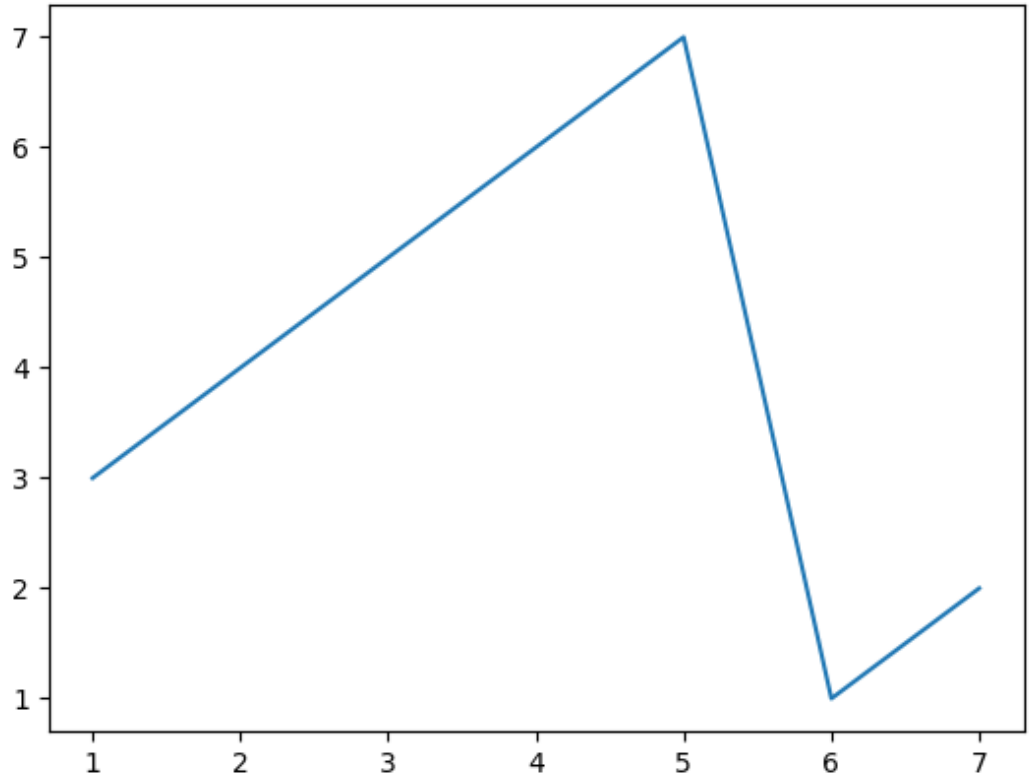
INSTALLING SEABORN LIBRARY

```
In [2]: !pip install seaborn
```

```
Requirement already satisfied: seaborn in ./anaconda3/lib/python3.11/site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in ./anaconda3/lib/python3.11/site-packages (from seaborn) (1.2
4.3)
Requirement already satisfied: pandas>=0.25 in ./anaconda3/lib/python3.11/site-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in ./anaconda3/lib/python3.11/site-packages (from seaborn)
(3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.
1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=
3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.
1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.
1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.1,
>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=
3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=
3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/python3.11/site-packages (from matplotlib!=
3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/site-packages (from pandas>=0.25->seabor
n) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in ./anaconda3/lib/python3.11/site-packages (from pandas>=0.25->seabo
rn) (2023.3)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.7->mat
plotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
```

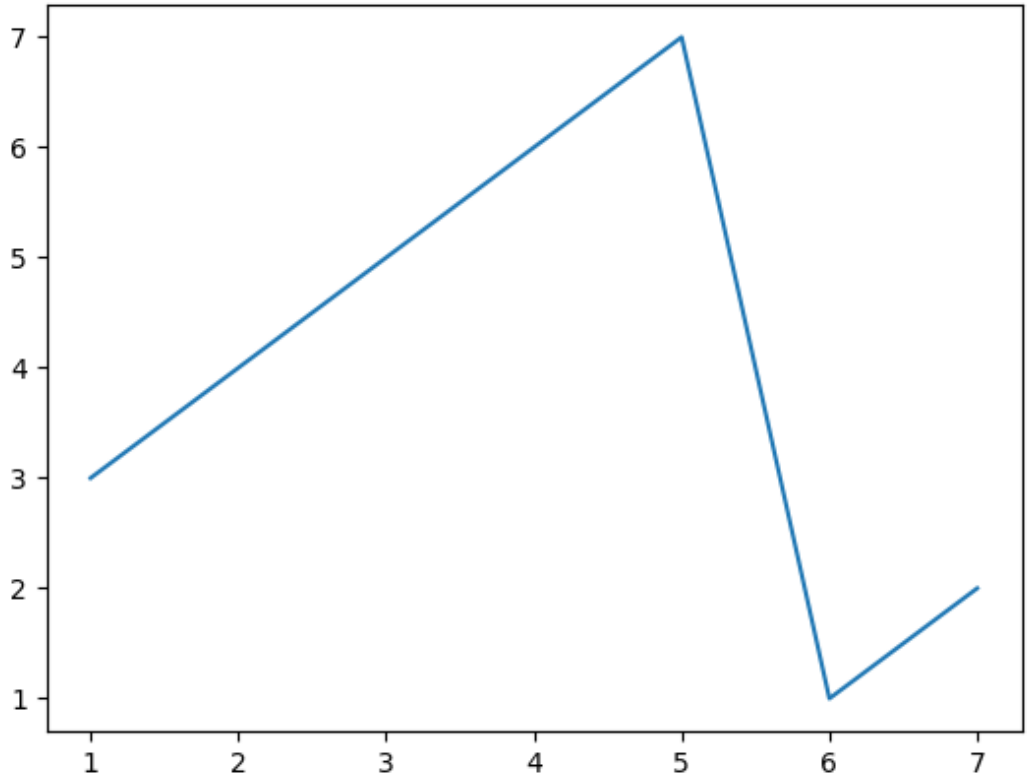
LINE PLOT using Matplotlib

```
!pip install matplotlib
!pip install seaborn
import matplotlib.pyplot as plt
import seaborn as sns
var = [1,2,3,4,5,6,7]
var_1 = [3,4,5,6,7,1,2]
plt.plot(var,var_1)
plt.show()
```



LINE PLOT using Seaborn

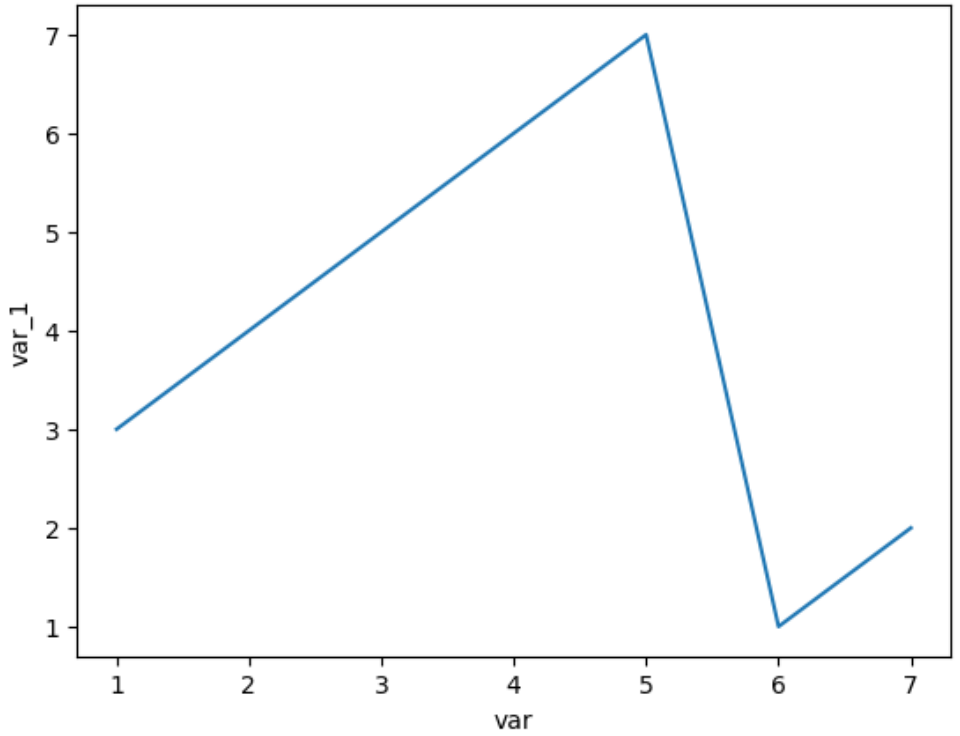
- Seaborn usually works on a dataset, so the code will have some changes as we will see next.



Dataframes

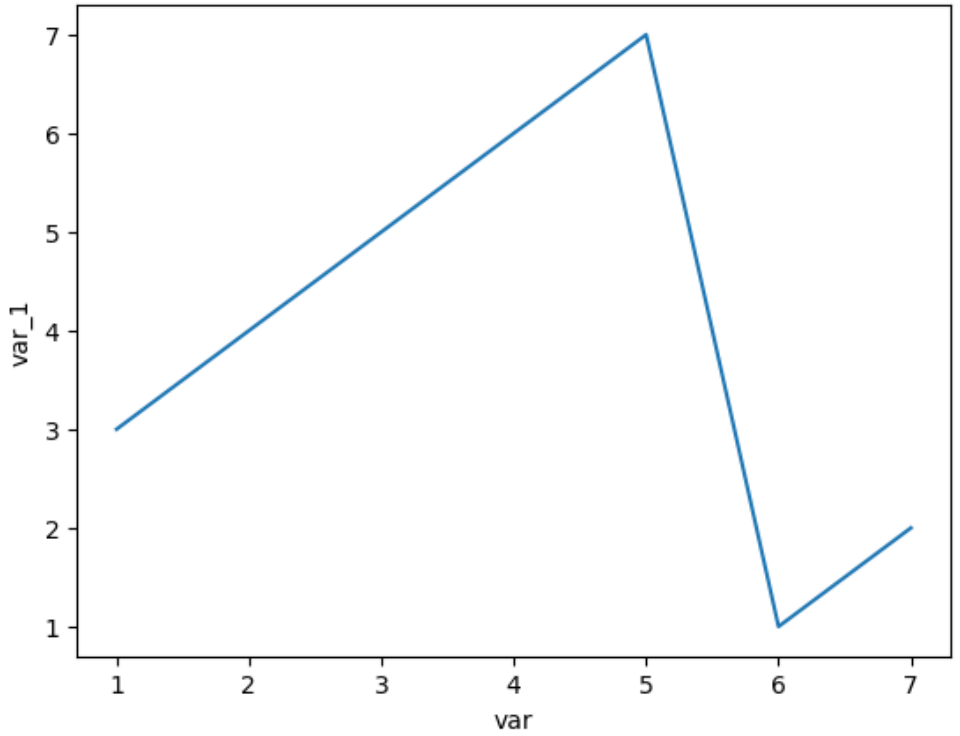
LINE PLOT using Seaborn (using DF)

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

var = [1,2,3,4,5,6,7]

var_1 = [3,4,5,6,7,1,2]

#creating a dataframe

x_1 = pd.DataFrame({"var":var, "var_1":var_1})

sns.lineplot(x='var', y='var_1', data=x_1) #data stores the data that comes from the dataframe

plt.show()
```



LINE PLOT using Seaborn (without DF)

```
#code without using a dataframe

import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = [1,2,3,4,5,6,7]
var_1 = [3,4,5,6,7,1,2]

sns.lineplot(x=var, y=var_1)
plt.show()
```



LINE PLOT using Seaborn (using external dataset)

Go to Google and search for:

Seaborn datasets

Then you will find:

Link: https://github.com/mwaskom/seaborn-data

Select and download: penguins.csv

LINE PLOT using Seaborn (using external dataset)

```
y_1 = sns.load_dataset("penguins")

y_1 #to view the dataset
```

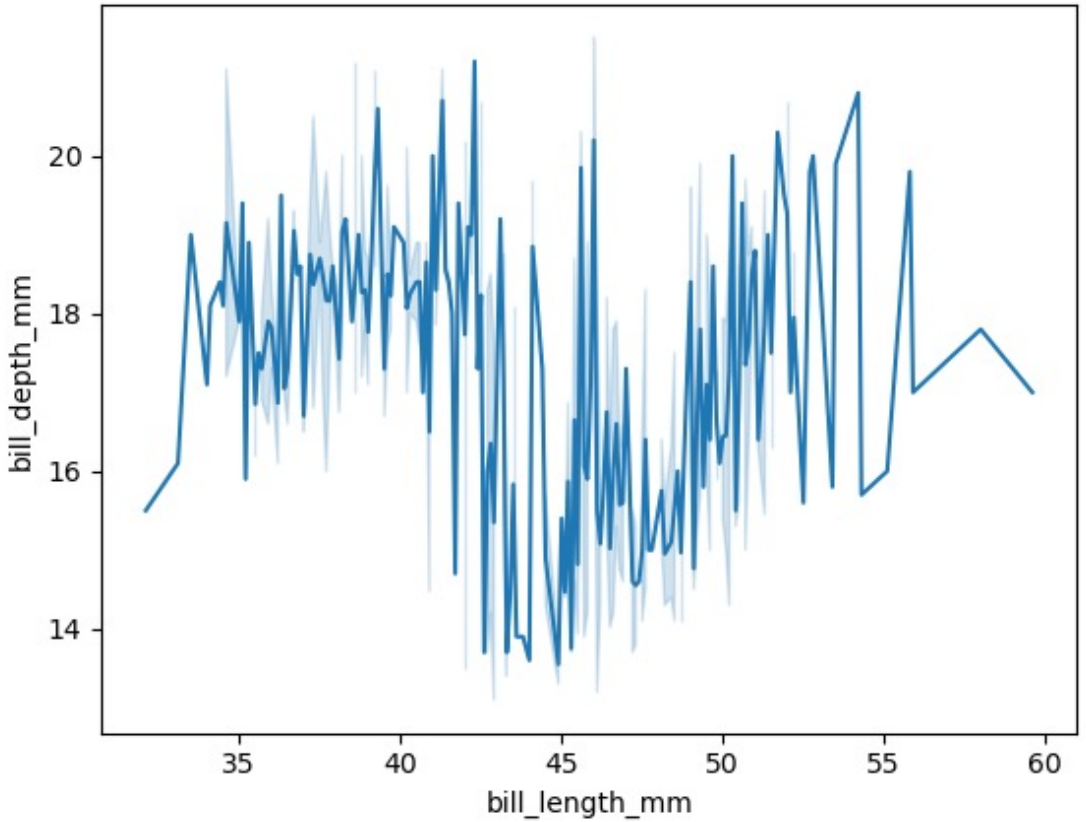
Out[8]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

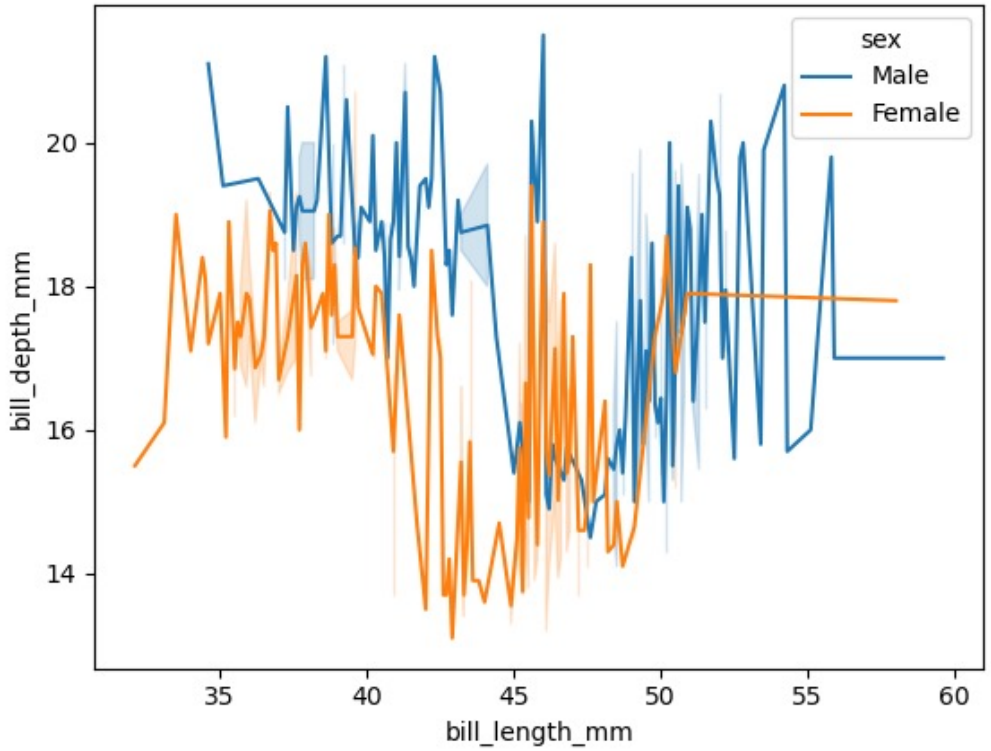
LINE PLOT using Seaborn (using external dataset)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1)

plt.show()
```



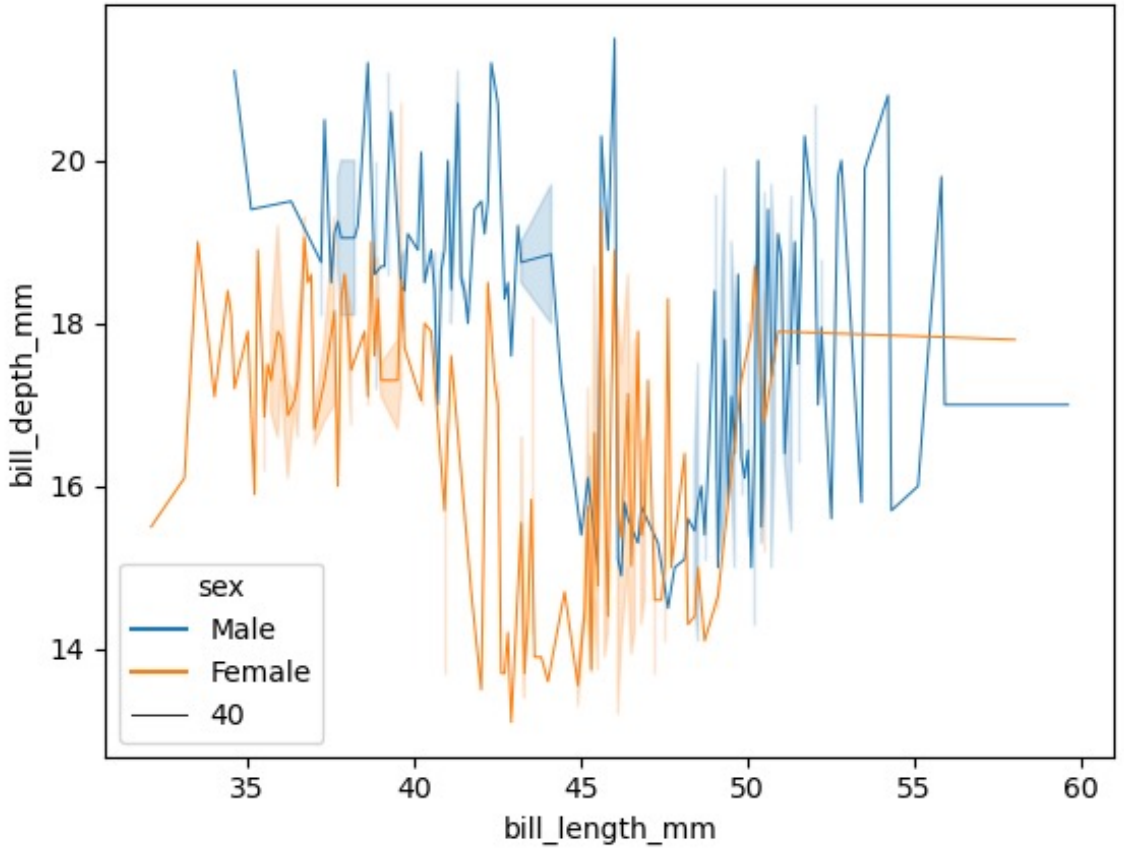
LINE PLOT using Seaborn (hue)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex')

plt.show()
```



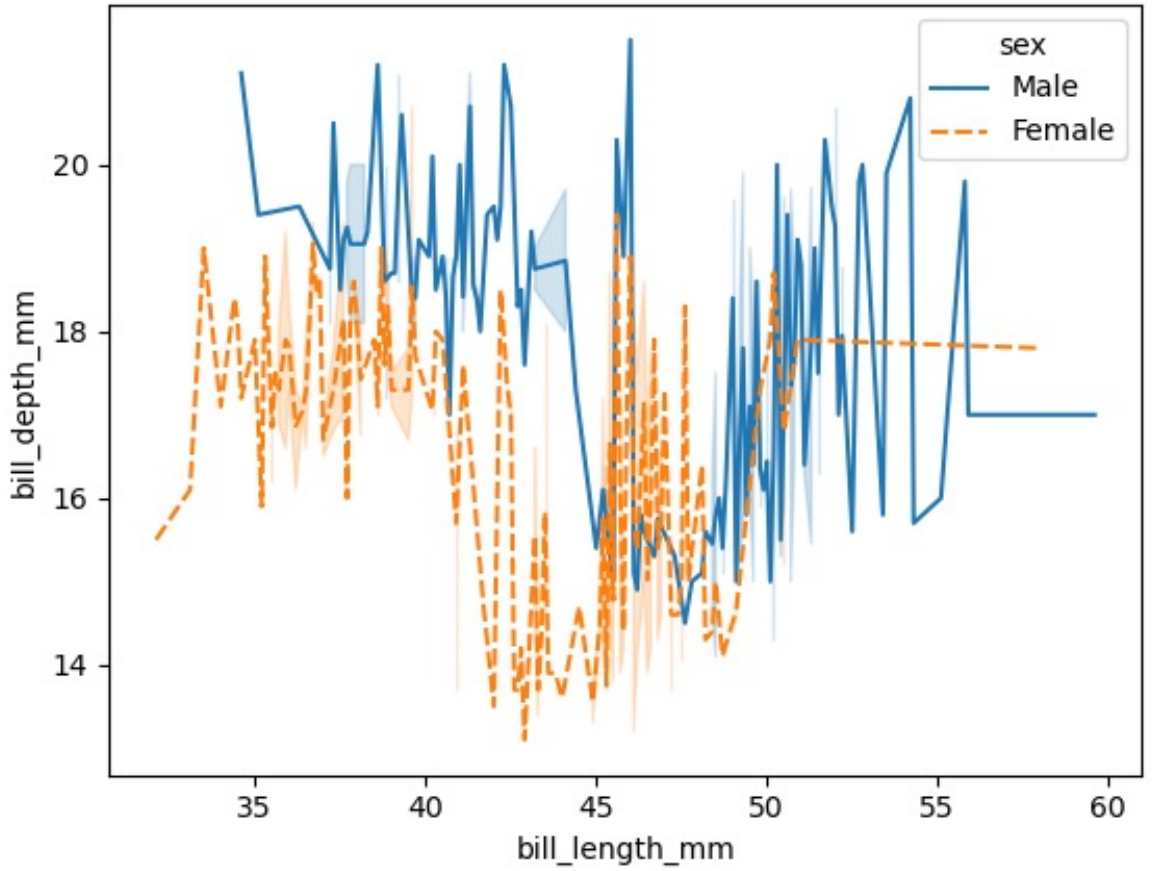
LINE PLOT using Seaborn (size)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
size=40)

plt.show()
```



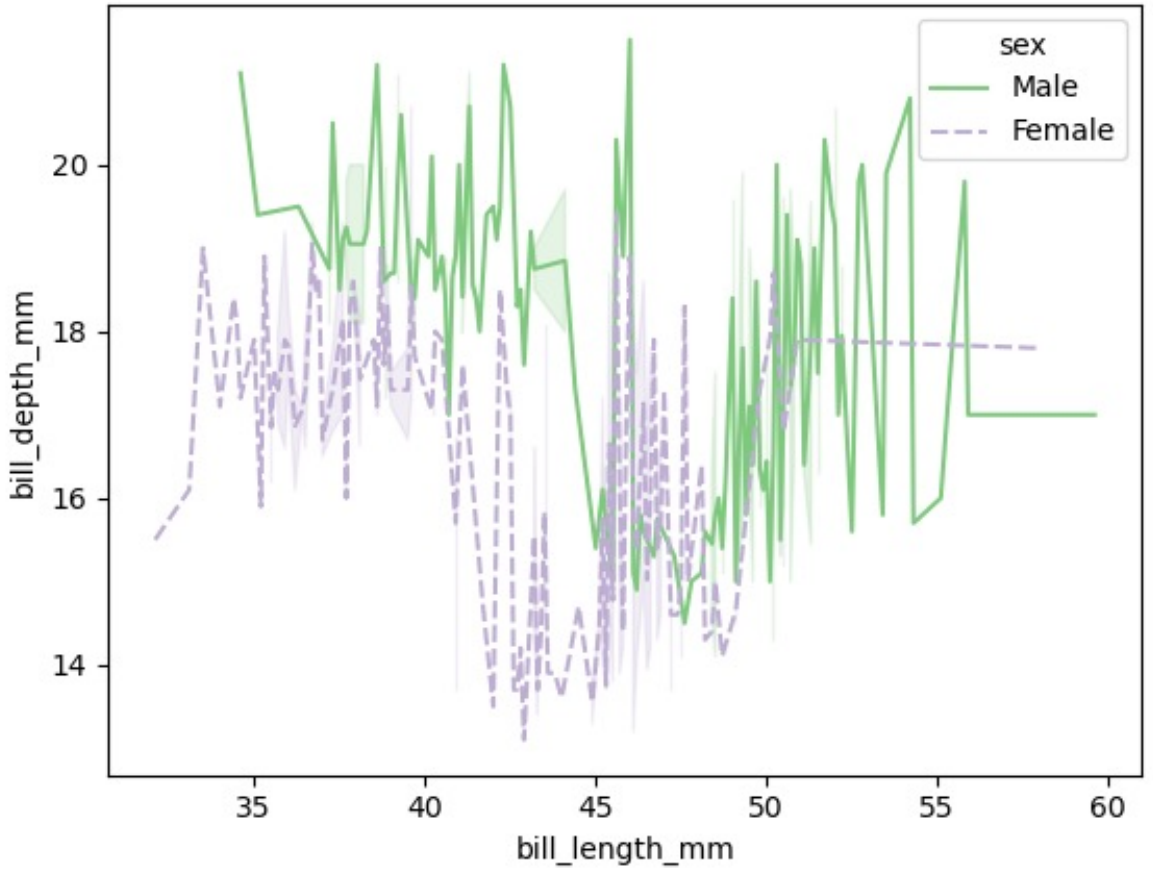
LINE PLOT using Seaborn (style)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex')

plt.show()
```



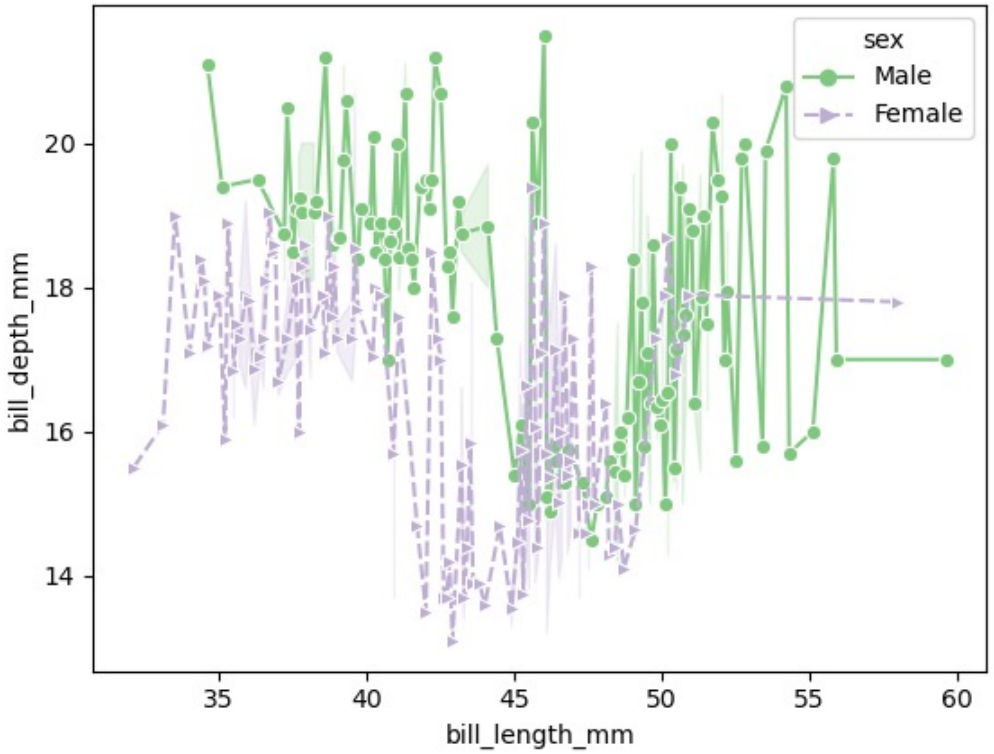
LINE PLOT using Seaborn (palette)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent')

plt.show()
```



LINE PLOT using Seaborn (marker)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent',
markers=["o", ">"])

plt.show()
```

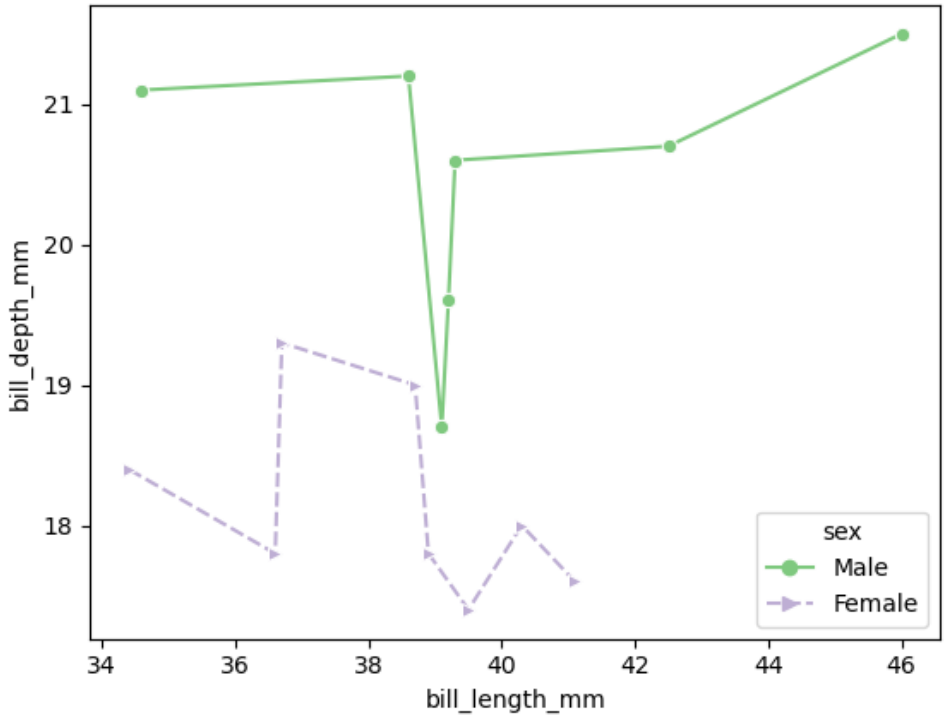


LINE PLOT using Seaborn (head)

y_1 =

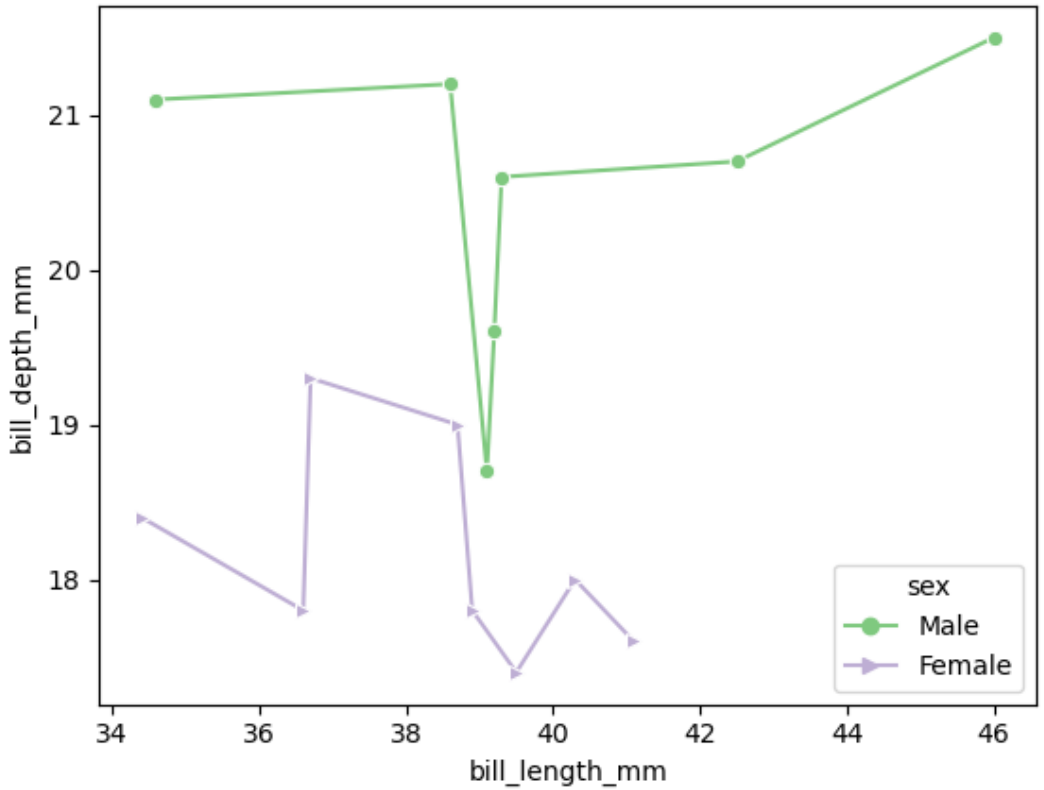
sns.load_dataset("penguins").head(20)

y_1 #to view the dataset



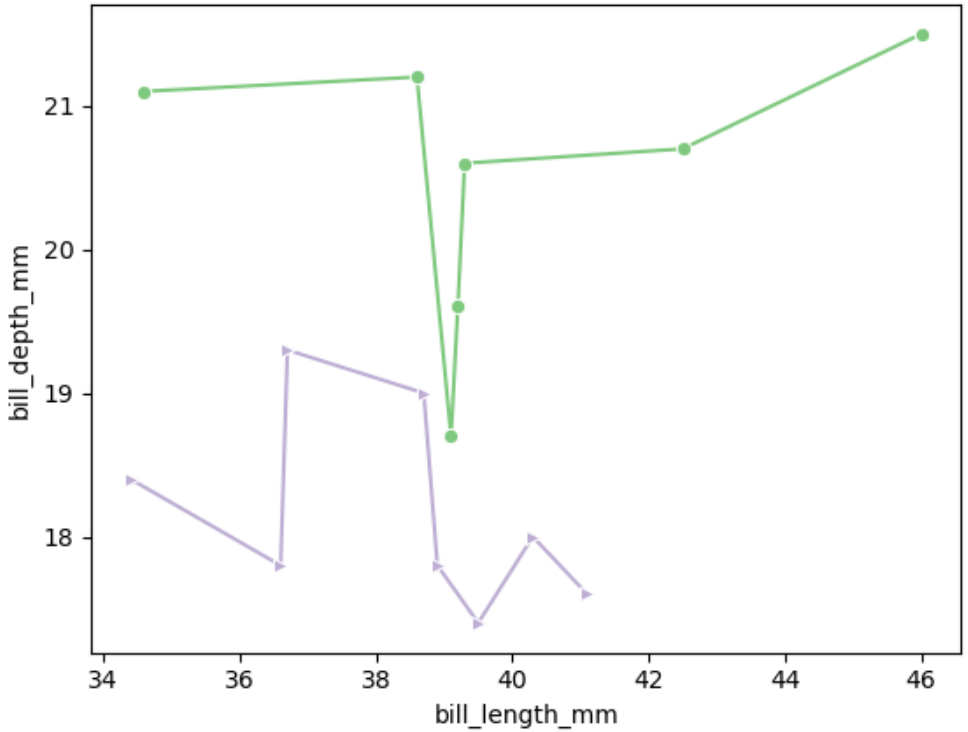
LINE PLOT using Seaborn (dashes)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent',
        markers=["o", ">"], dashes=False)

plt.show()
```



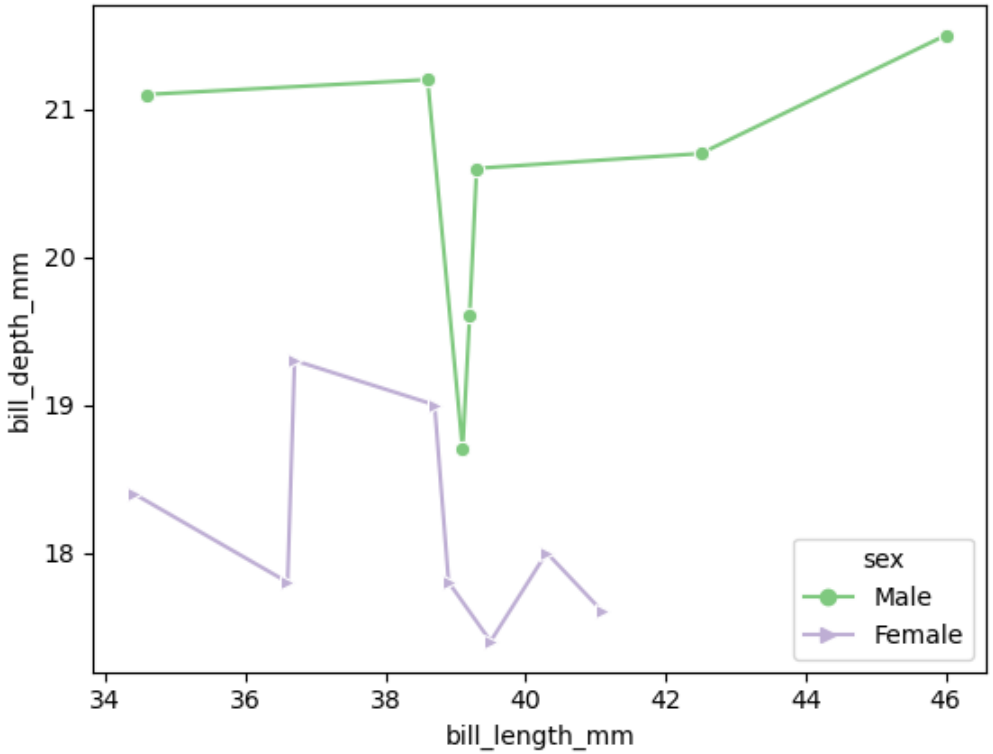
LINE PLOT using Seaborn (removing legend)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent',
        markers=["o", ">"], dashes=False,
legend=False)

plt.show()
```



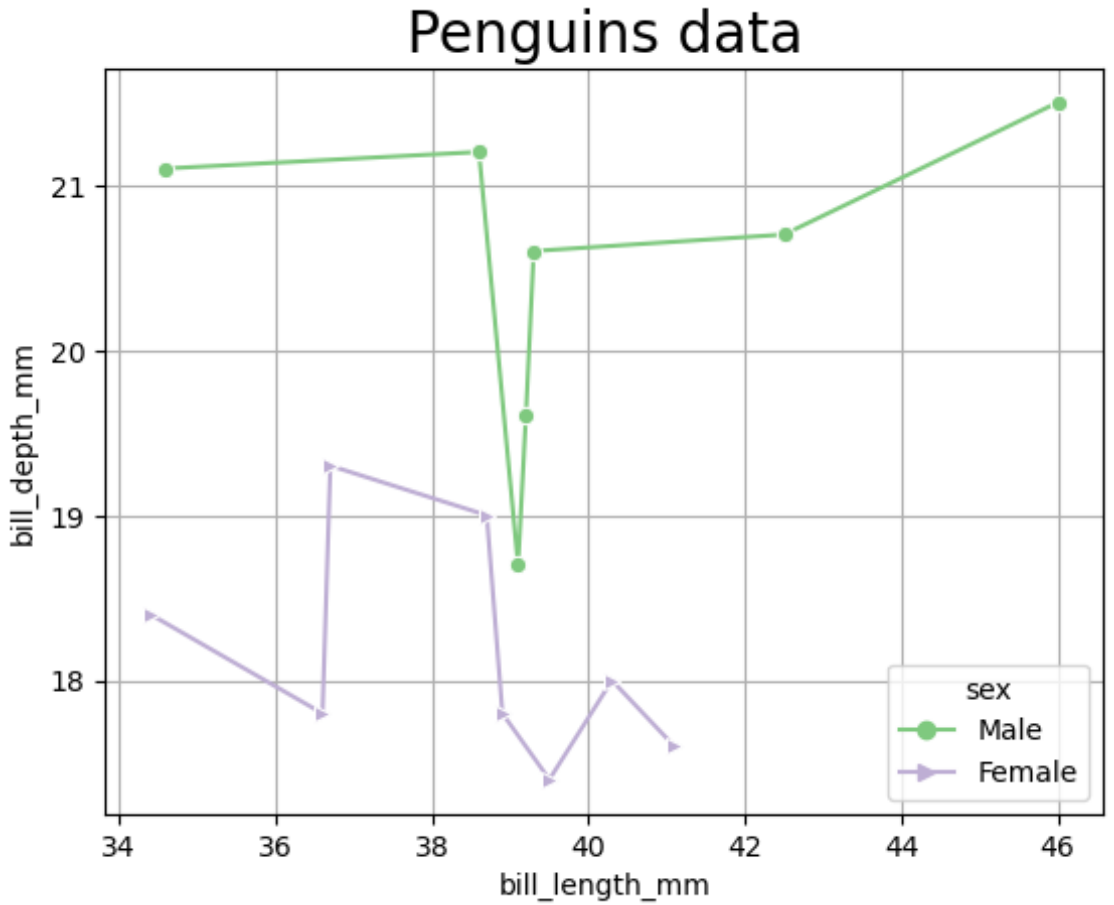
LINE PLOT using Seaborn (removing legend)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent',
        markers=["o", ">"], dashes=False,
legend="full")

plt.show()
```



LINE PLOT using Seaborn (title)

```
sns.lineplot(x='bill_length_mm',
y='bill_depth_mm', data=y_1, hue='sex',
style='sex', palette='Accent',
markers=["o", ">"], dashes=False, legend="full")

plt.grid()
plt.title("Penguins data", fontsize=20)

plt.show()
```



Styling plot in Seaborn

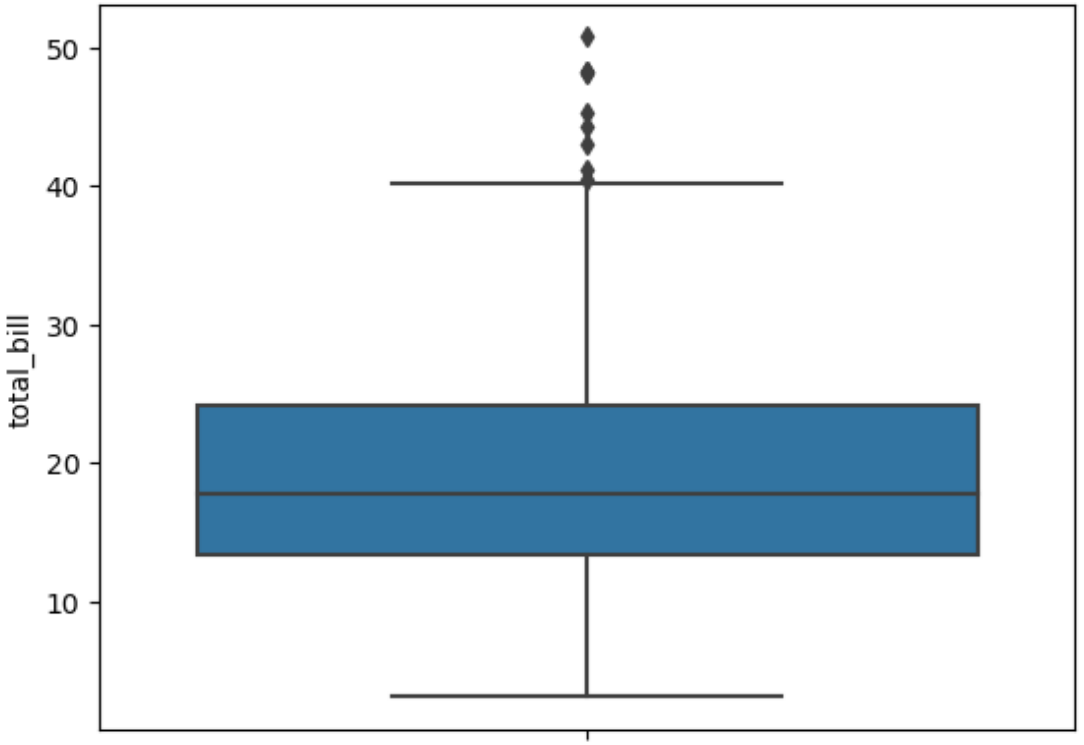
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

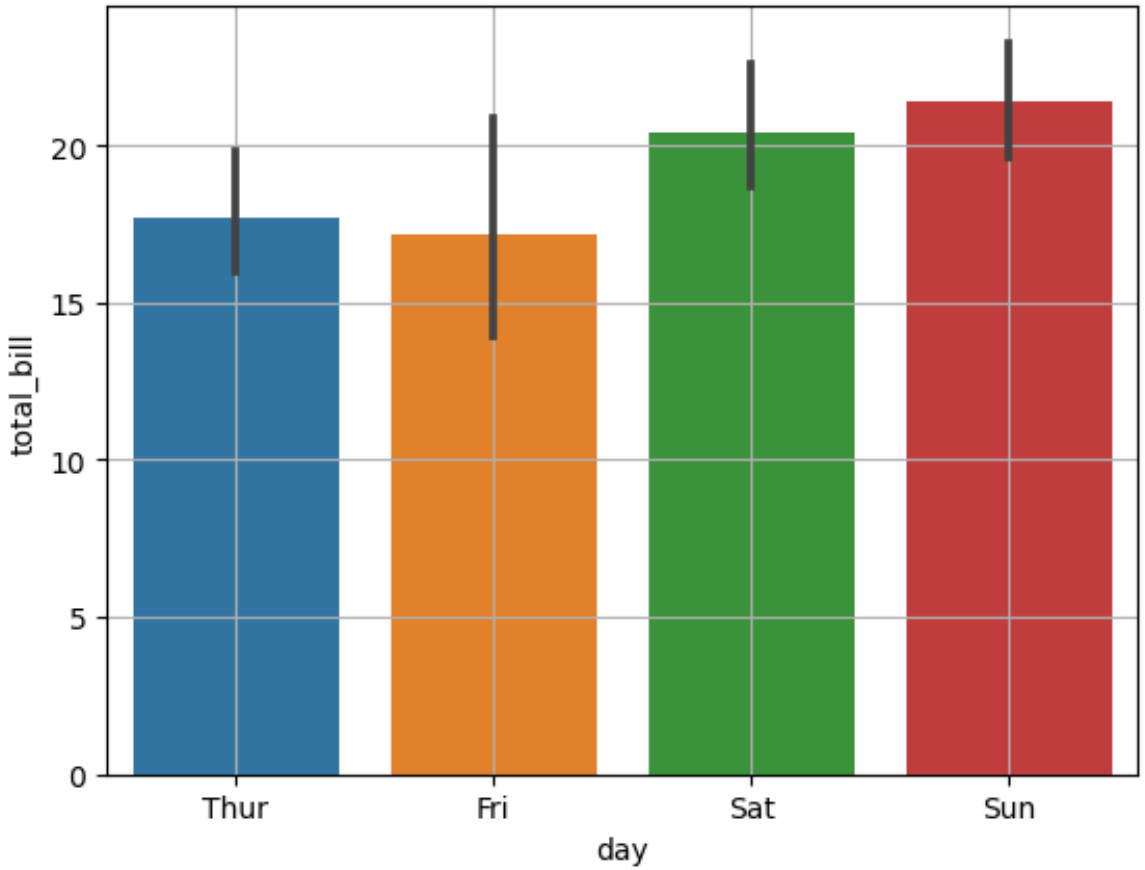
Styling plot in Seaborn (boxplot)

```
sns.boxplot(y=var["total_bill"])

plt.show()
```



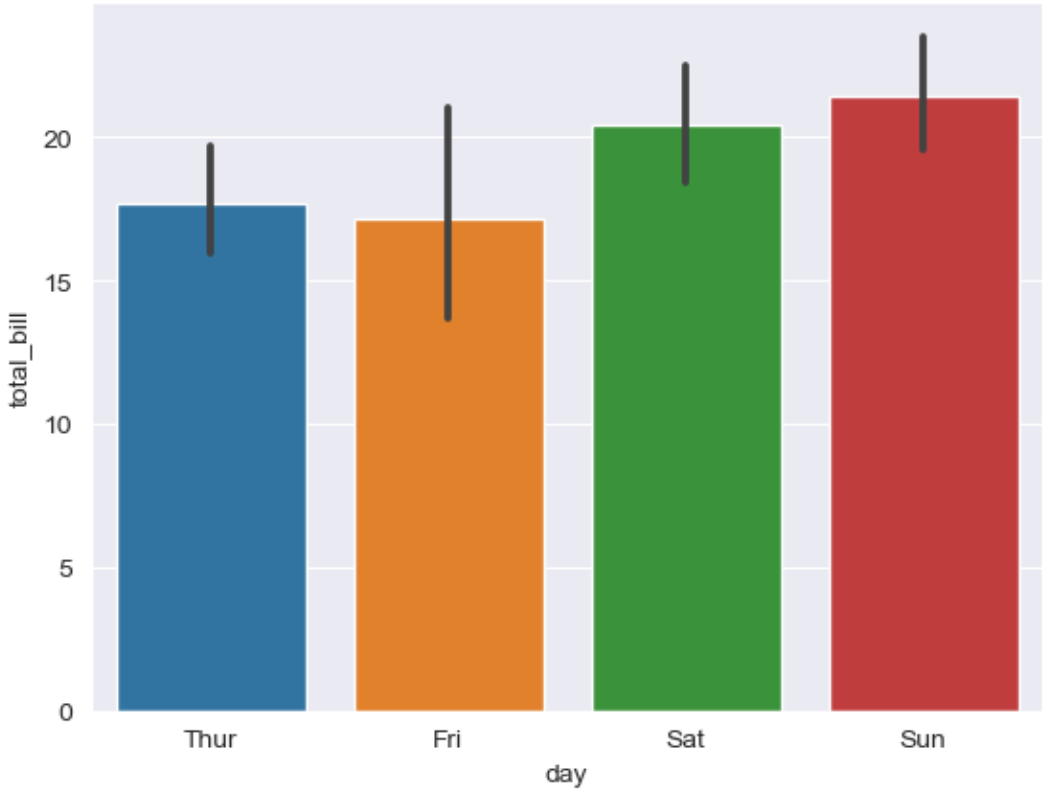
Styling plot in Seaborn (boxplot)

```
sns.barplot(x='day’,
y='total_bill', data=var)

plt.grid()

plt.show()
```



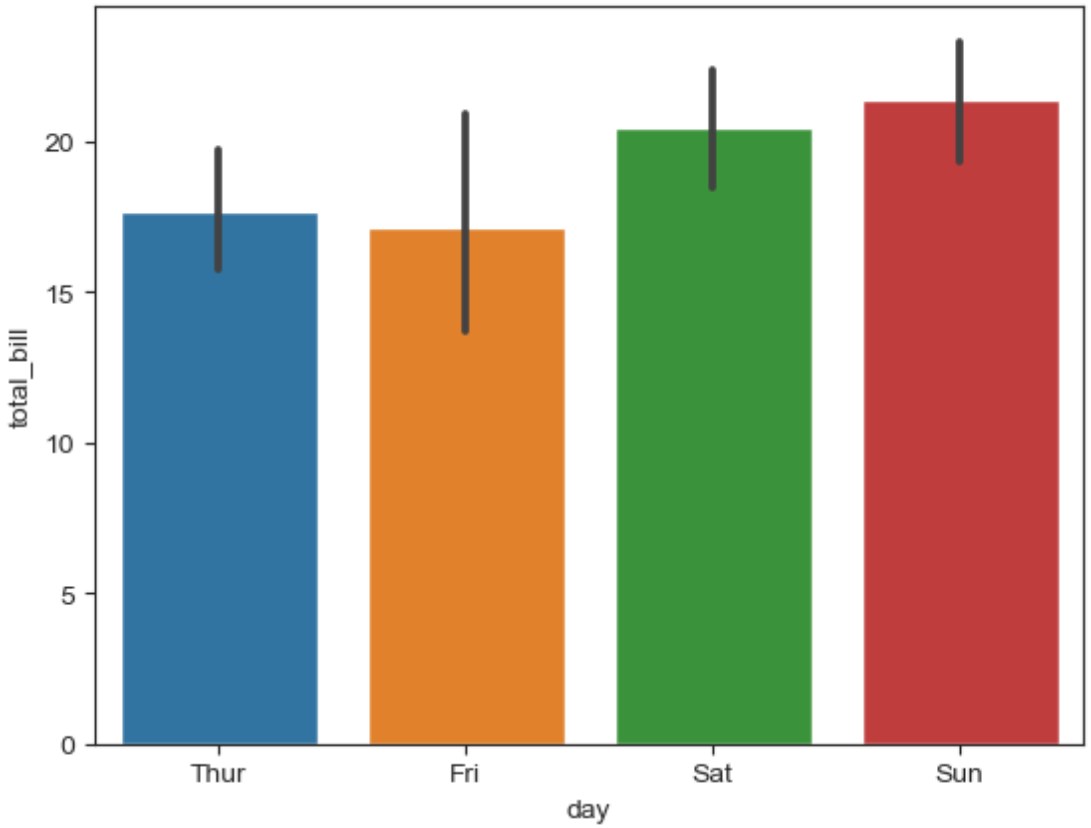
Styling plot in Seaborn (boxplot)

```
sns.set_style("darkgrid")

sns.barplot(x='day', y='total_bill', data=var)

plt.show()
```



Styling plot in Seaborn (ticks)

```
sns.set_style("ticks")

sns.barplot(x='day', y='total_bill', data=var)

plt.show()
```

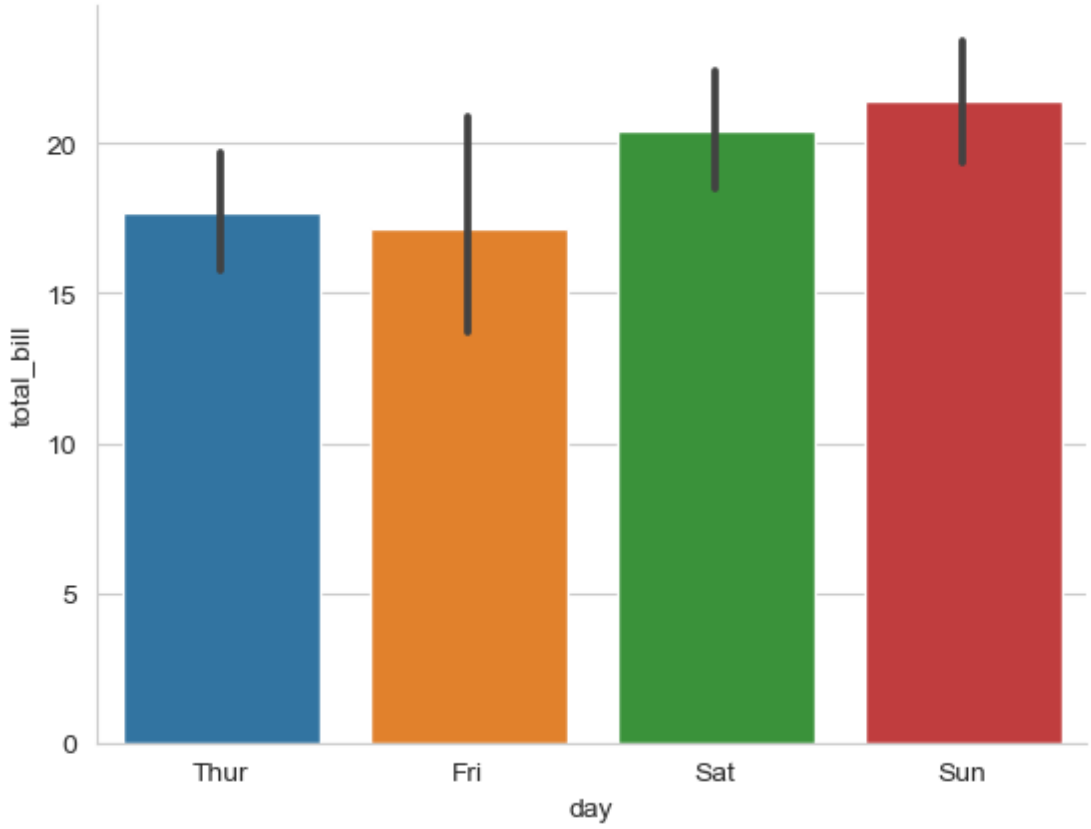


Styling plot in Seaborn

- Seaborn Figure styles
- Removing Axes Spines
- Scale and Context

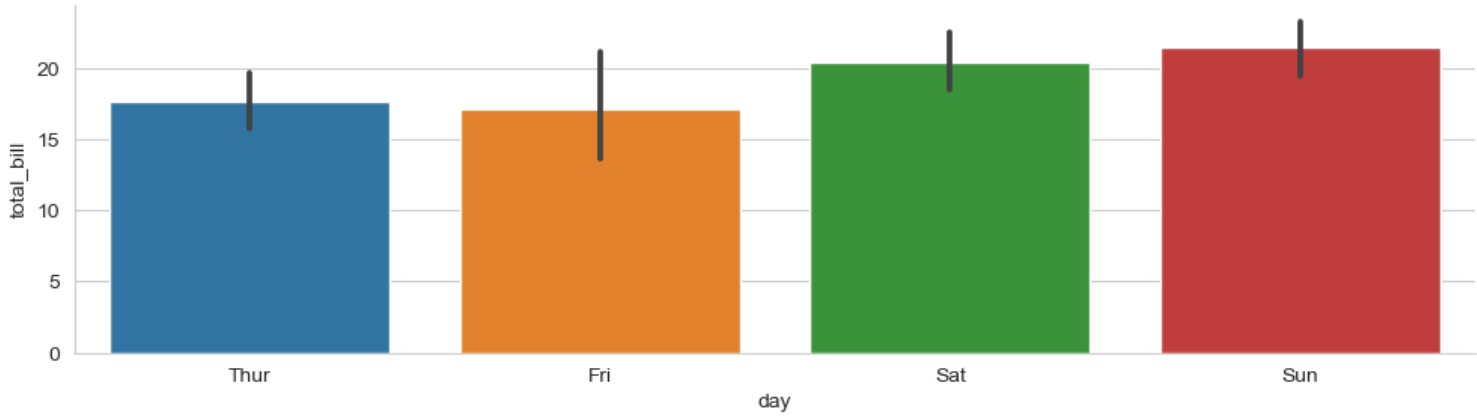
Styling plot in Seaborn

```
sns.set_style("whitegrid")
sns.barplot(x='day', y='total_bill', data=var)
sns.despine()
plt.show()
```



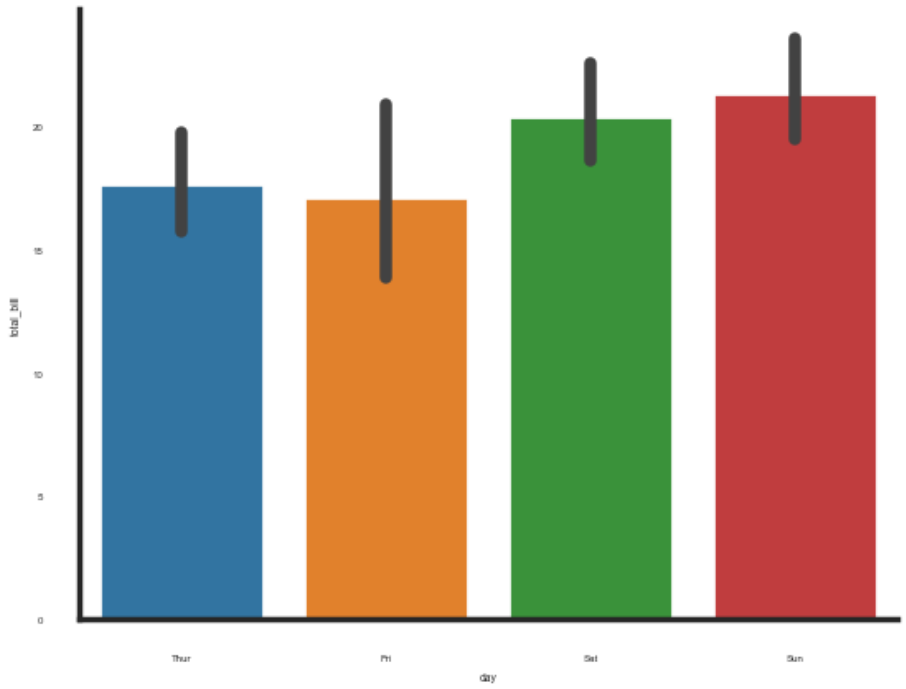
Styling plot in Seaborn

```
sns.set_style("whitegrid")

plt.figure(figsize=(12,3))

sns.barplot(x='day', y='total_bill', data=var)

sns.despine()

plt.show()
```



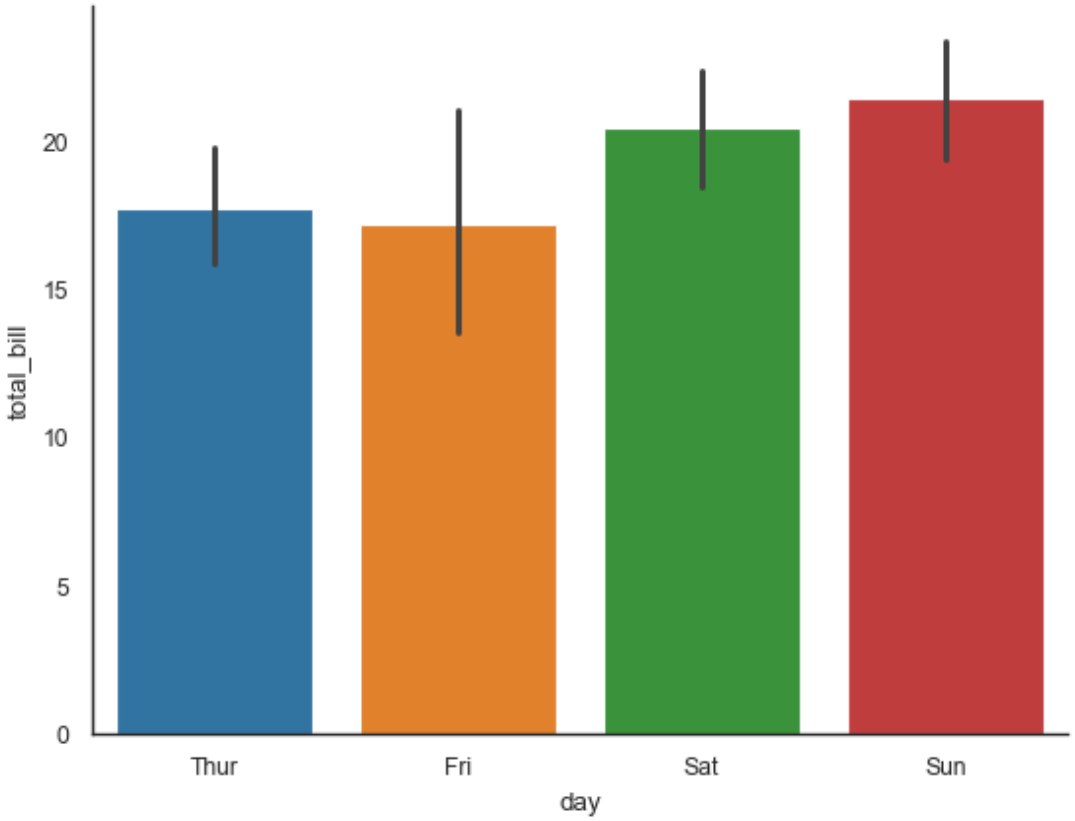
Styling plot in Seaborn (poster)

```
sns.set_style("white")

sns.set_context("poster", font_scale=0.2)

sns.barplot(x='day', y='total_bill', data=var)

sns.despine()

plt.show()
```



Styling plot in Seaborn (paper)

```
sns.set_style("white")

sns.set_context("paper", font_scale=1)

sns.barplot(x='day', y='total_bill', data=var)

sns.despine()

plt.show()
```



Histogram PLOT using Seaborn

```
import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd


var = sns.load_dataset("penguins")


var
```

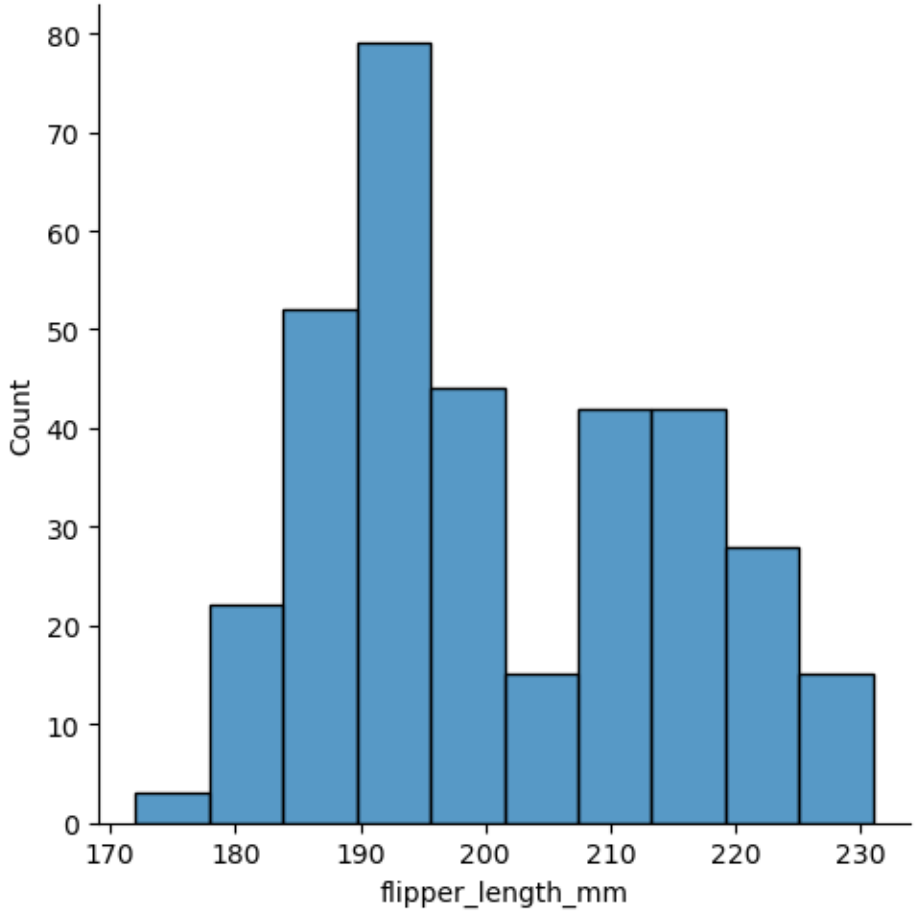
Out[1]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

Histogram PLOT using Seaborn

```
sns.displot(var["flipper_length_mm"])

plt.show()
```



Histogram PLOT using Seaborn (bins)

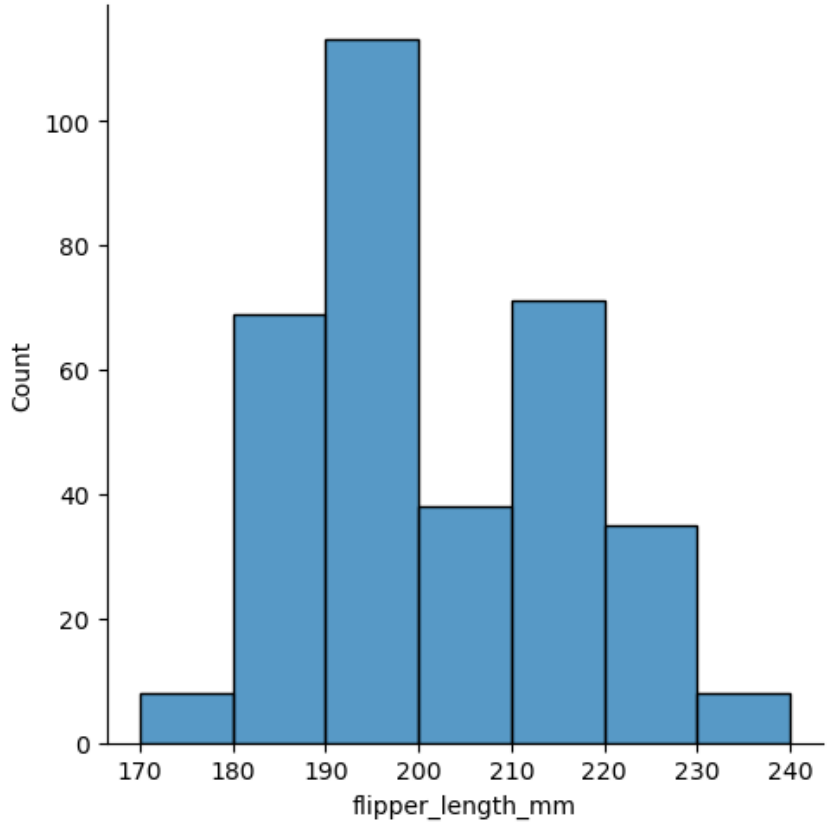
#to organize the X-axis into proper bins

(fixing range)

sns.displot(var["flipper_length_mm"],

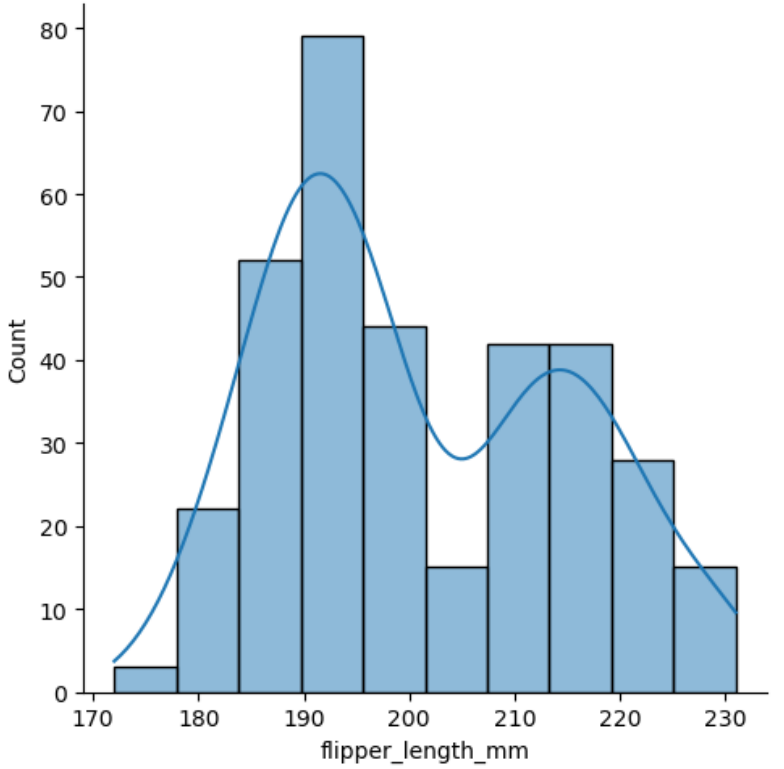
bins=[170,180,190,200,210,220,230,240])

plt.show()



Histogram PLOT using Seaborn (kde)

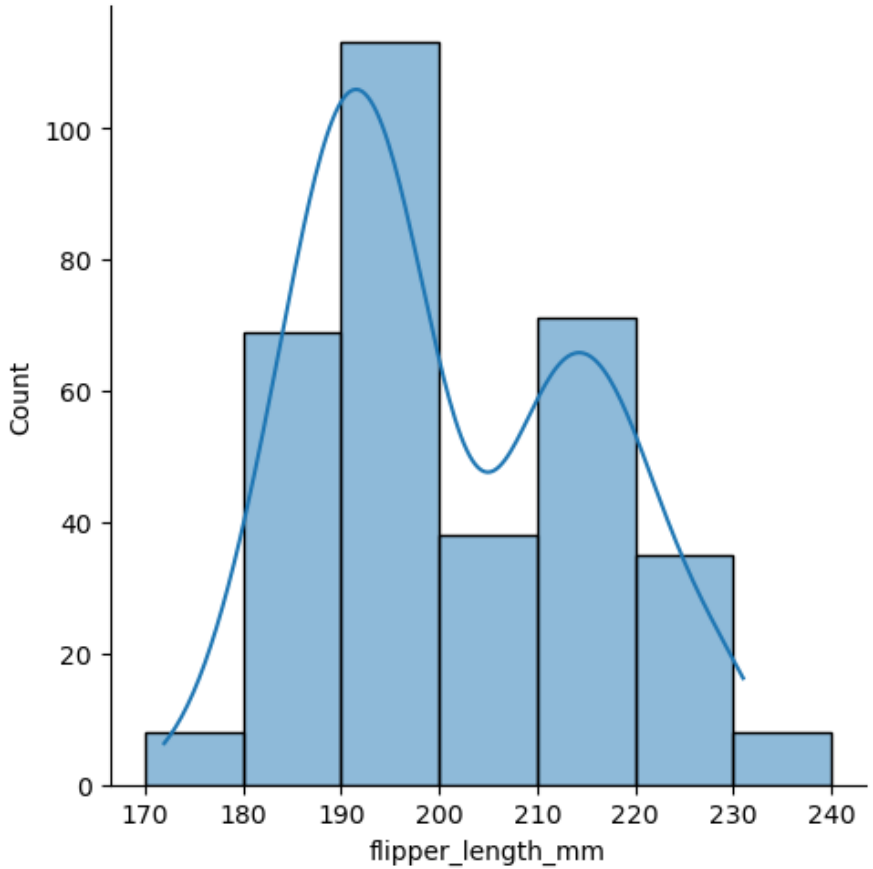
```
sns.displot(var["flipper_length_mm"],
kde=True)

plt.show()
```



A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.
Link: https://seaborn.pydata.org/generated/seaborn.kdeplot.html

Histogram PLOT using Seaborn (kde)

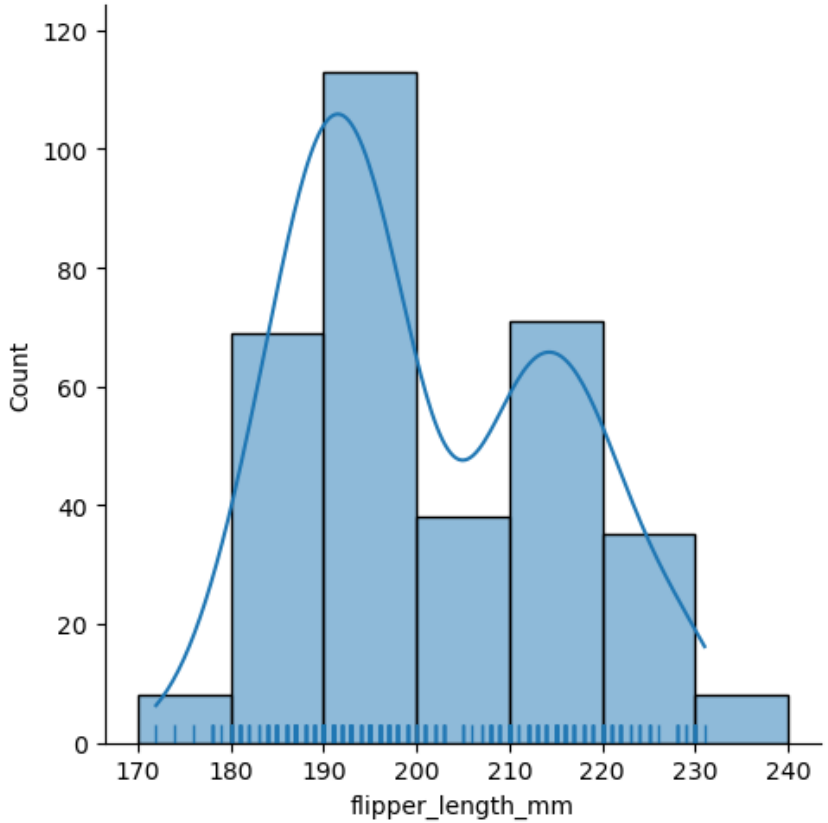
```
sns.displot(var["flipper_length_mm"],
bins=[170,180,190,200,210,220,230,240],
kde=True)

plt.show()
```



A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.
Link: https://seaborn.pydata.org/generated/seaborn.kdeplot.html

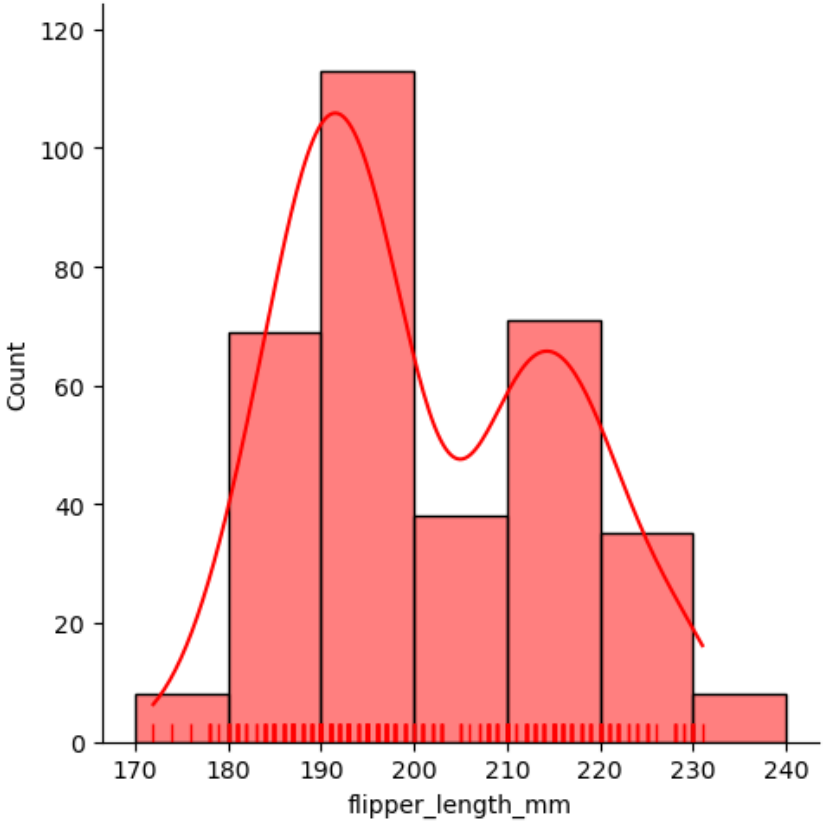
Histogram PLOT using Seaborn (rug)

```
sns.displot(var["flipper_length_mm"],
bins=[170,180,190,200,210,220,230,240],
kde=True, rug=True)

plt.show()
```



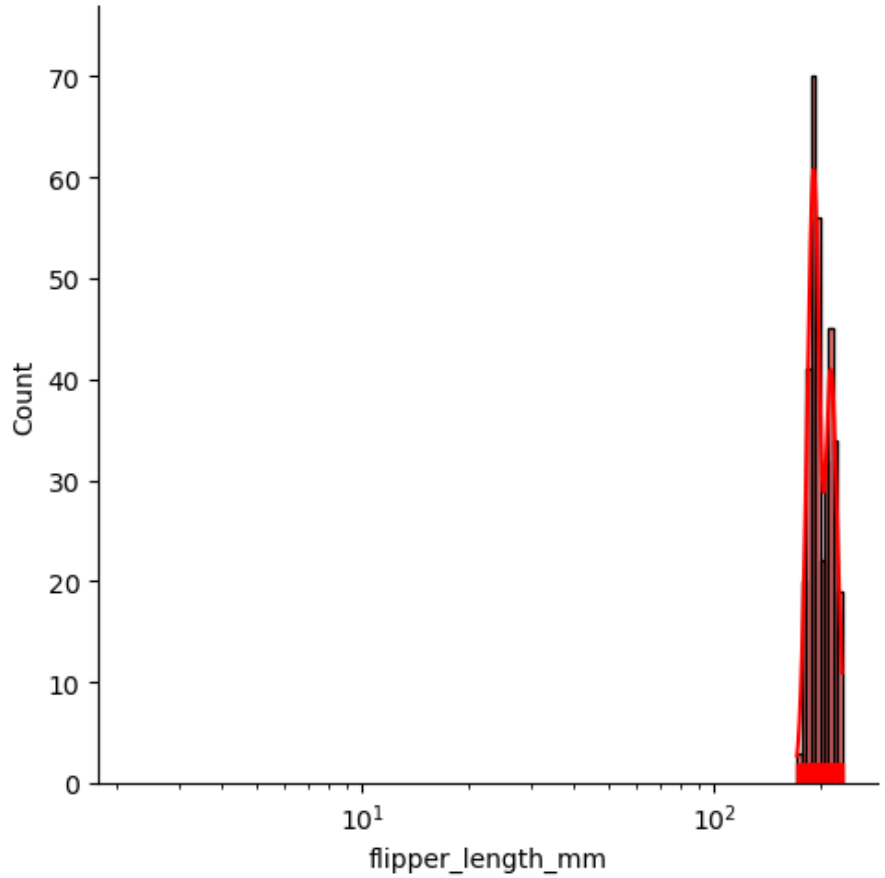
Histogram PLOT using Seaborn (color)

```
sns.displot(var["flipper_length_mm"],
bins=[170,180,190,200,210,220,230,240],
kde=True, rug=True, color='r')

plt.show()
```



Histogram PLOT using Seaborn (log_scale)

```
sns.displot(var["flipper_length_mm"],
kde=True, rug=True, color='r',
log_scale=True)

plt.show()
```



BAR PLOT using Seaborn

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

var = sns.load_dataset("penguins")
var
```

Out[1]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

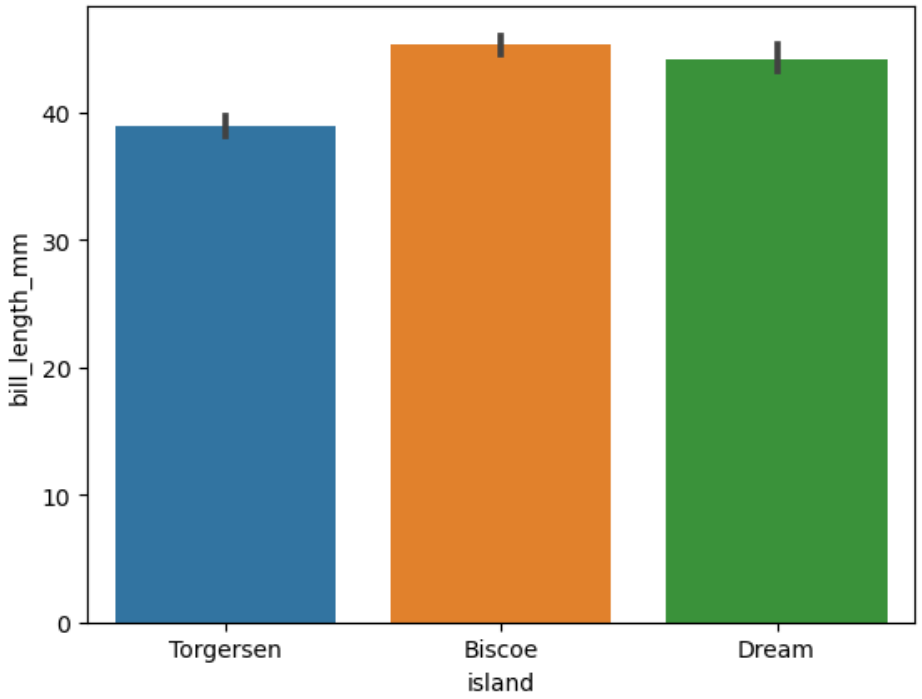
BAR PLOT using Seaborn

Create a heading: Bar plot

#option 1 of writing

```
sns.barplot(x=var.island,
y=var.bill_length_mm)


plt.show()
```

Output graph



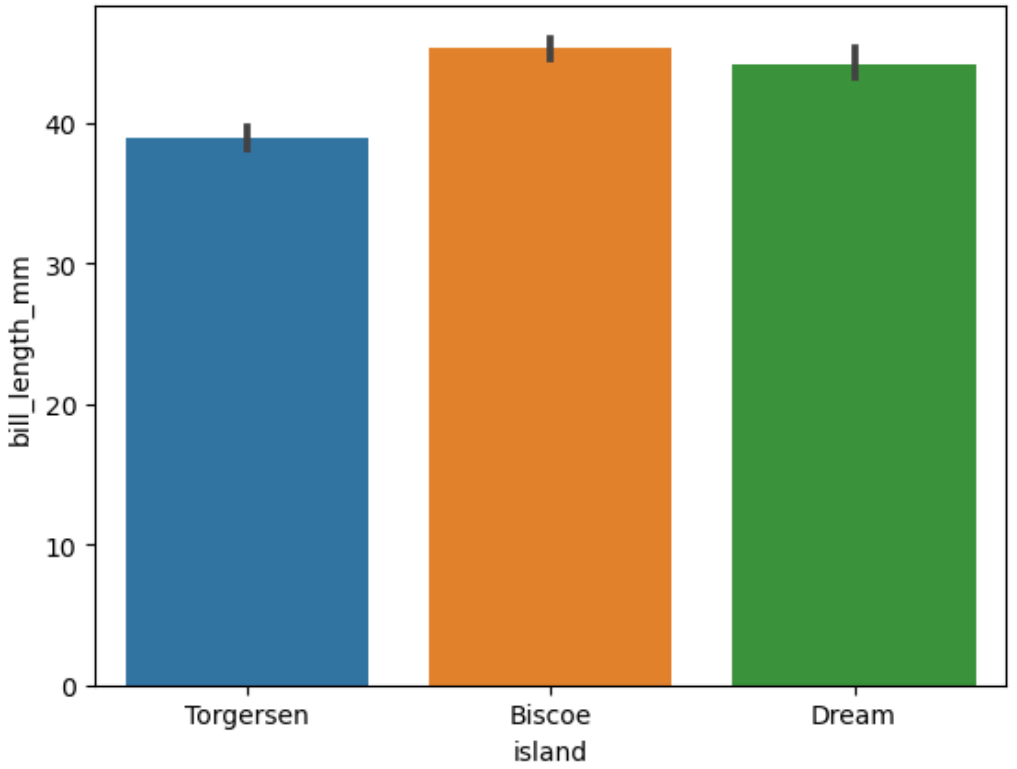
BAR PLOT using Seaborn

Create a heading: Bar plot

#option 2 of writing

```
sns.barplot(x='island', y='bill_length_mm', data=var)

plt.show()
```

Output graph



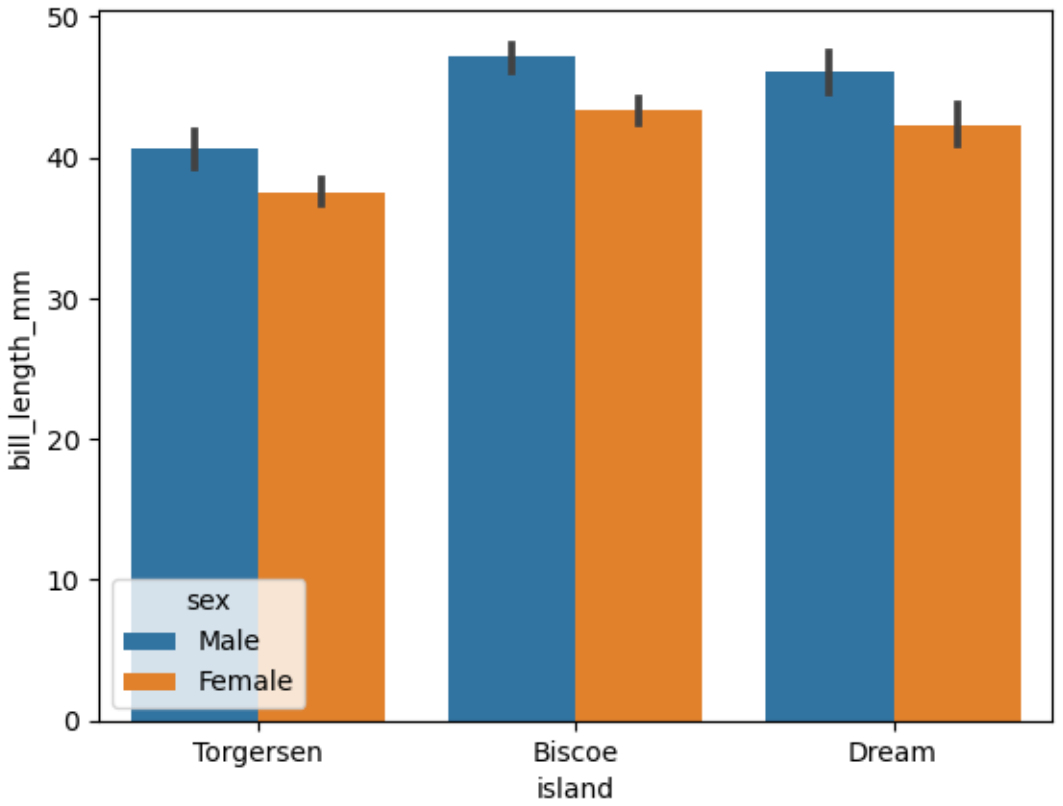
BAR PLOT using Seaborn (hue)

Create a heading: Bar plot

#option 2 of writing

```
sns.barplot(x='island', y='bill_length_mm', data=var, hue='sex')

plt.show()
```

Output graph



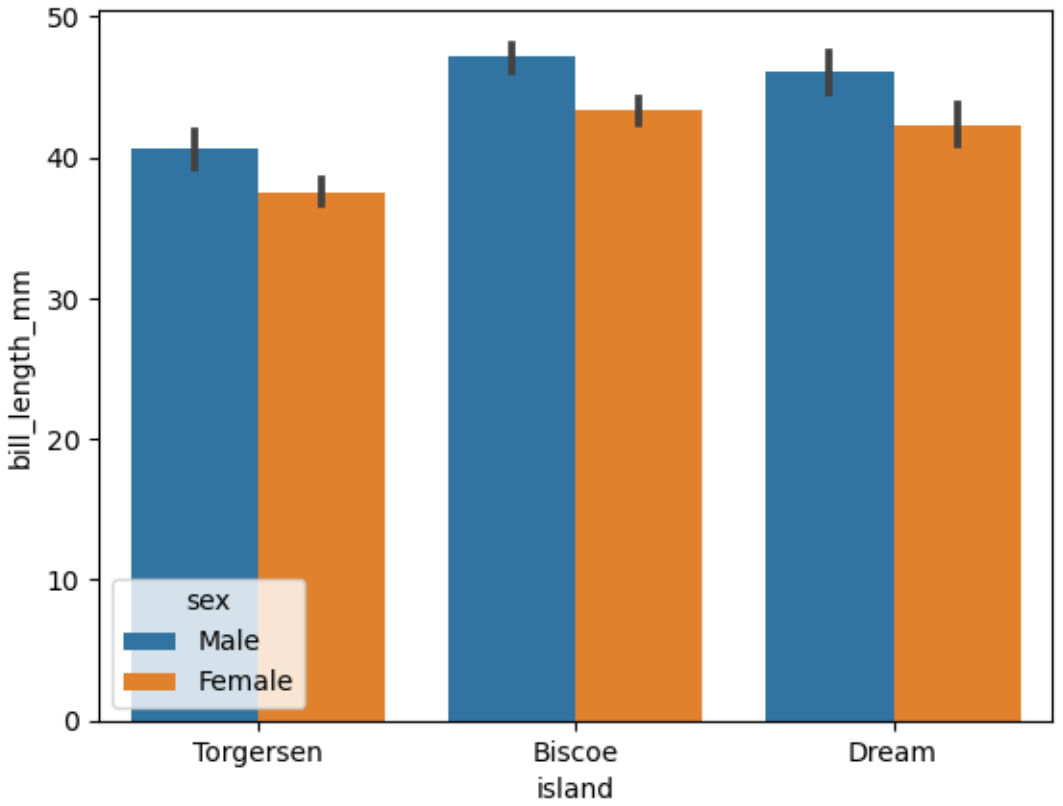
BAR PLOT using Seaborn (hue)

Create a heading: Bar plot

#option 2 of writing

```
sns.barplot(x='island', y='bill_length_mm', data=var, hue='sex')

plt.show()
```

Output graph

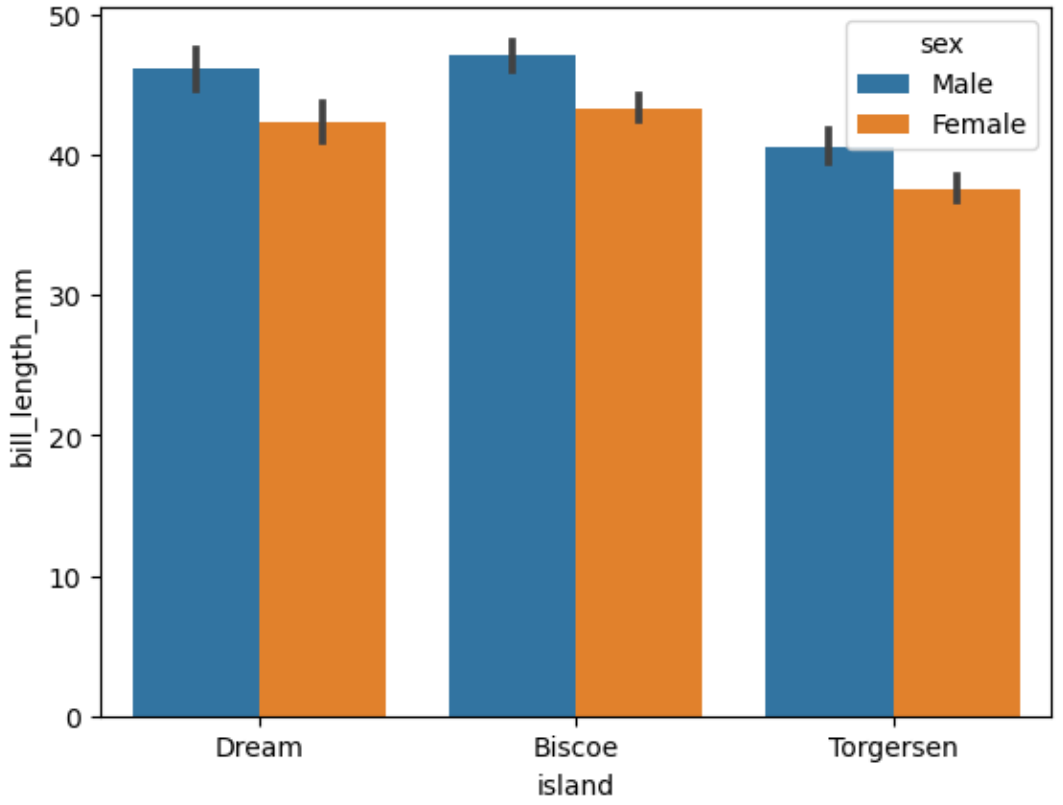


BAR PLOT using Seaborn (order)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1)

plt.show()
```

Output graph

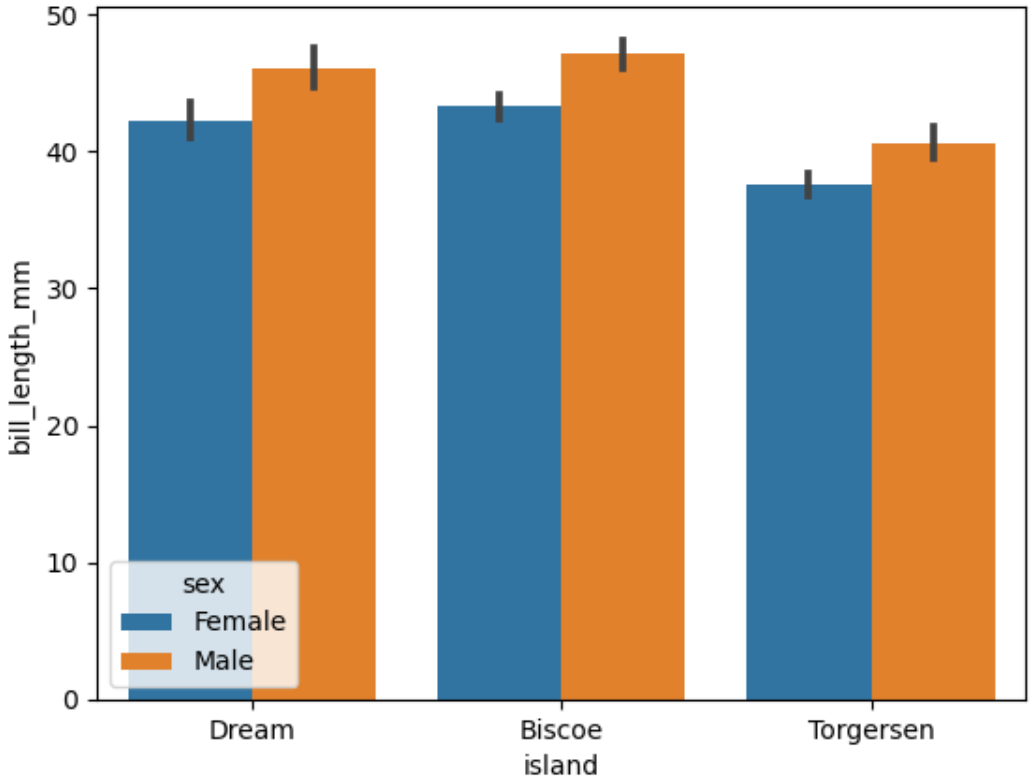


BAR PLOT using Seaborn (order)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"])

plt.show()
```

Output graph

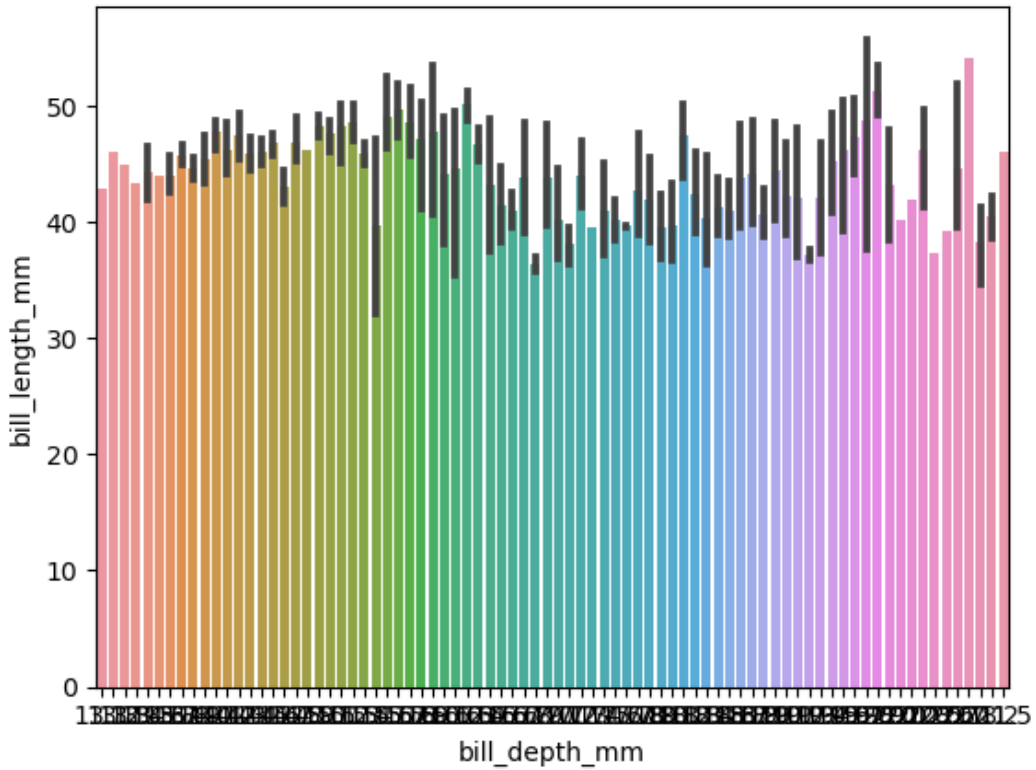


BAR PLOT using Seaborn (orientation)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='bill_depth_mm', y='bill_length_mm',
data=var, orient='v')

plt.show()
```

Output graph

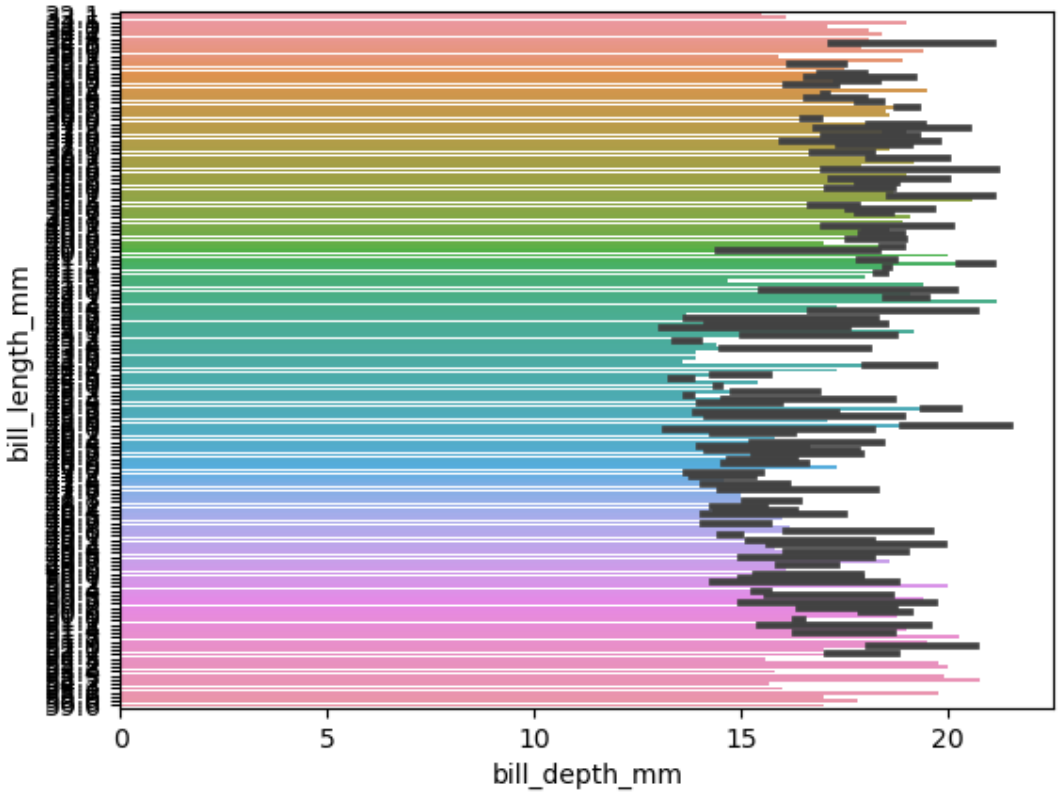


BAR PLOT using Seaborn (orientation)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='bill_depth_mm', y='bill_length_mm',
data=var, orient='h')

plt.show()
```

Output graph

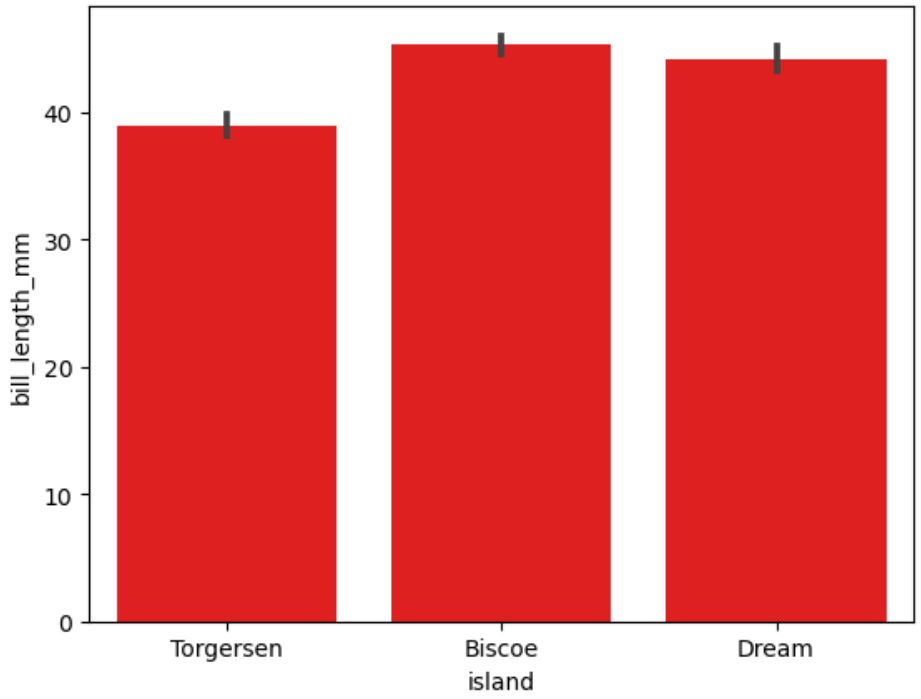


BAR PLOT using Seaborn (color)

Output graph

Create a heading: Bar plot

```
sns.barplot(x='island', y='bill_length_mm', data=var,
color='r')

plt.show()
```

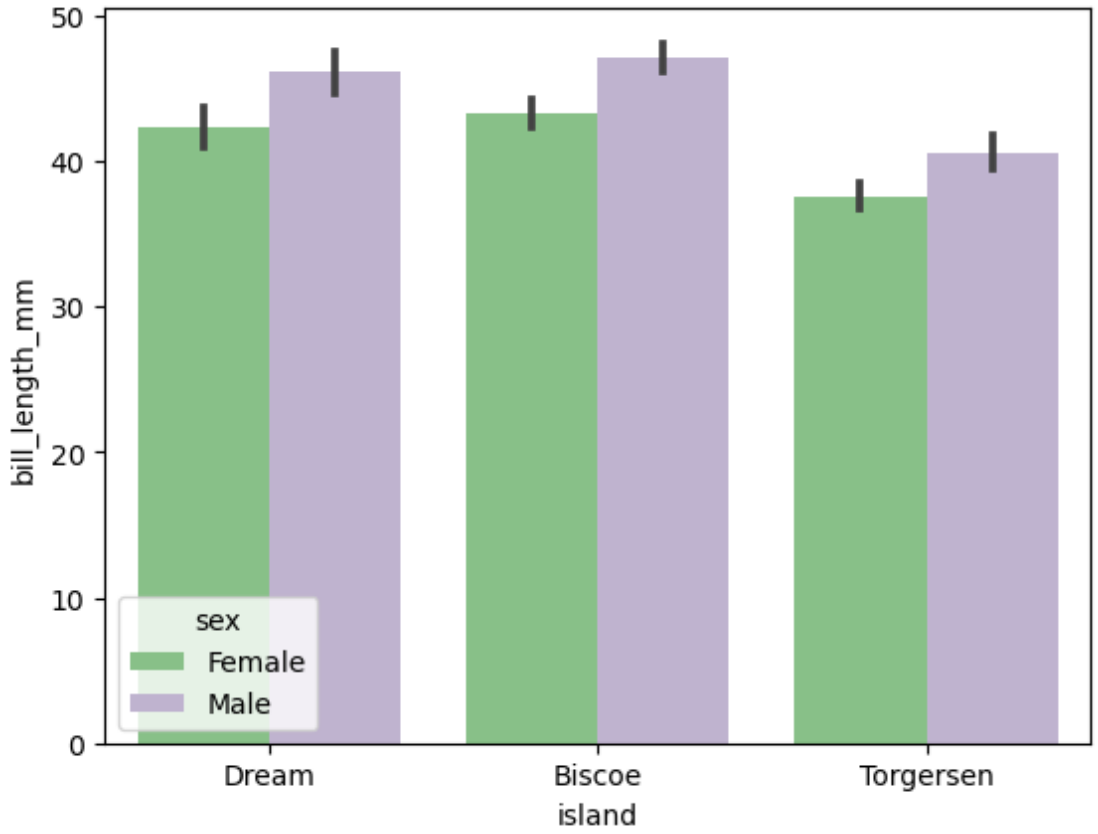


BAR PLOT using Seaborn (palette)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], palette='Accent')

plt.show()
```

Output graph

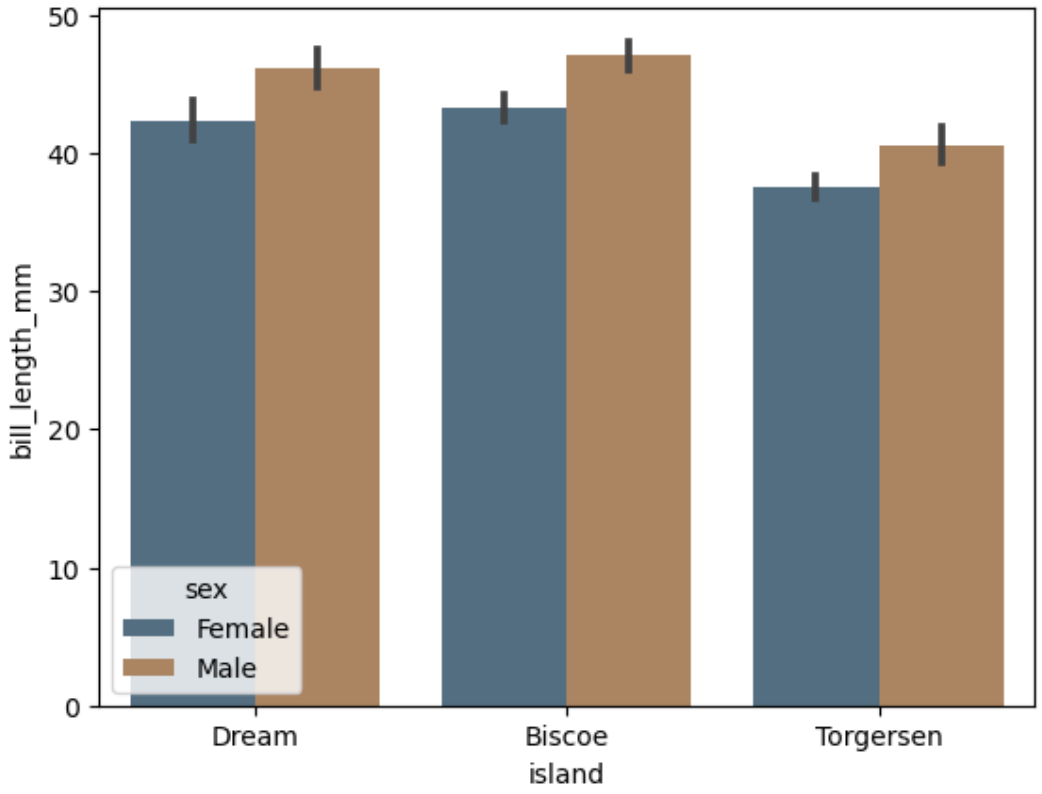


BAR PLOT using Seaborn (saturation)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], saturation=0.3)

plt.show()
```

Output graph

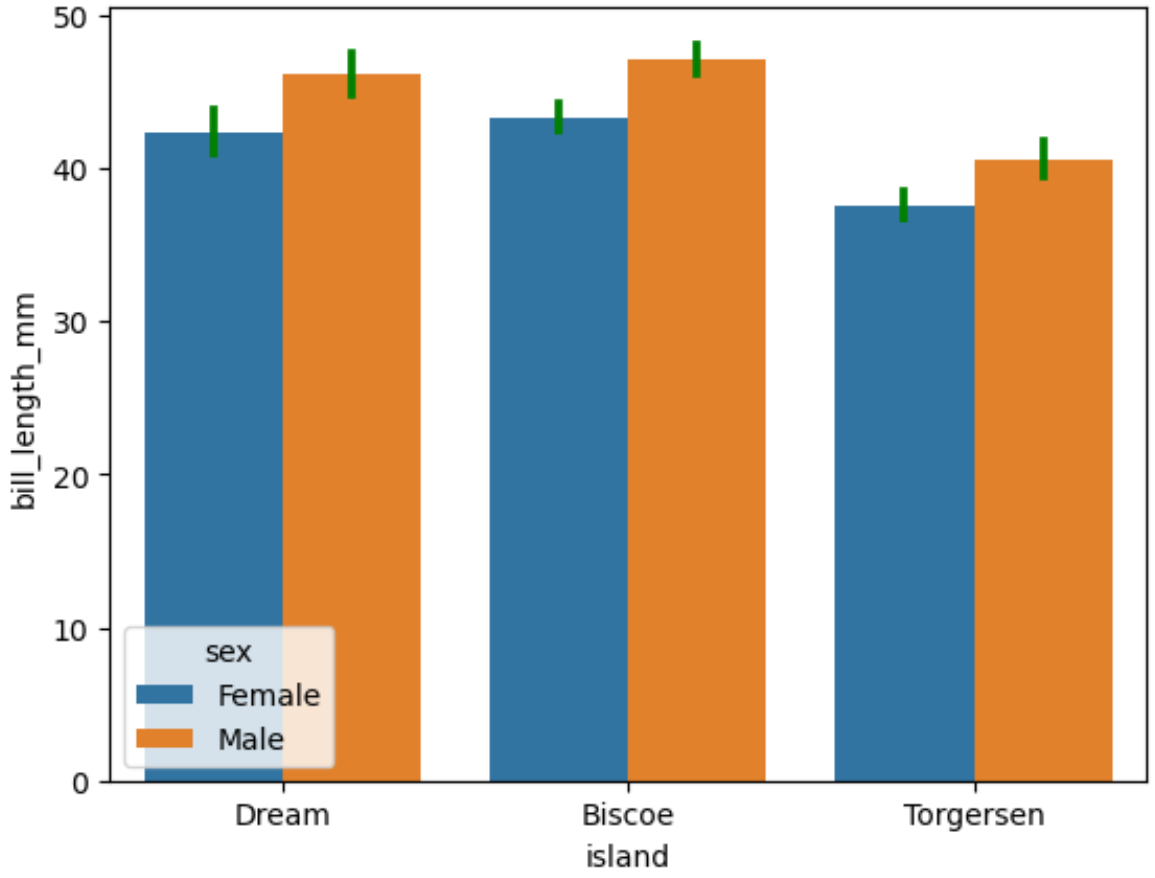


BAR PLOT using Seaborn (errcolor)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], errcolor='g')

plt.show()
```

Output graph

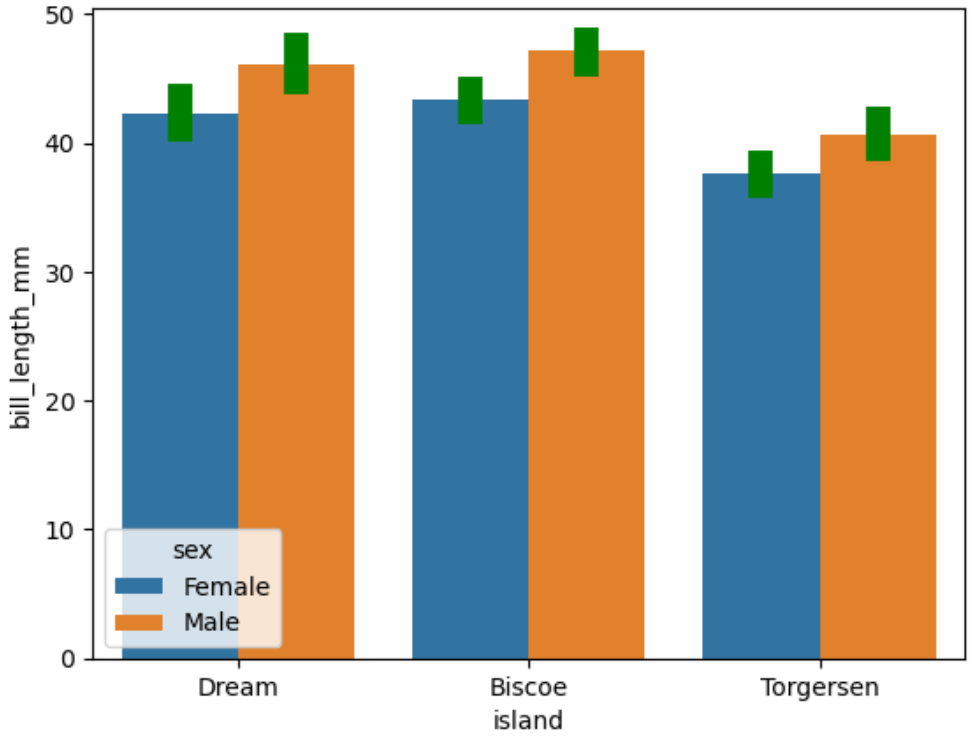


BAR PLOT using Seaborn (errwidth)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], errcolor='g', errwidth=10)

plt.show()
```

Output graph

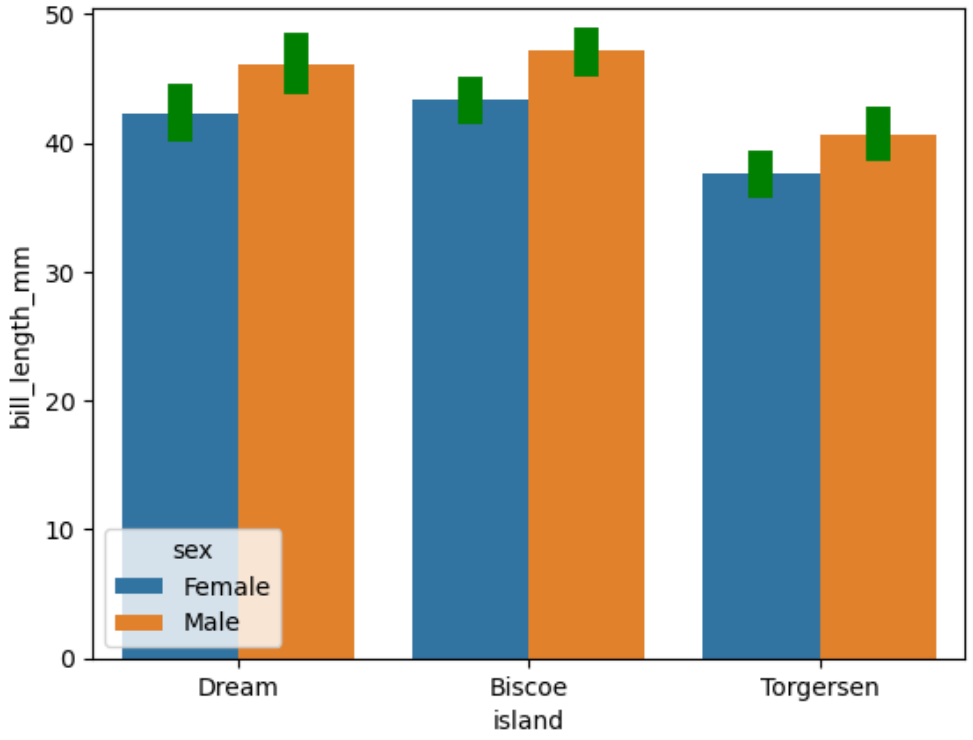


BAR PLOT using Seaborn (errwidth)

Create a heading: Bar plot

```
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], errcolor='g', errwidth=10)

plt.show()
```

Output graph

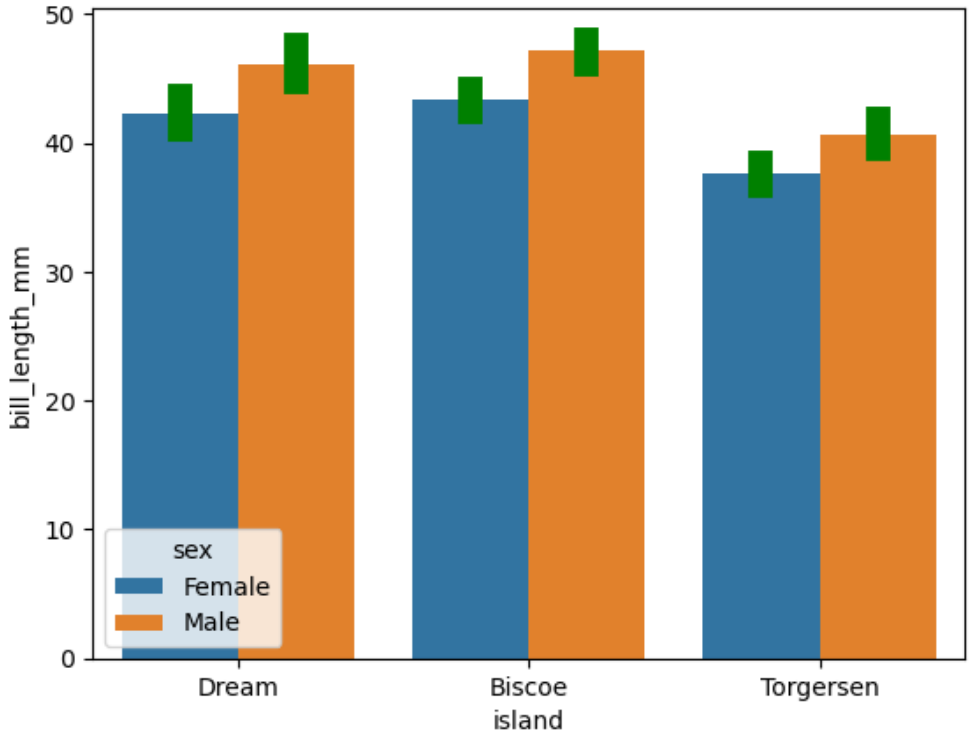


BAR PLOT using Seaborn (darkgrid)

Create a heading: Bar plot

```
sns.set(style='darkgrid')
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], errcolor='g')

plt.show()
```

Output graph

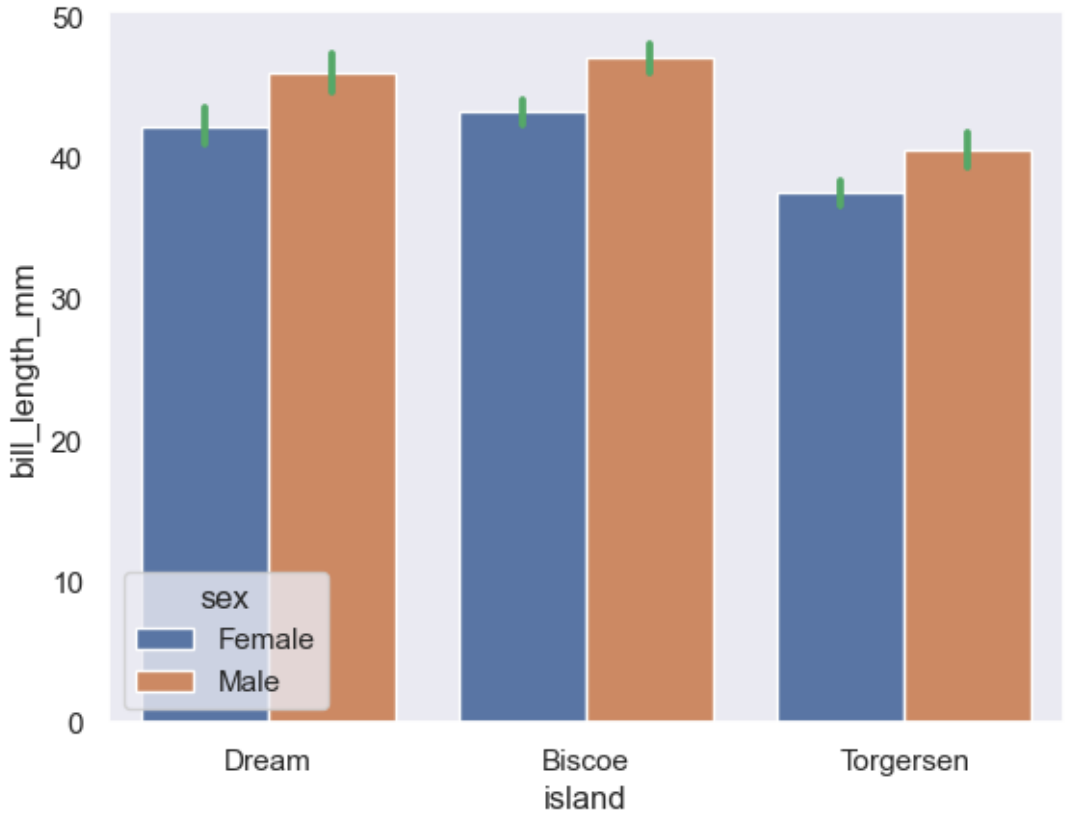


BAR PLOT using Seaborn (darkgrid)

Create a heading: Bar plot

```
sns.set(style='dark')
order_1 = ["Dream", "Biscoe", "Torgersen"]

sns.barplot(x='island', y='bill_length_mm', data=var,
hue='sex', order=order_1, hue_order=["Female",
"Male"], errcolor='g')

plt.show()
```

Output graph



Scatterplot using Seaborn

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

var = sns.load_dataset("penguins")
var
```

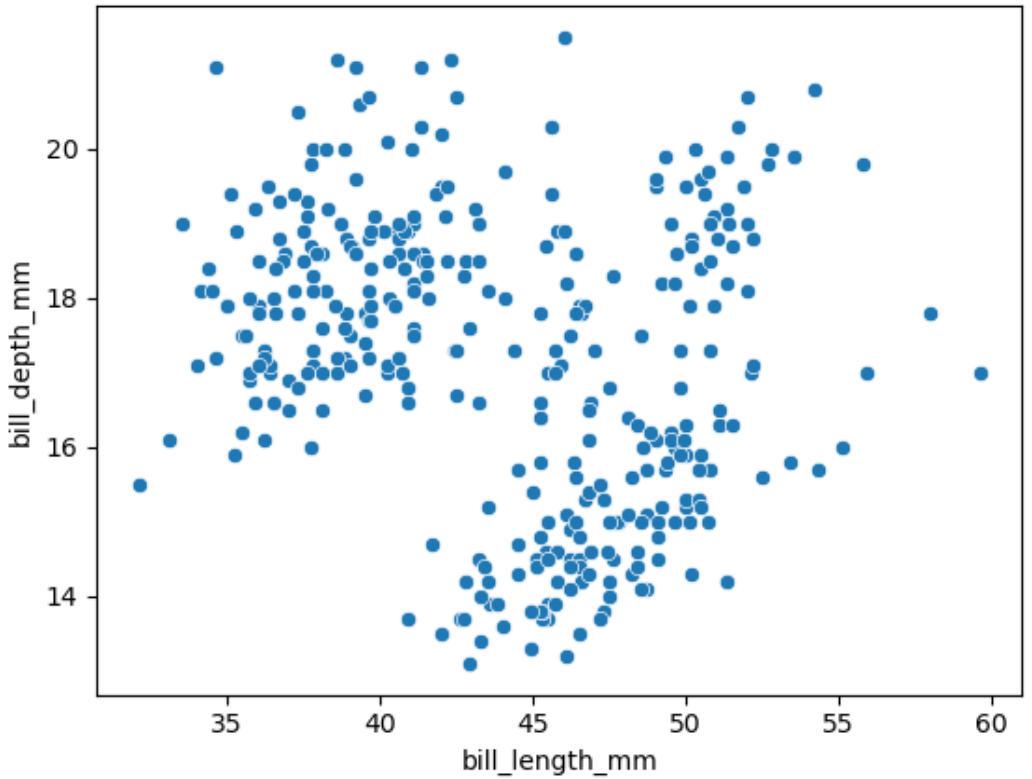
Out[1]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

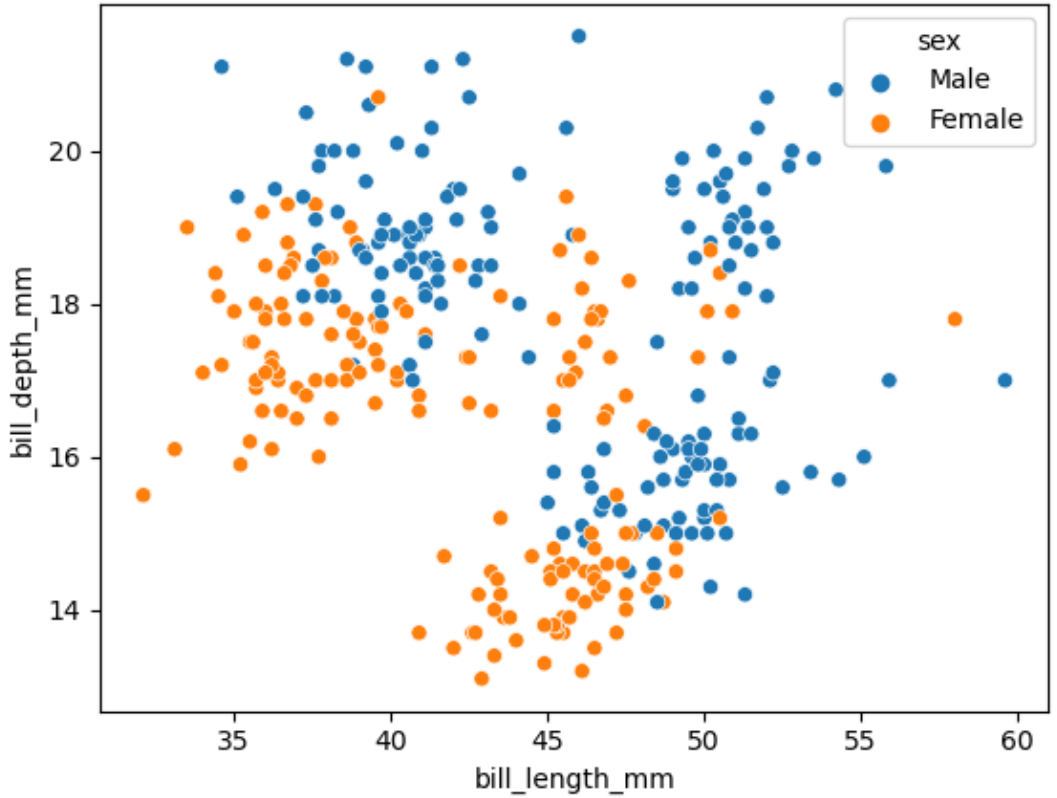
Scatterplot using Seaborn

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var)

plt.show()
```



Scatterplot using Seaborn (hue)

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var, hue='sex')

plt.show()
```



Scatterplot using Seaborn (head)

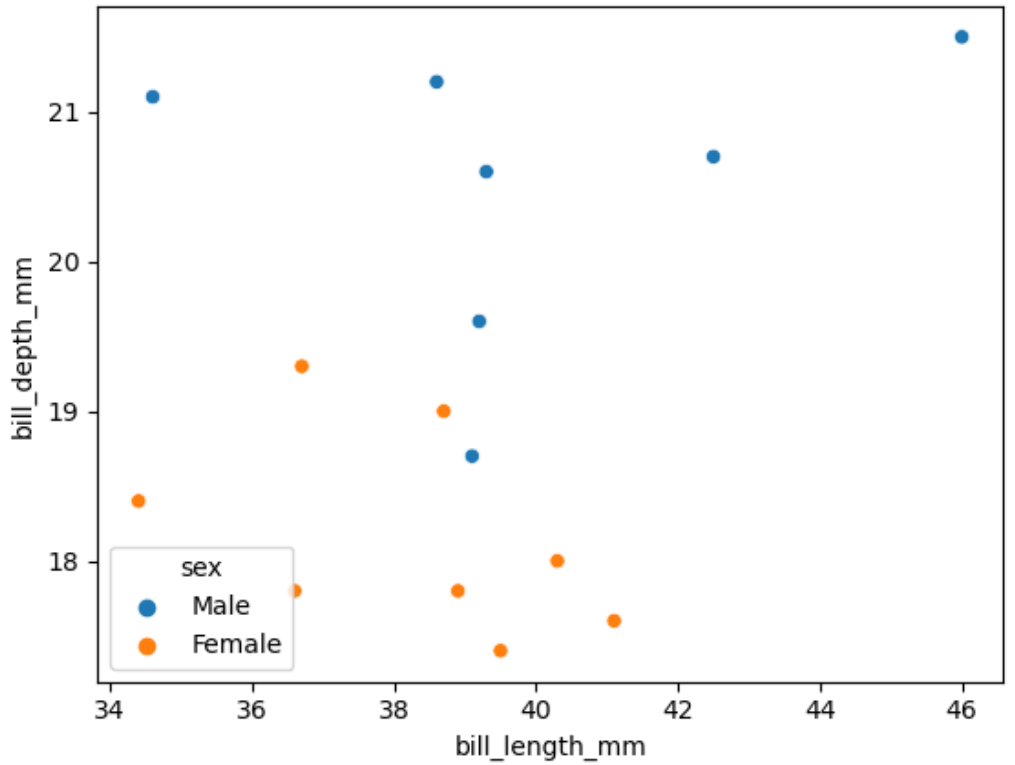
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("penguins").head(20)
var
```

Out[5]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Male
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	Female
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	Male
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN
10	Adelie	Torgersen	37.8	17.1	186.0	3300.0	NaN
11	Adelie	Torgersen	37.8	17.3	180.0	3700.0	NaN
12	Adelie	Torgersen	41.1	17.6	182.0	3200.0	Female
13	Adelie	Torgersen	38.6	21.2	191.0	3800.0	Male
14	Adelie	Torgersen	34.6	21.1	198.0	4400.0	Male
15	Adelie	Torgersen	36.6	17.8	185.0	3700.0	Female
16	Adelie	Torgersen	38.7	19.0	195.0	3450.0	Female
17	Adelie	Torgersen	42.5	20.7	197.0	4500.0	Male
18	Adelie	Torgersen	34.4	18.4	184.0	3325.0	Female
19	Adelie	Torgersen	46.0	21.5	194.0	4200.0	Male

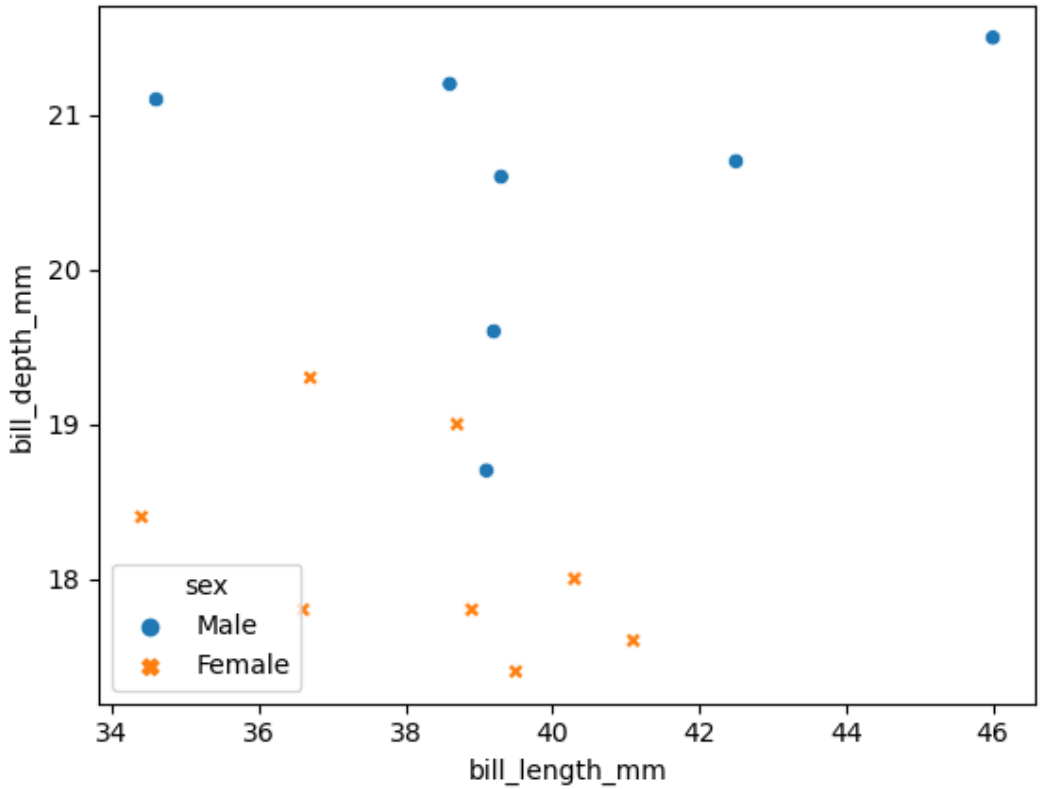
Scatterplot using Seaborn (head)

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var, hue='sex')

plt.show()
```



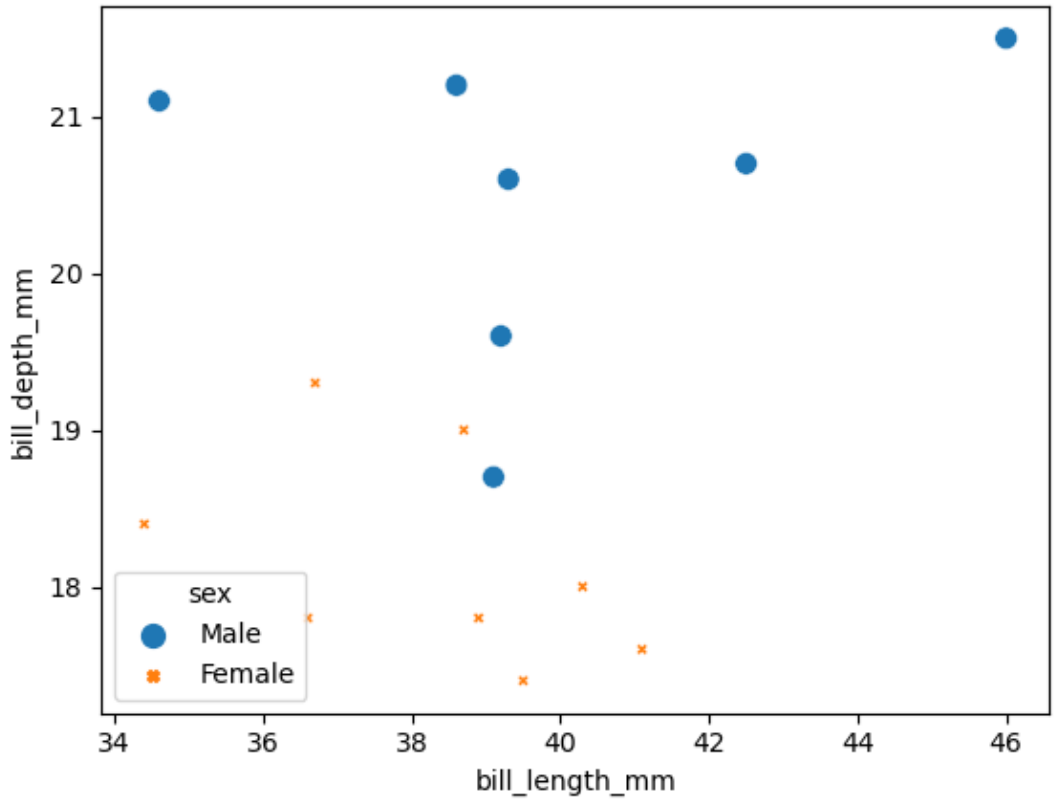
Scatterplot using Seaborn (style)

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var, hue='sex’,
style='sex')

plt.show()
```



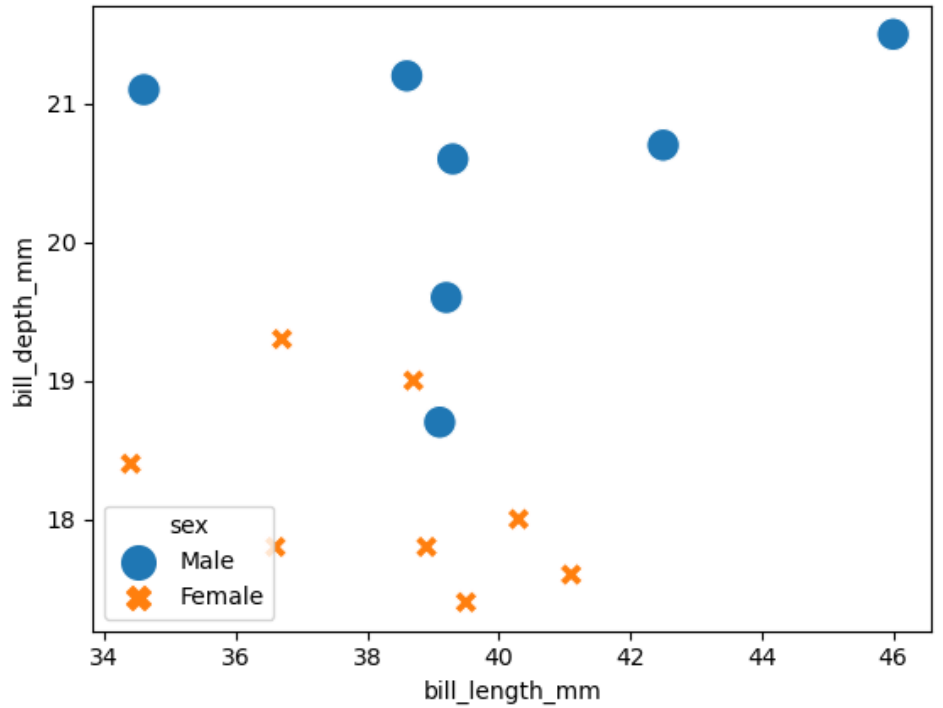
Scatterplot using Seaborn (size)

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var, hue='sex’,
style='sex', size='sex')

plt.show()
```



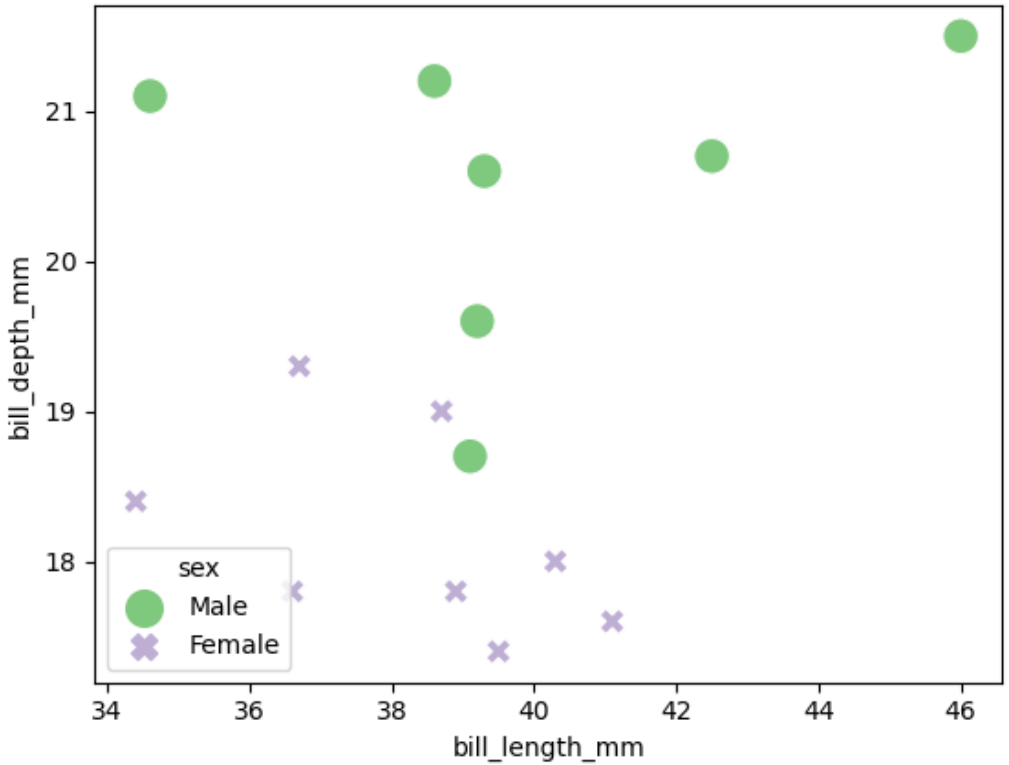
Scatterplot using Seaborn (sizes)

```
sns.scatterplot(x="bill_length_mm", y="bill_depth_mm", data=var, hue='sex',
style='sex', size='sex', sizes =(100,200))

plt.show()
```



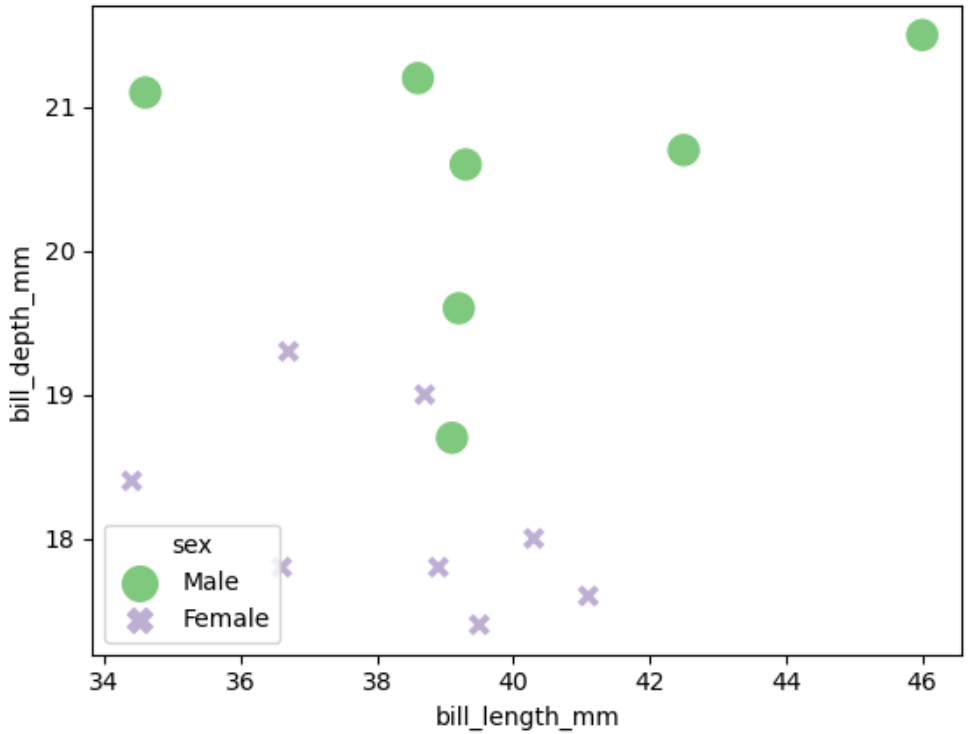
Scatterplot using Seaborn (palette)

```
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var, hue='sex’,
style='sex', size='sex', sizes =(100,200),
palette='Accent')

plt.show()
```



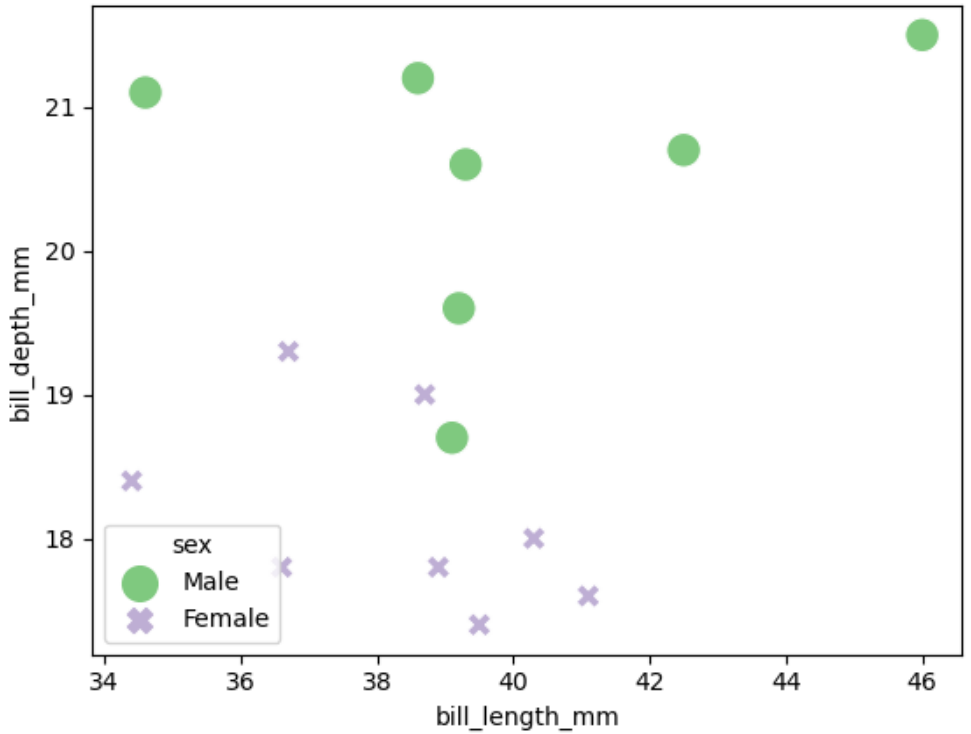
Scatterplot using Seaborn (alpha)

```
sns.scatterplot(x="bill_length_mm", y="bill_depth_mm", data=var, hue='sex',
style='sex', size='sex', sizes =(100,200),
palette='Accent', alpha=0.7)

plt.show()
```



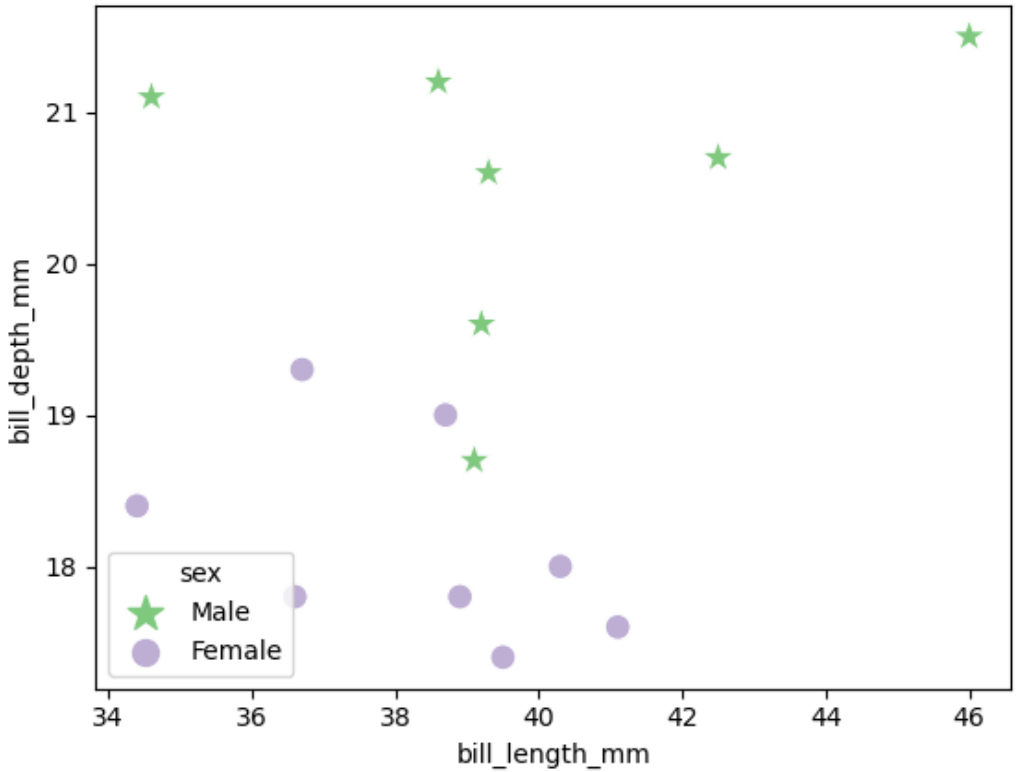
Scatterplot using Seaborn (alpha)

```
sns.scatterplot(x="bill_length_mm", y="bill_depth_mm", data=var, hue='sex',
style='sex', size='sex', sizes =(100,200),
palette='Accent', alpha=0.7)

plt.show()
```



Scatterplot using Seaborn (markers)

```
m={"Male":"*", "Female":"o"}
sns.scatterplot(x="bill_length_mm",
y="bill_depth_mm", data=var,
hue='sex', style='sex', size='sex’,
sizes =(100,200), palette='Accent’,
alpha=1, markers=m)

plt.show()
```



Heatmap using Seaborn

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

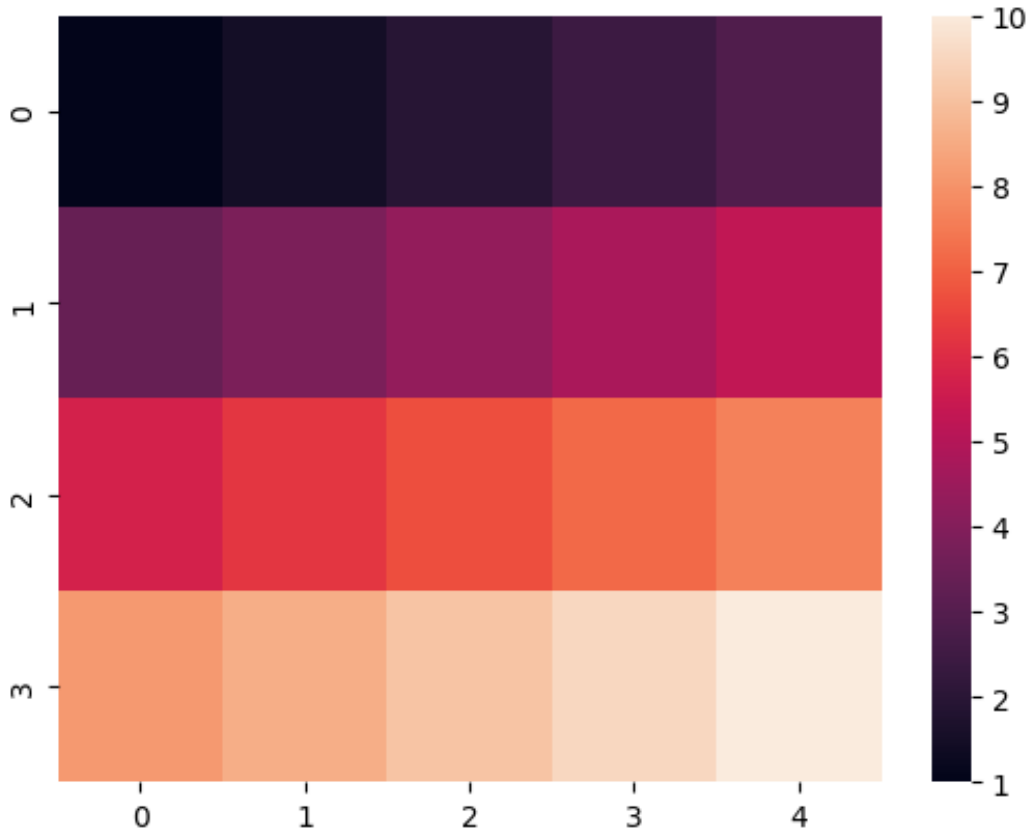
```
Out[1]: array([[ 1.        ,  1.47368421,  1.94736842,  2.42105263,  2.89473684],
               [ 3.36842105,  3.84210526,  4.31578947,  4.78947368,  5.26315789],
               [ 5.73684211,  6.21052632,  6.68421053,  7.15789474,  7.63157895],
               [ 8.10526316,  8.57894737,  9.05263158,  9.52631579, 10.        ]])
```

```
#creating an array using numpy library through linspace function

var=np.linspace(1,10,20).reshape(4,5)

var
```

Heatmap using Seaborn

sns.heatmap(var)

plt.show()



Heatmap using Seaborn

```
data = sns.load_dataset('anagrams’)
data
```

	subidr	attnr	num1	num2	num3
0	1	divided	2	4.0	7
1	2	divided	3	4.0	5
2	3	divided	3	5.0	6
3	4	divided	5	7.0	5
4	5	divided	4	5.0	8
5	6	divided	5	5.0	6
6	7	divided	5	4.5	6
7	8	divided	5	7.0	8
8	9	divided	2	3.0	7
9	10	divided	6	5.0	6
10	11	focused	6	5.0	6
11	12	focused	8	9.0	8
12	13	focused	6	5.0	9
13	14	focused	8	8.0	7
14	15	focused	8	8.0	7
15	16	focused	6	8.0	7
16	17	focused	7	7.0	6
17	18	focused	7	8.0	6
18	19	focused	5	6.0	6
19	Reminders		6	6.0	5

Heatmap using Seaborn

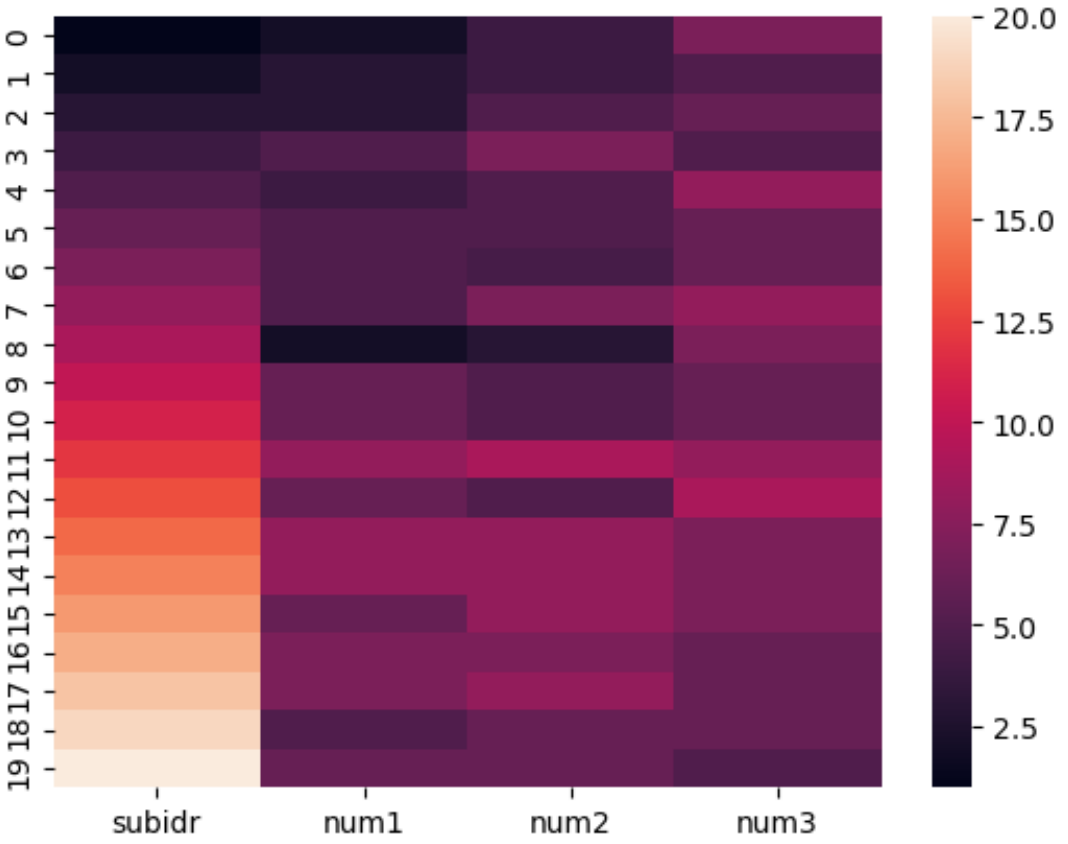
```
data = sns.load_dataset('anagrams')
x = data.drop(columns=["attnr"], axis=1)

x
```

	subidr	num1	num2	num3
0	1	2	4.0	7
1	2	3	4.0	5
2	3	3	5.0	6
3	4	5	7.0	5
4	5	4	5.0	8
5	6	5	5.0	6
6	7	5	4.5	6
7	8	5	7.0	8
8	9	2	3.0	7
9	10	6	5.0	6
10	11	6	5.0	6
11	12	8	9.0	8
12	13	6	5.0	9
13	14	8	8.0	7
14	15	8	8.0	7
15	16	6	8.0	7
16	17	7	7.0	6
17	18	7	8.0	6
18	19	5	6.0	6
19	20	6	6.0	5

Heatmap using Seaborn

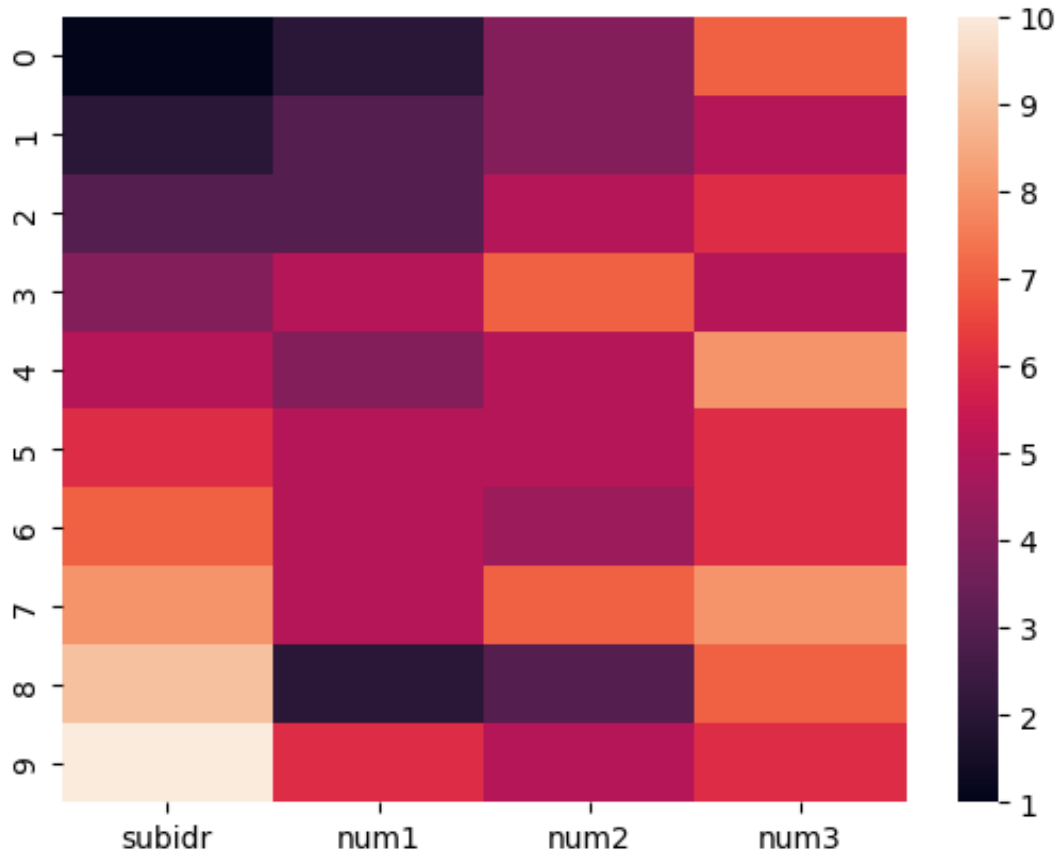
sns.heatmap(x)

plt.show()



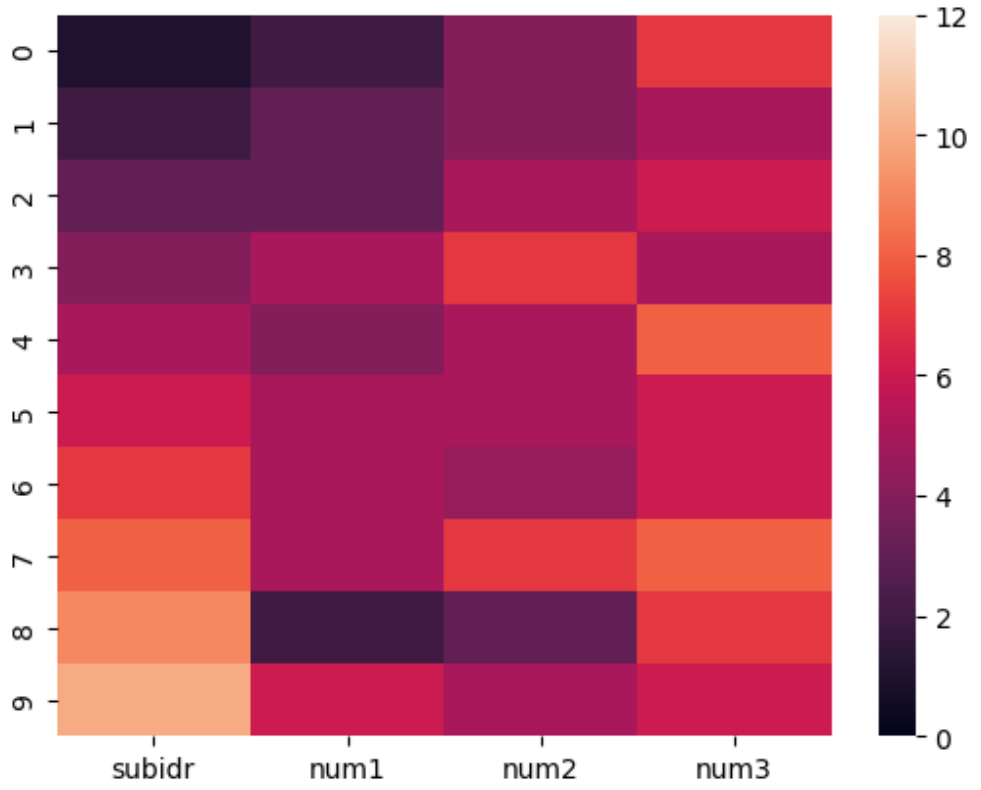
Heatmap using Seaborn

```
data = sns.load_dataset('anagrams').head(10)
x = data.drop(columns=["attnr"], axis=1)
sns.heatmap(x)

plt.show()
```



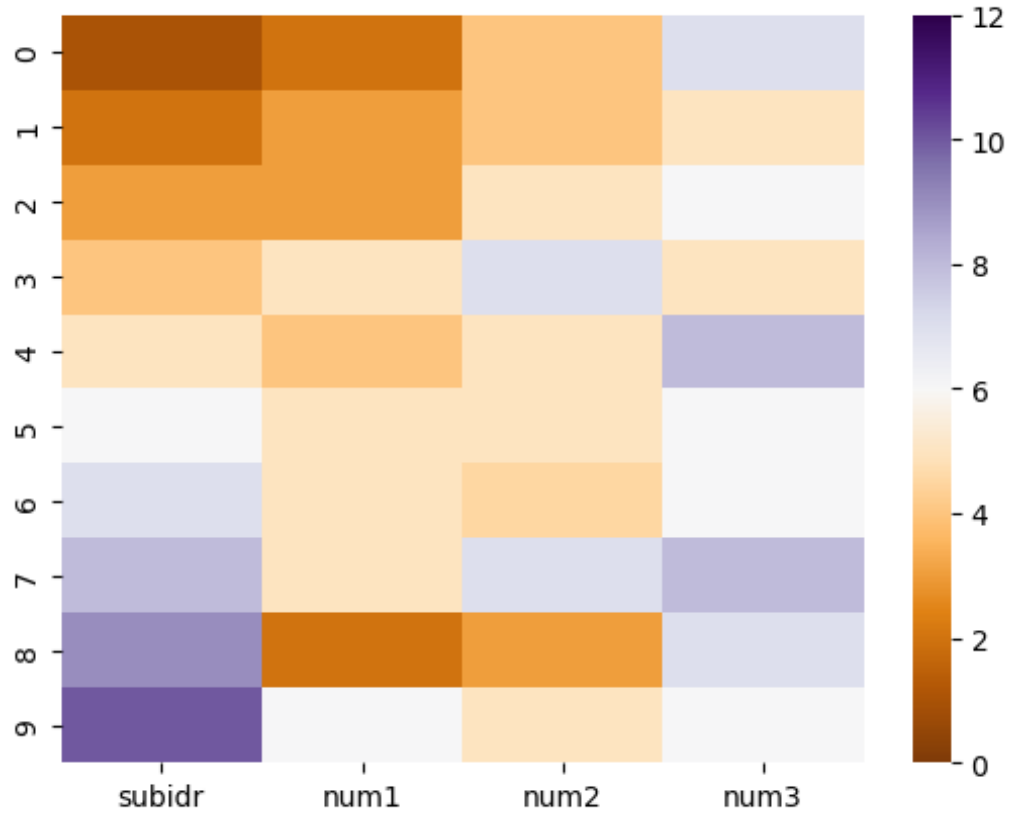
Heatmap using Seaborn (min, max)

```
data = sns.load_dataset('anagrams').head(10)
x = data.drop(columns=["attnr"], axis=1)
sns.heatmap(x, vmin=0, vmax=12)

plt.show()
```



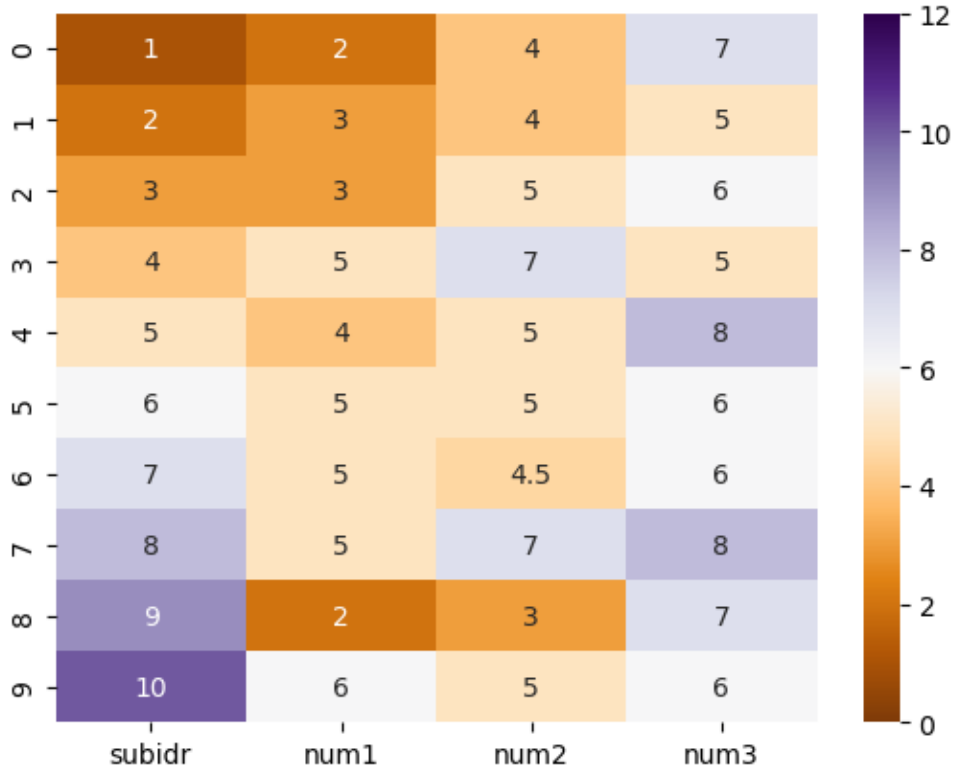
Heatmap using Seaborn (cmap)

```
data = sns.load_dataset('anagrams').head(10)
x = data.drop(columns=["attnr"], axis=1)
sns.heatmap(x, vmin=0, vmax=12, cmap='PuOr')

plt.show()
```



Heatmap using Seaborn (annot)

```
data = sns.load_dataset('anagrams').head(10)

x = data.drop(columns=["attnr"], axis=1)

sns.heatmap(x, vmin=0, vmax=12, cmap='PuOr', annot=True)


plt.show()
```



Heatmap using Seaborn (annot)

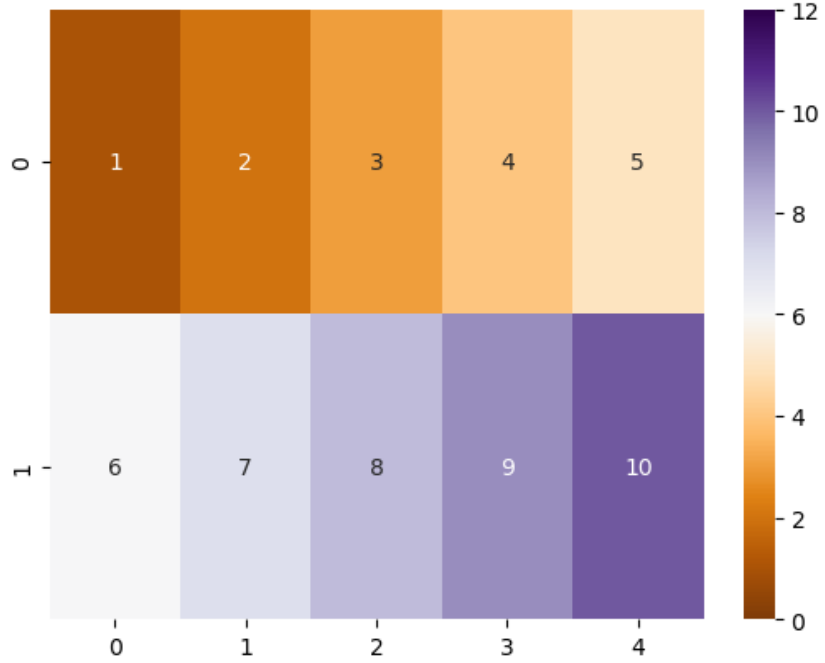
var1=np.linspace(1,10,10).reshape(2,5)

var1

```
Out[23]: array([[ 1.,  2.,  3.,  4.,  5.],
                [ 6.,  7.,  8.,  9., 10.]])
```

Heatmap using Seaborn (annot)

```
sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=True)

plt.show()
```



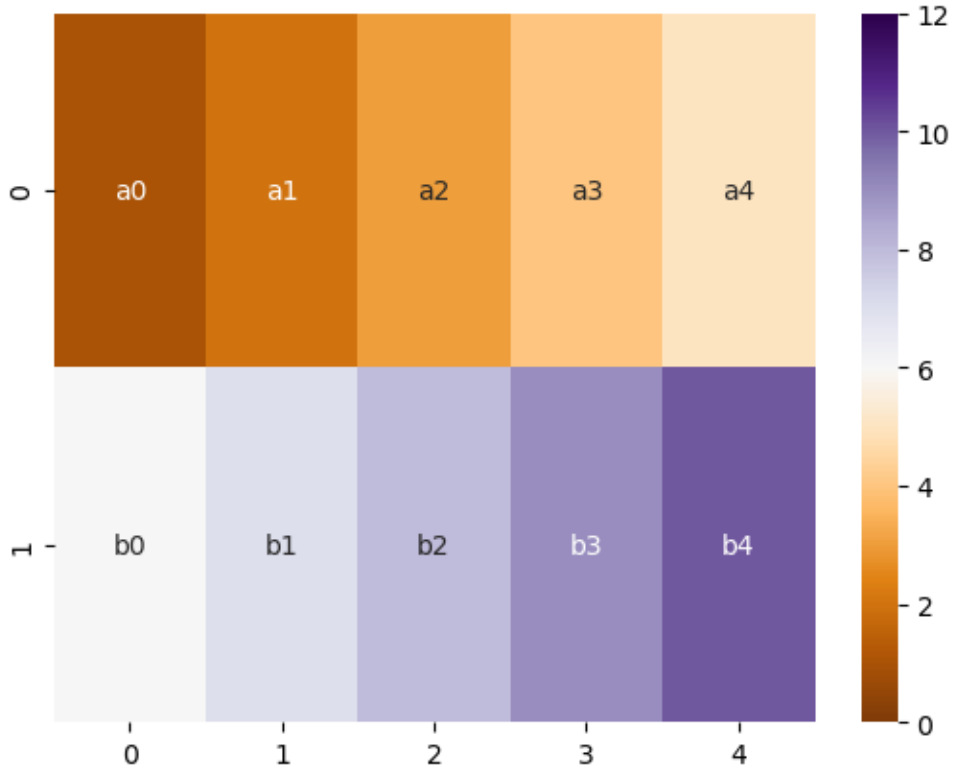
Heatmap using Seaborn (annot)

```
#creating a new array

ar = np.array([["a0","a1","a2","a3","a4"],
               ["b0","b1","b2","b3","b4"]])

ar
```

```
Out[30]: array([['a0', 'a1', 'a2', 'a3', 'a4'],
                ['b0', 'b1', 'b2', 'b3', 'b4']], dtype='<U2')
```

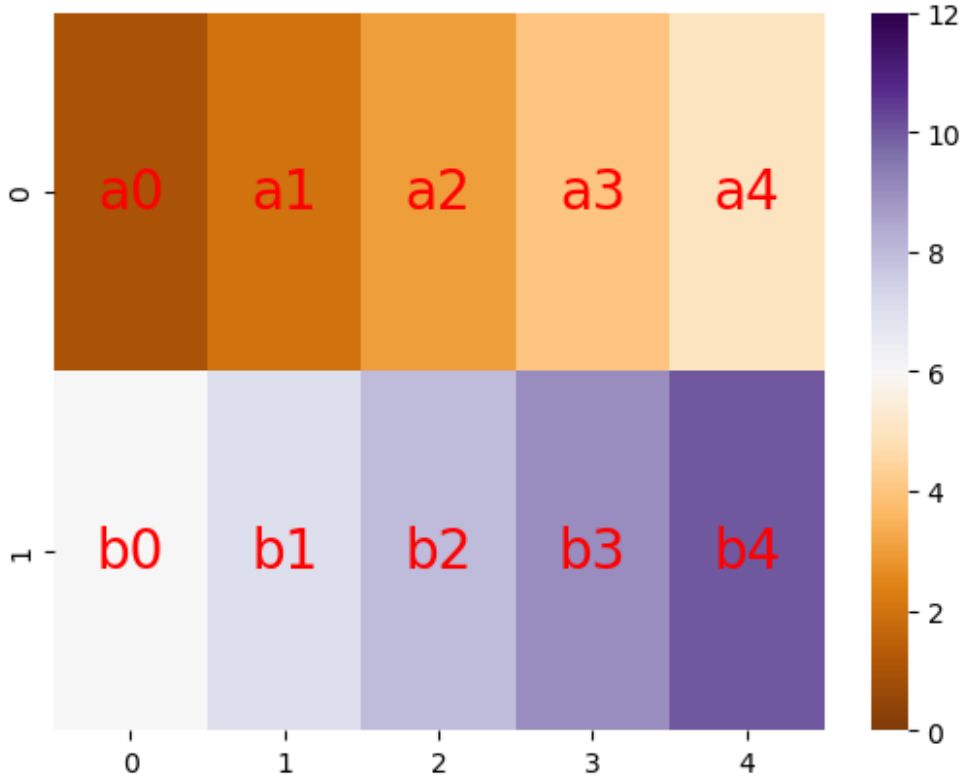
Heatmap using Seaborn (annot)

```
sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s')

plt.show()
```



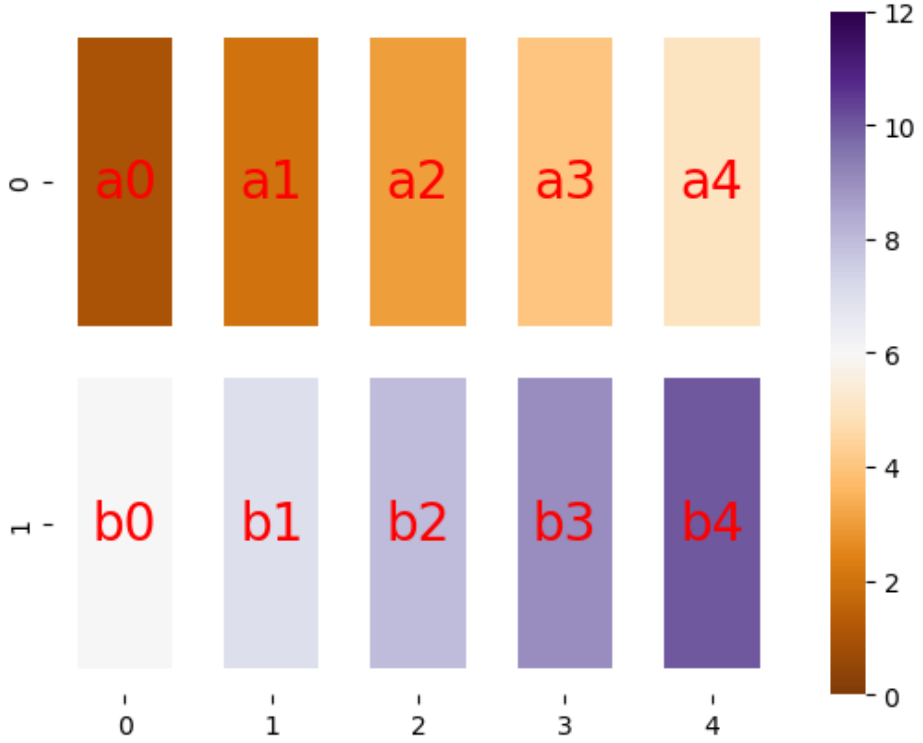
Heatmap using Seaborn (annot_kws)

```
y={"fontsize":20, "color":'r'}

sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y)


plt.show()
```



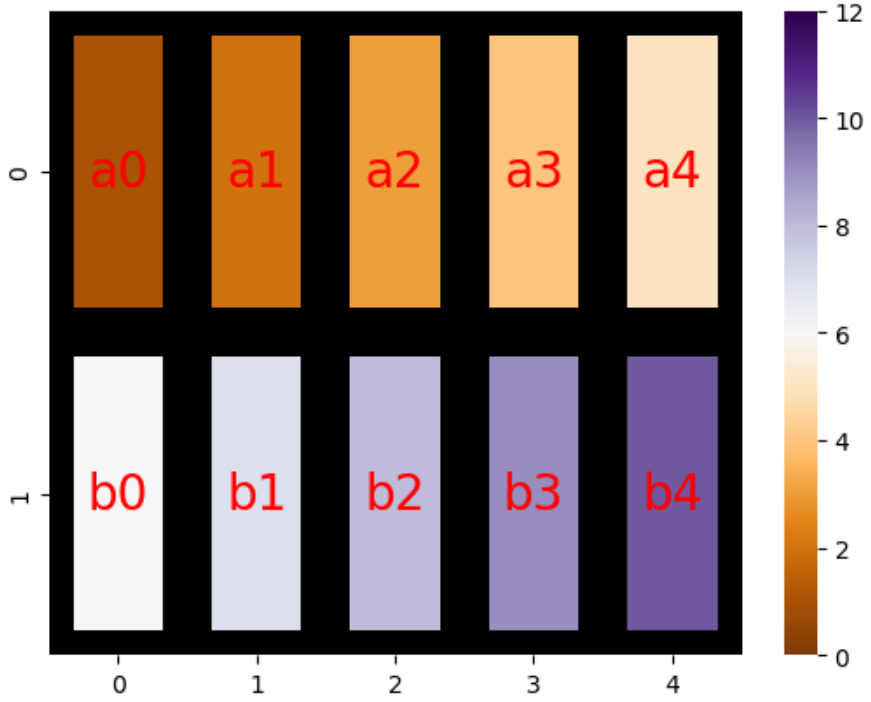
Heatmap using Seaborn (linewidth)

```
y={"fontsize":20, "color":'r'}

sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20)


plt.show()
```



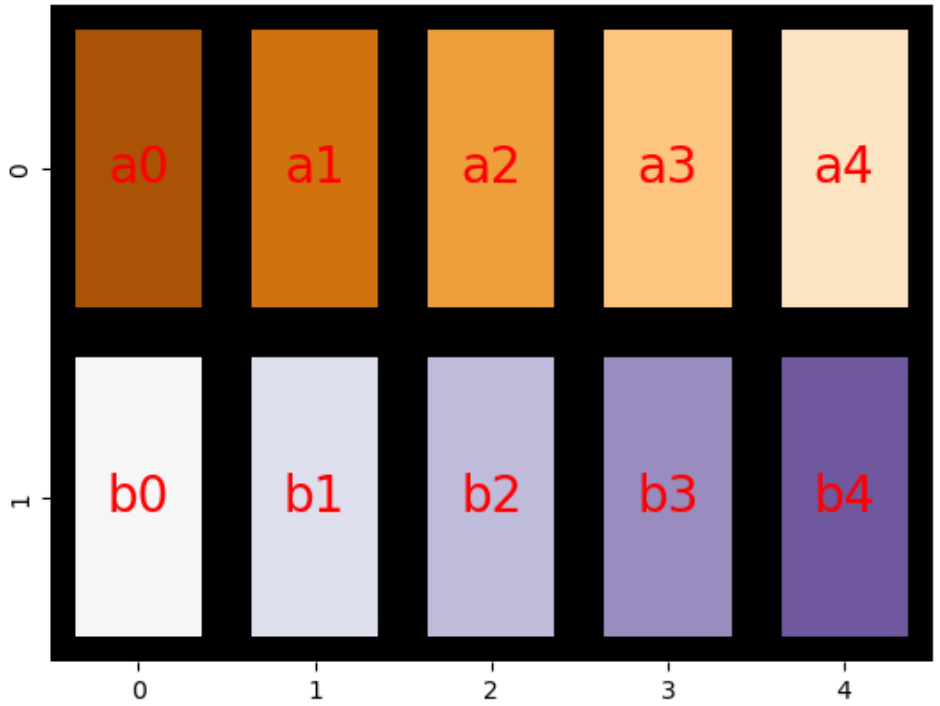
Heatmap using Seaborn (linecolor)

```
y={"fontsize":20, "color":'r'}

sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20,
linecolor='k')


plt.show()
```



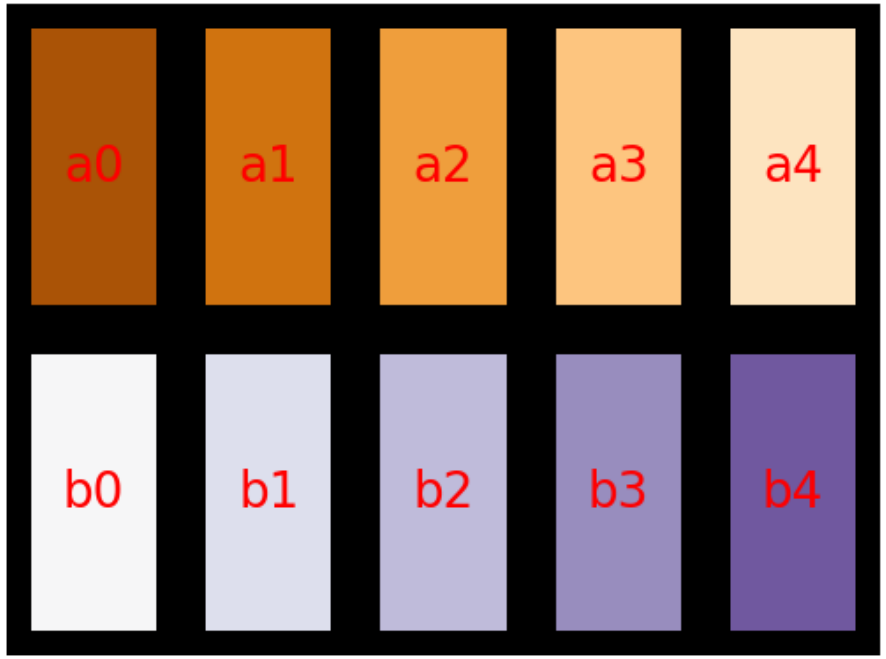
Heatmap using Seaborn (colorbar)

```
y={"fontsize":20, "color":'r'}

sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20,

linecolor='k', cbar=False)


plt.show()
```



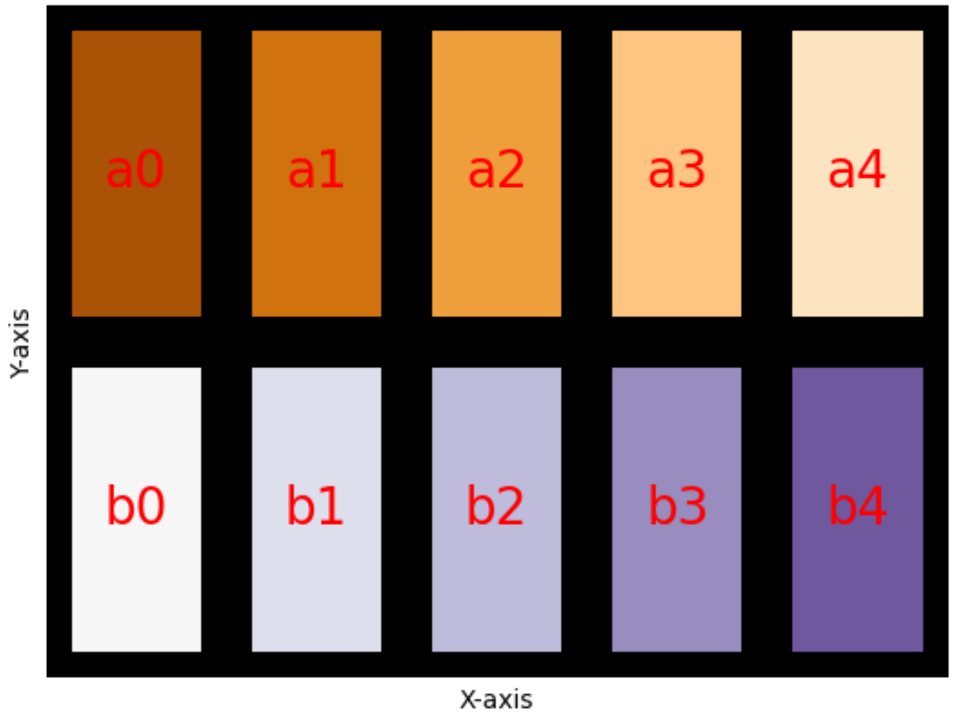
Heatmap using Seaborn (xtick, ytick)

```
y={"fontsize":20, "color":'r'}

sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20,

linecolor='k', cbar=False, xticklabels=False,
yticklabels=False)


plt.show()
```



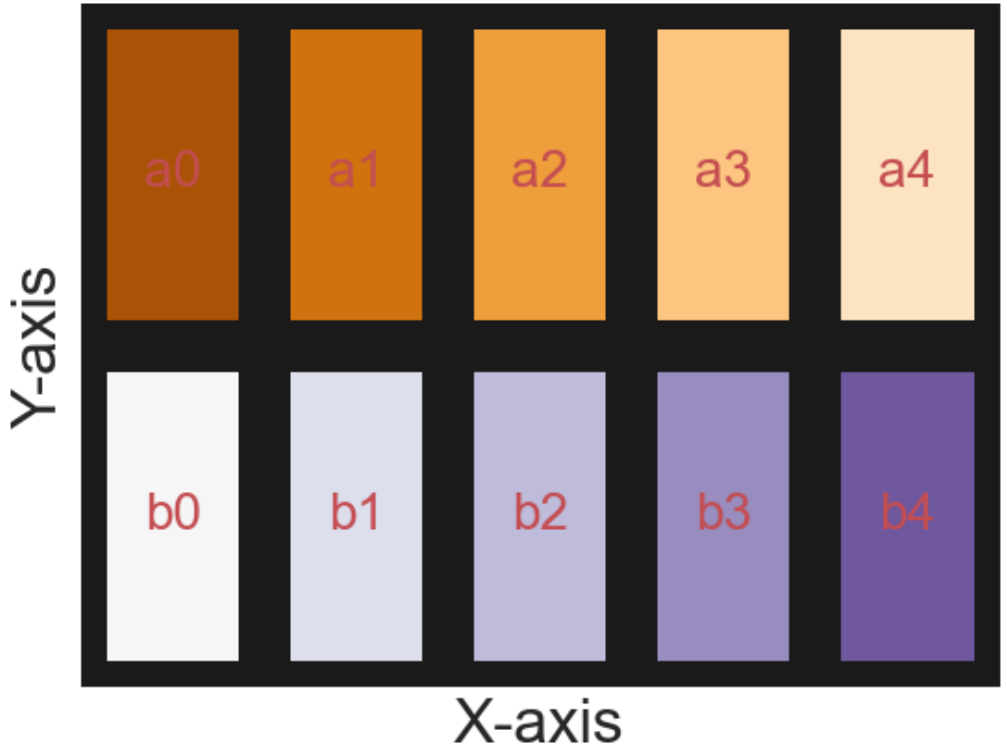
Heatmap using Seaborn (a and y axis)

```
y={"fontsize":20, "color":'r'}

v = sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20,

linecolor='k', cbar=False, xticklabels=False,
yticklabels=False)

v.set(xlabel='X-axis', ylabel='Y-axis')


plt.show()
```



Heatmap using Seaborn (a and y axis)

```
y={"fontsize":20, "color":'r'}

v = sns.heatmap(var1, vmin=0, vmax=12, cmap='PuOr',
annot=ar,fmt='s', annot_kws=y, linewidth=20,

linecolor='k', cbar=False, xticklabels=False,
yticklabels=False)

v.set(xlabel='X-axis', ylabel='Y-axis')

sns.set(font_scale=5)


plt.show()
```



Count vs Bar plot. What is the difference?

Count plot using Seaborn

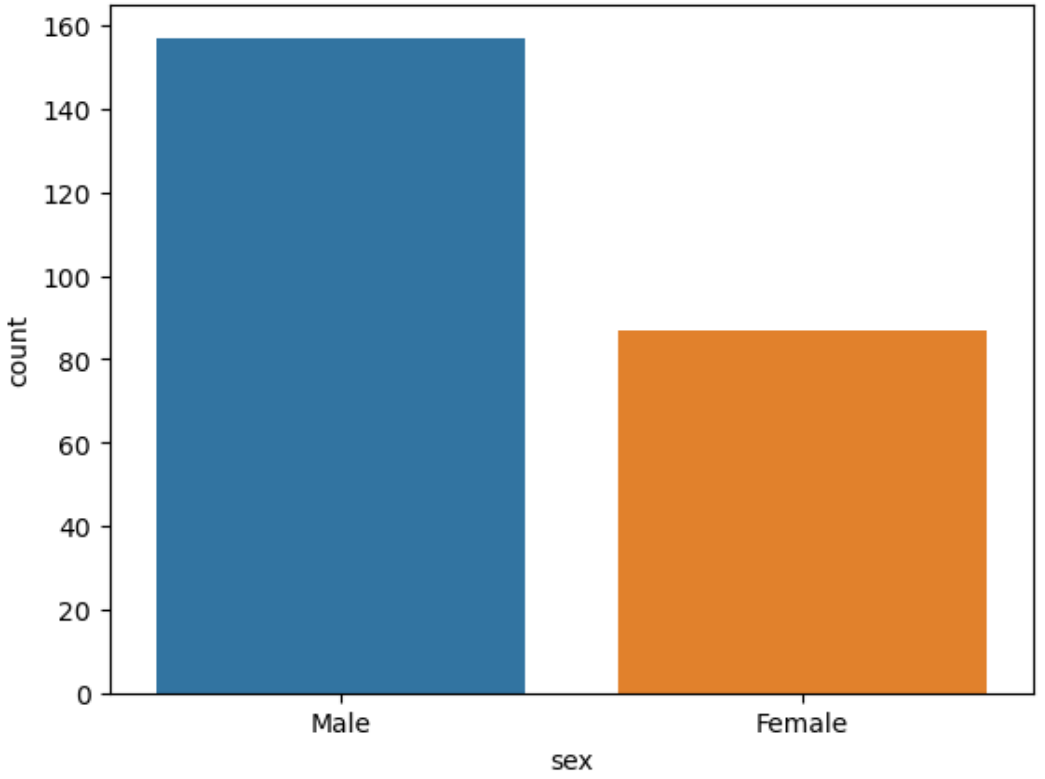
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

Count plot using Seaborn

```
sns.countplot(x="sex", data=var)

plt.show()
```

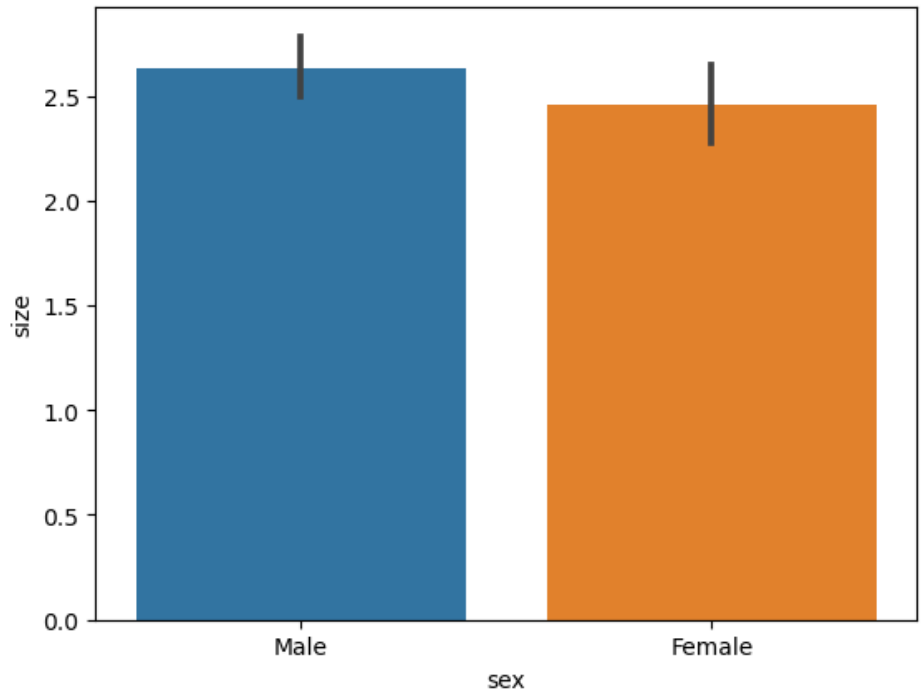


Similar but not same result using bar plot

#using barplot

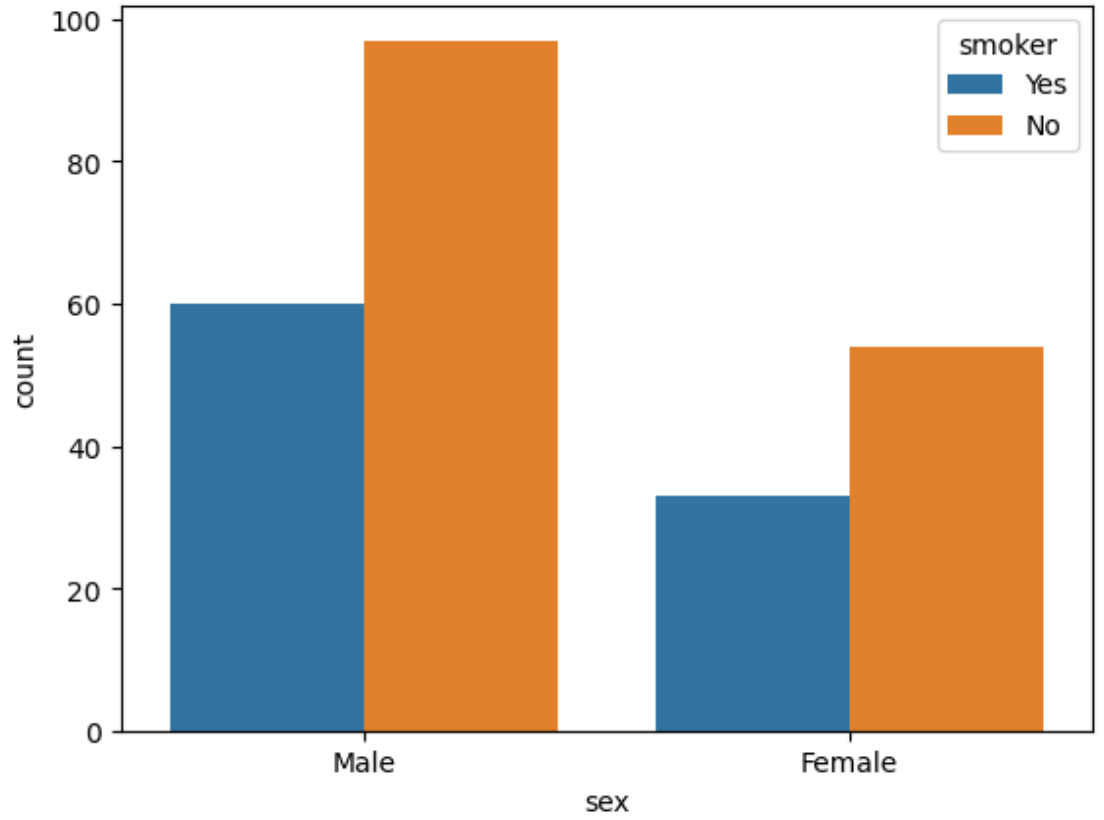
sns.barplot(x='sex', y="size", data=var)

plt.show()



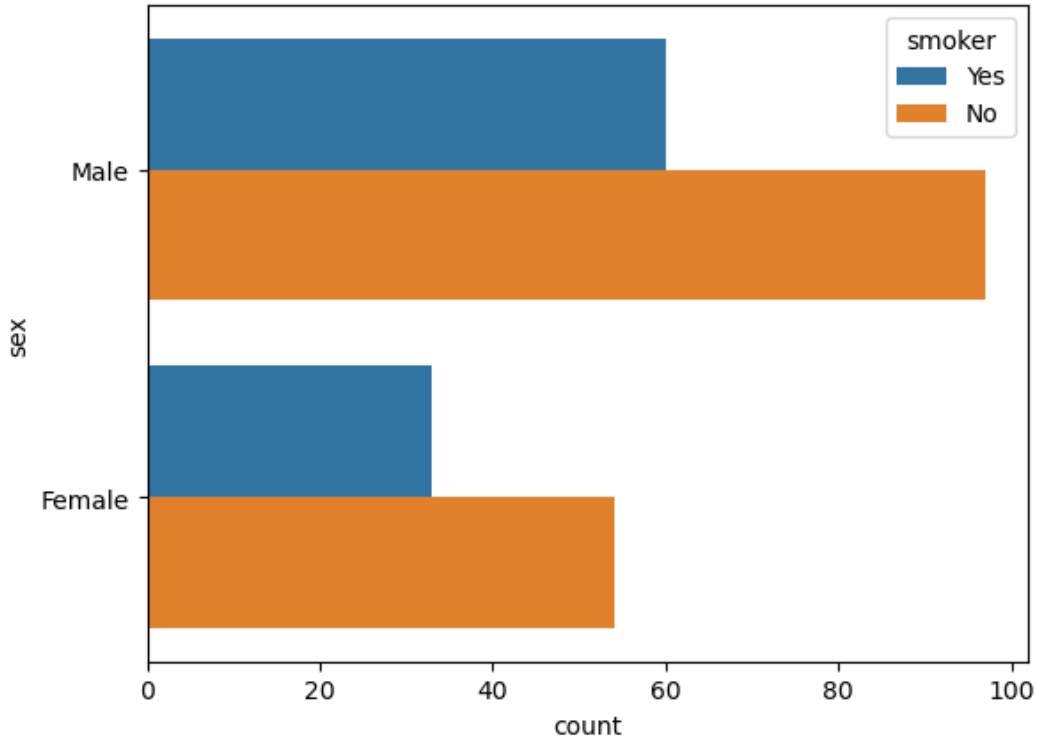
Count plot using Seaborn (hue)

```
sns.countplot(x="sex", data=var,
hue='smoker')

plt.show()
```



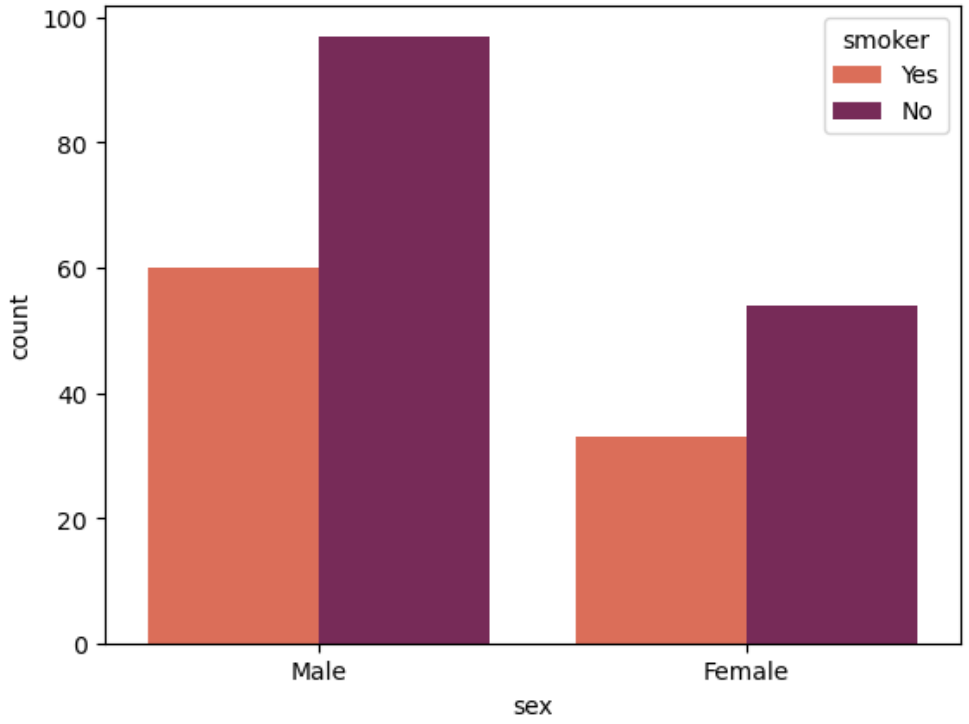
Count plot using Seaborn (horizontal)

```
#horizontal plot


sns.countplot(y="sex", data=var,
hue='smoker')


plt.show()
```



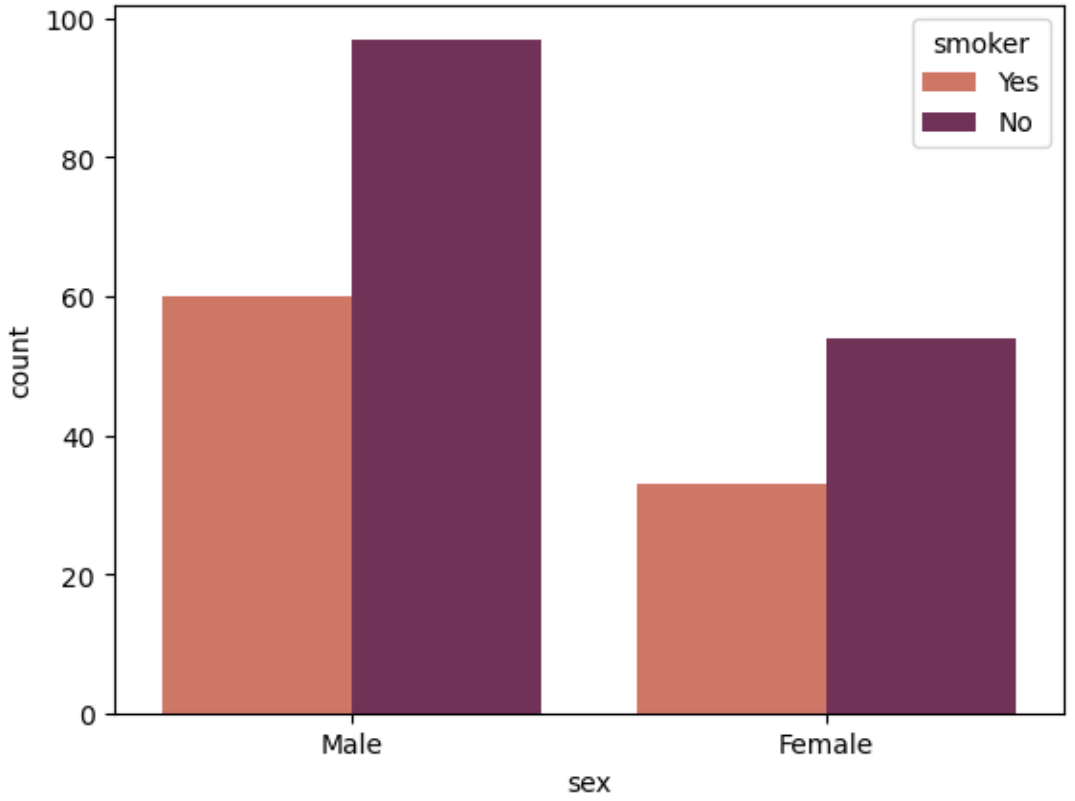
Count plot using Seaborn (palette)

```
sns.countplot(x="sex", data=var,
hue='smoker', palette='rocket_r')

plt.show()
```



Count plot using Seaborn (saturation)

```
sns.countplot(x="sex", data=var,
hue='smoker', palette='rocket_r',
saturation=0.6)

plt.show()
```



Violin plot using Seaborn

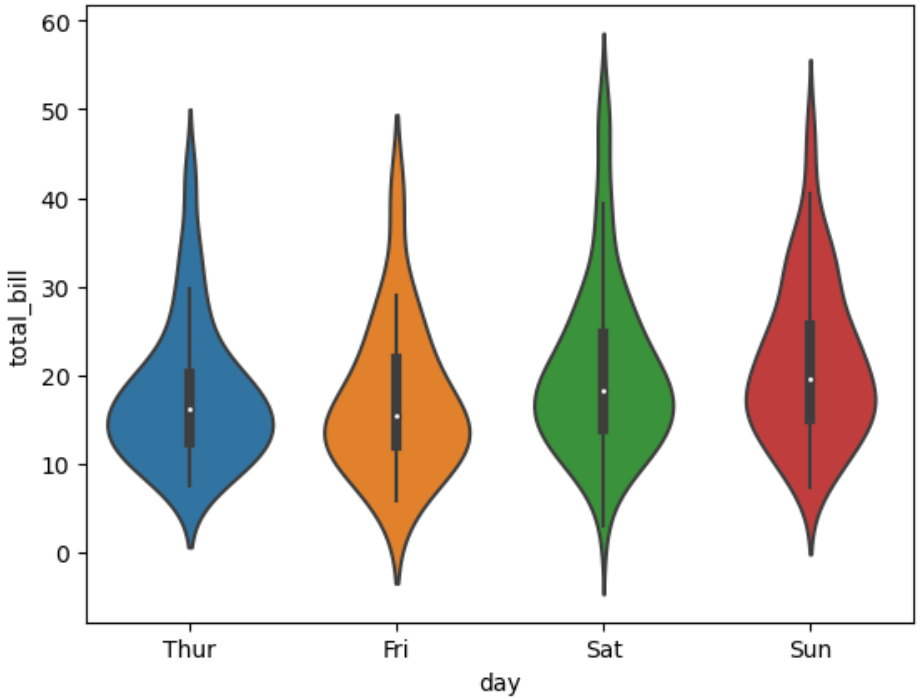
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

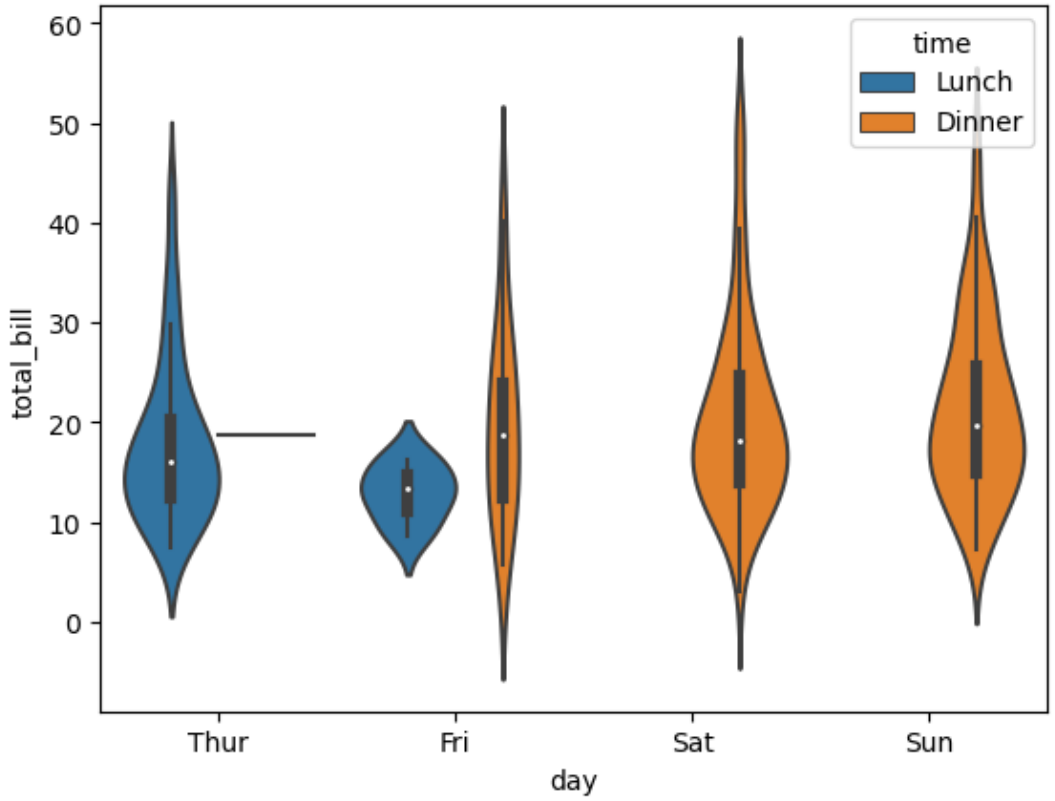
Violin plot using Seaborn

```
sns.violinplot(x='day', y='total_bill',
data = var)

plt.show()
```



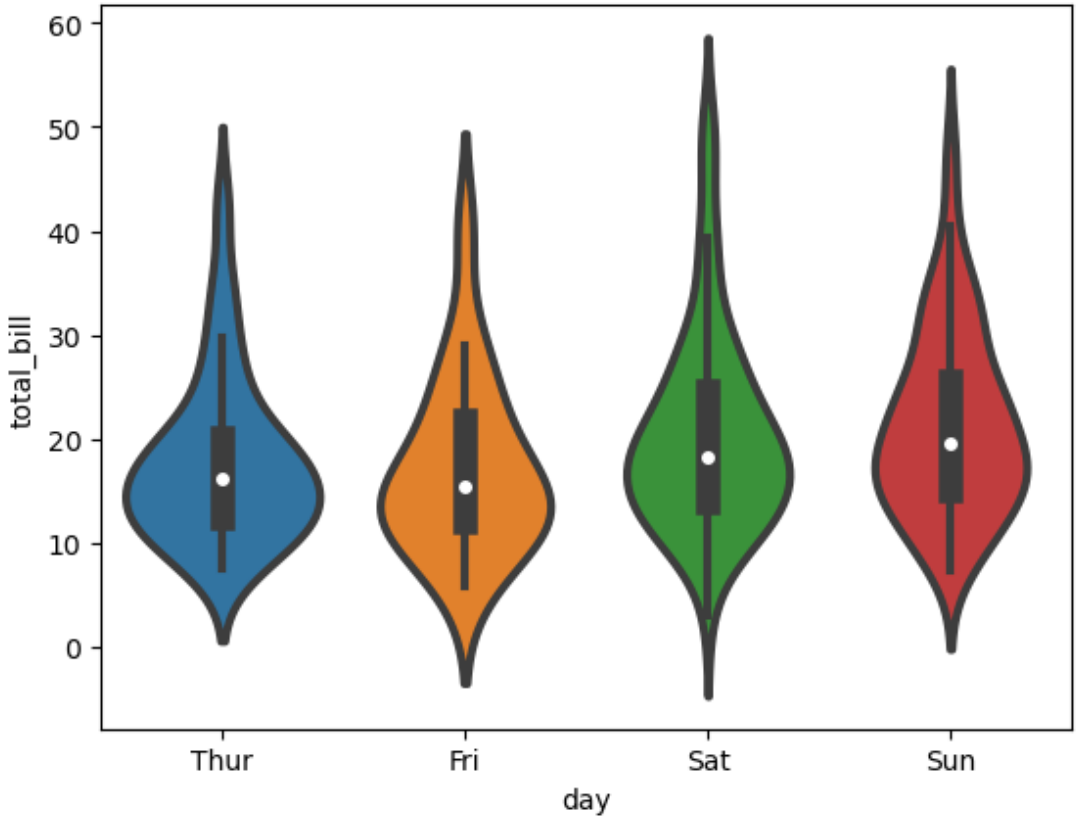
Violin plot using Seaborn (hue)

```
sns.violinplot(x='day', y='total_bill',
data = var, hue='time')

plt.show()
```



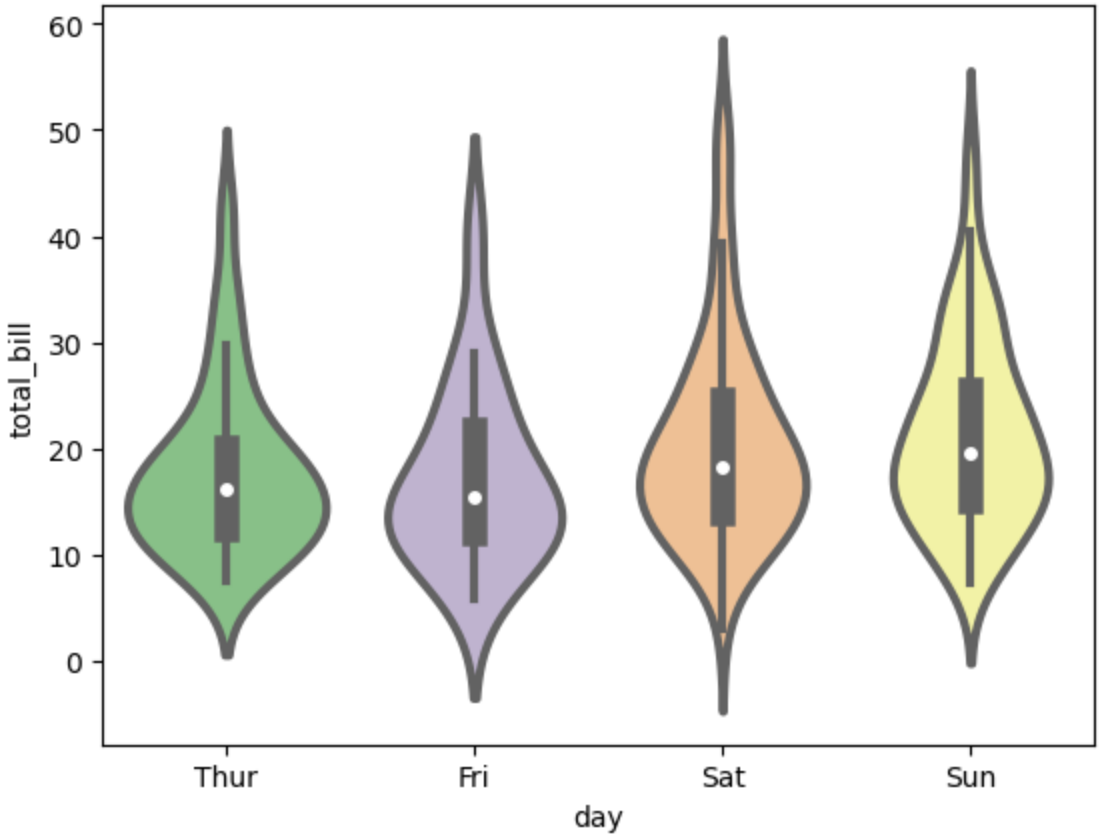
Violin plot using Seaborn (linewidth)

```
sns.violinplot(x='day', y='total_bill',
data = var, linewidth=3)

plt.show()
```



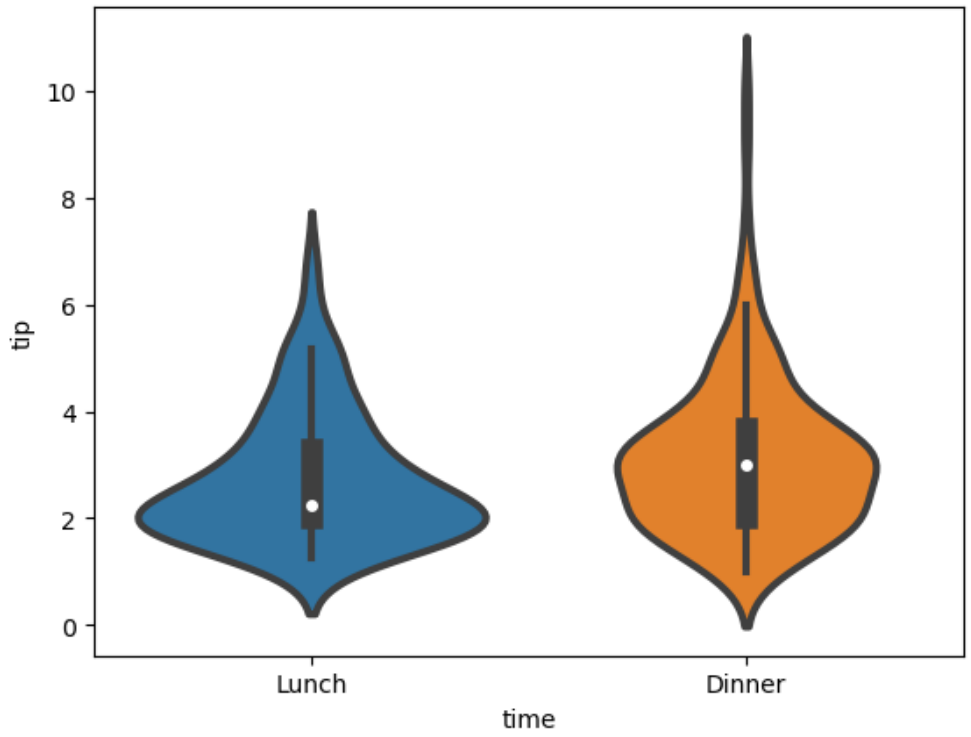
Violin plot using Seaborn (palette)

```
sns.violinplot(x='day', y='total_bill',
data = var, linewidth=3,
palette='Accent')


plt.show()
```



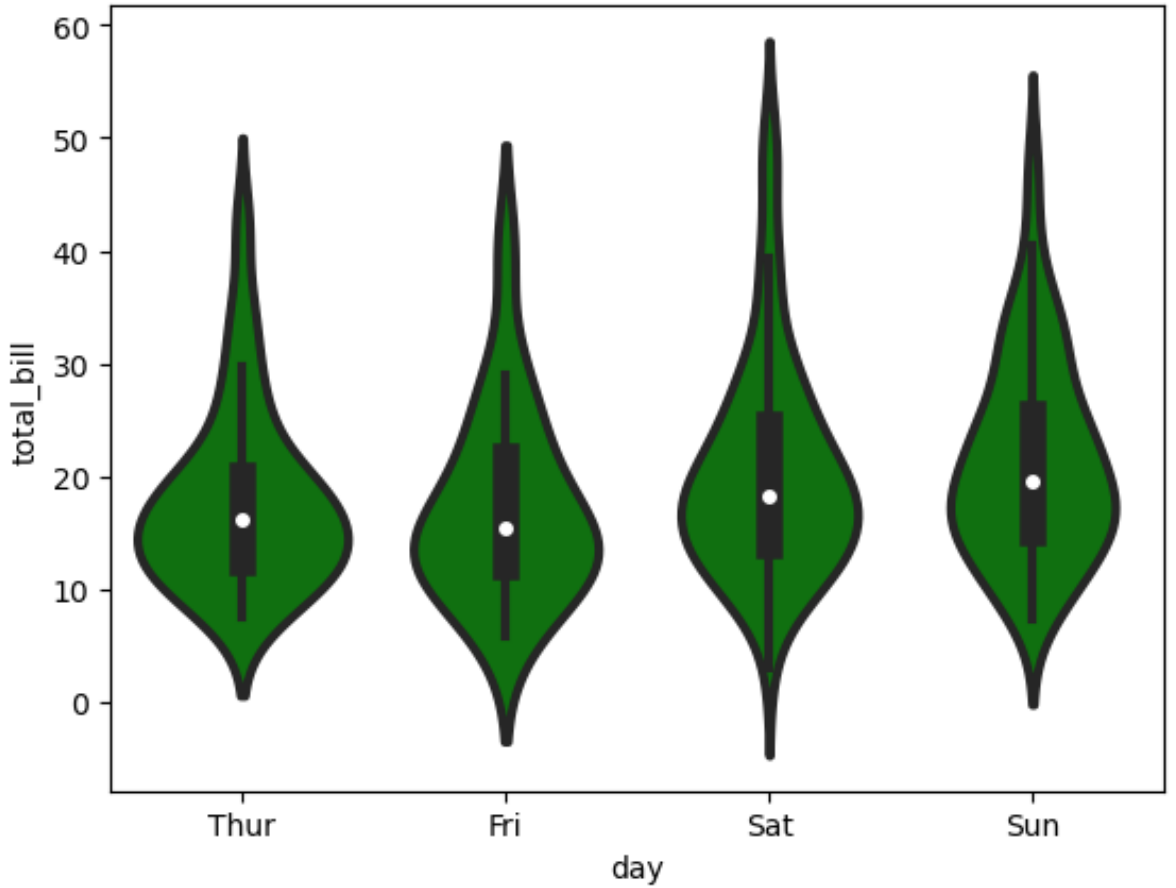
Violin plot using Seaborn (order)

```
sns.violinplot(x='time', y='tip', data =
var, linewidth=3,
order=["Lunch","Dinner"])

plt.show()
```



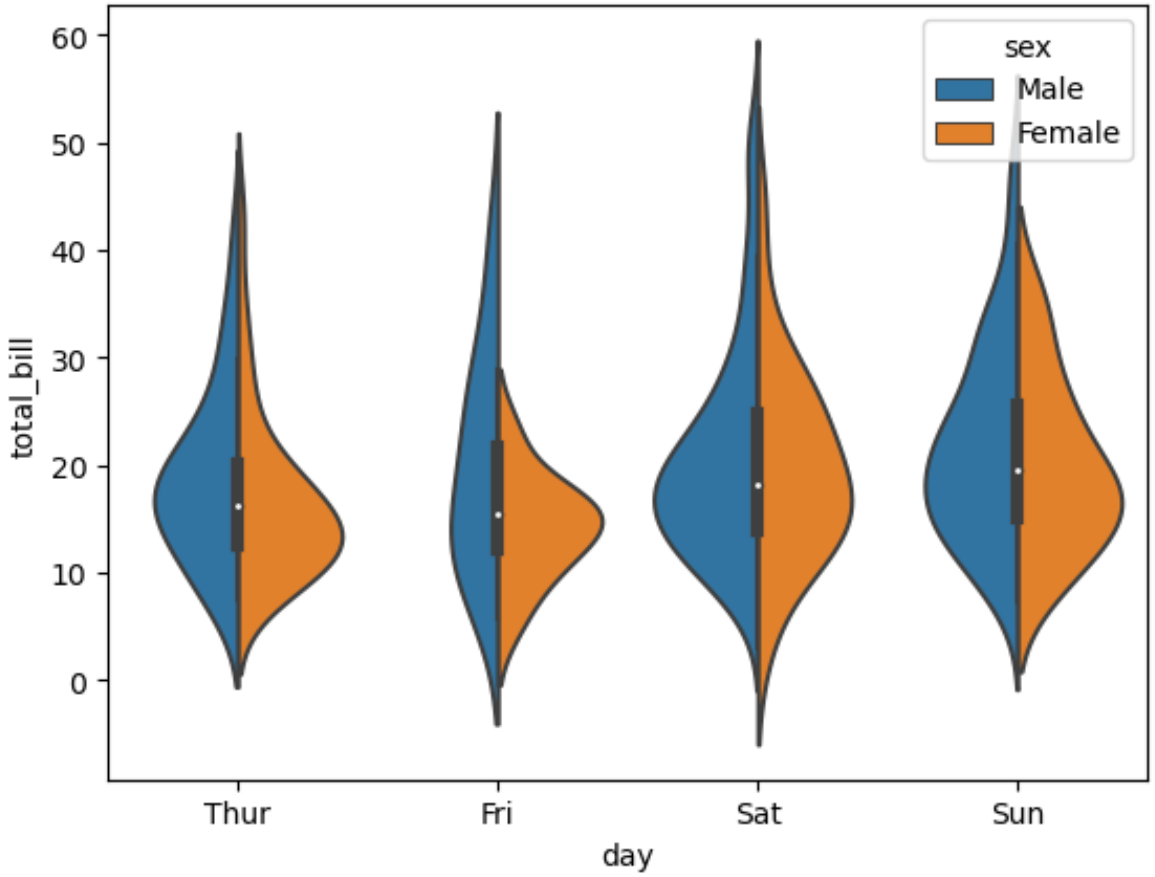
Violin plot using Seaborn (color)

```
sns.violinplot(x='day', y='total_bill',
data = var, linewidth=3, color='g')

plt.show()
```



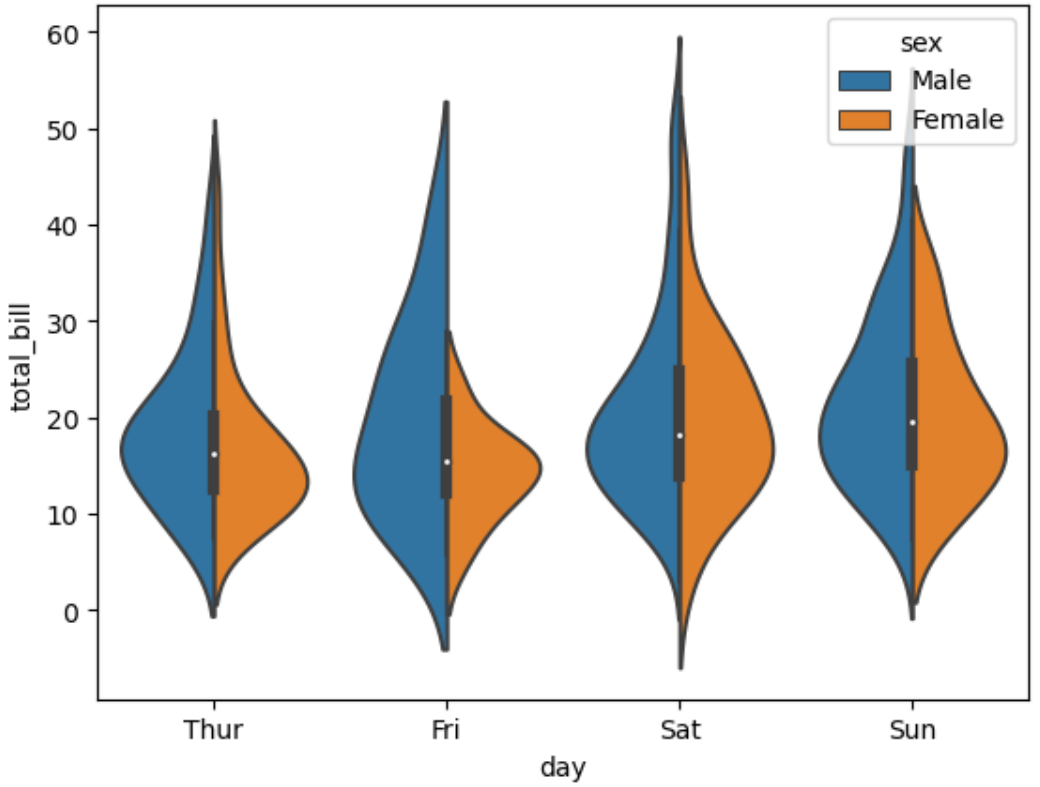
Violin plot using Seaborn (split)

```
sns.violinplot(x='day', y='total_bill',
data = var, hue='sex', split=True)

plt.show()
```



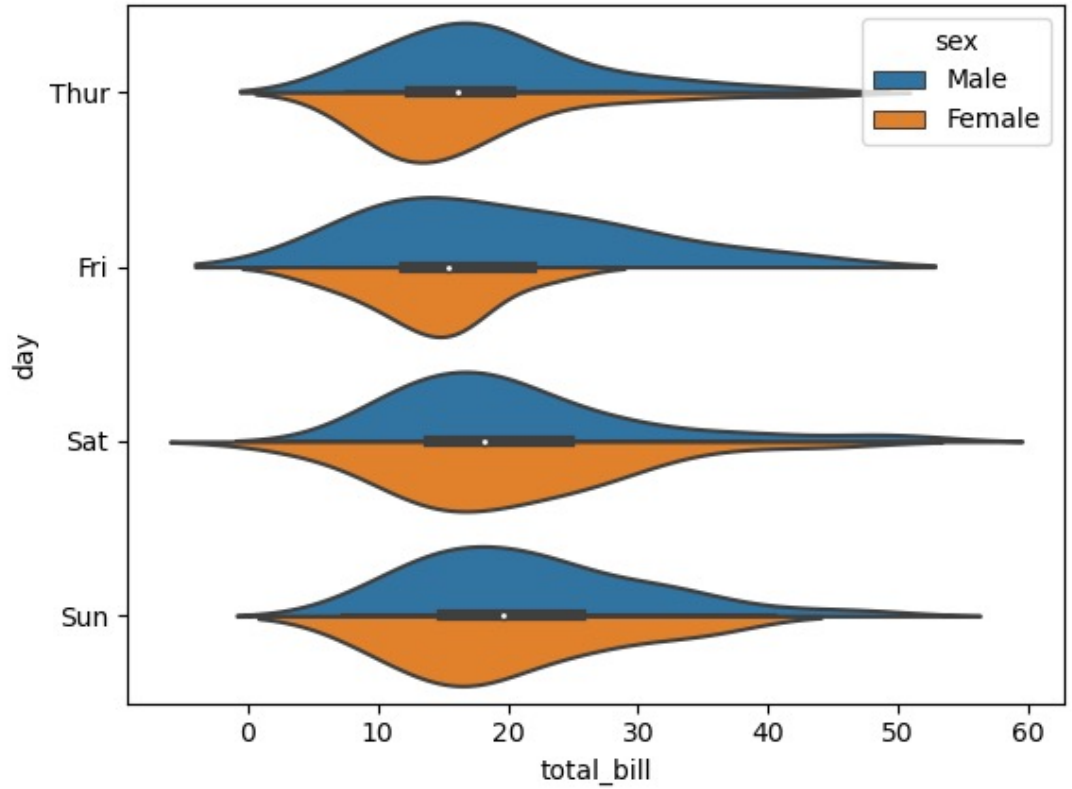
Violin plot using Seaborn (scale)

```
sns.violinplot(x='day', y='total_bill',
data = var, hue='sex', split=True,
scale="width")


plt.show()


#count, width, area parameters can
come inside scale
```



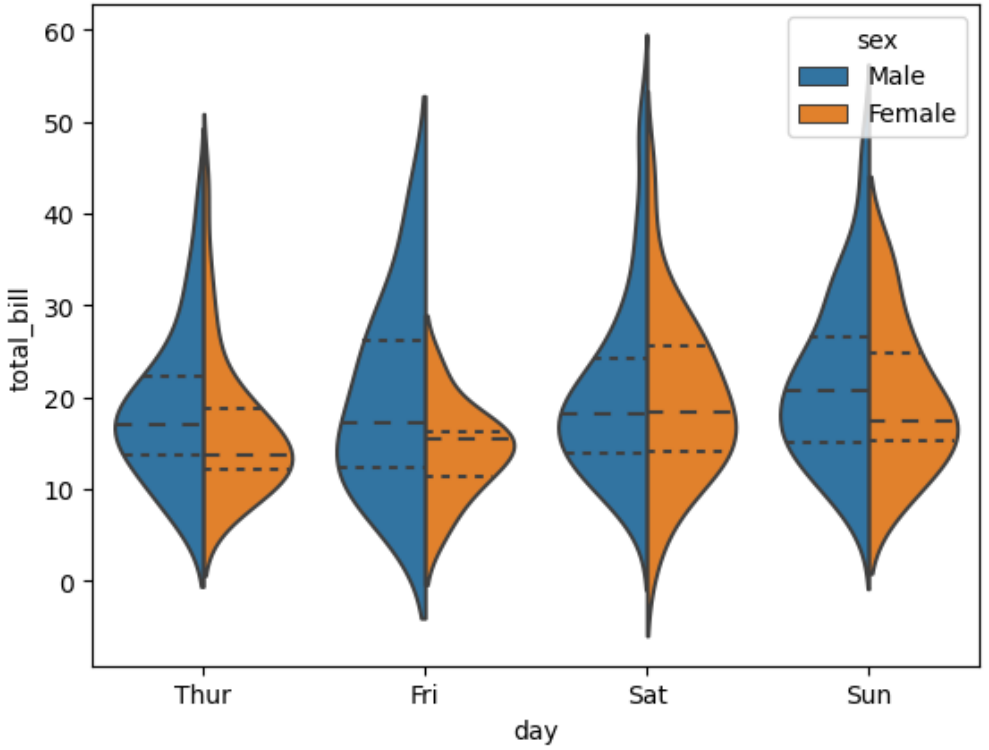
Violin plot using Seaborn (horixzontal)

```
#horizontal plot


sns.violinplot(x='total_bill', y='day',
data = var, hue='sex', split=True,
scale="width")


plt.show()
```



Violin plot using Seaborn (inner)

```
sns.violinplot(x='day', y='total_bill',
data = var, hue='sex', split=True,
scale="width", inner='quart')


plt.show()


#box, quartile, point, stick, none
```

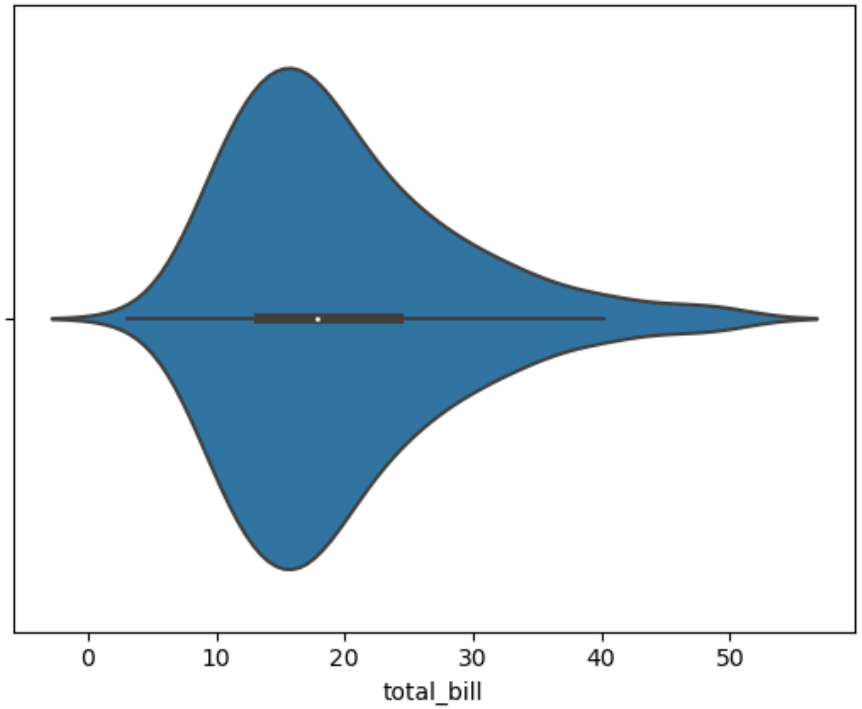


Violin plot using Seaborn (single plot)

```
sns.violinplot(x=var['total_bill'])

plt.show()
```

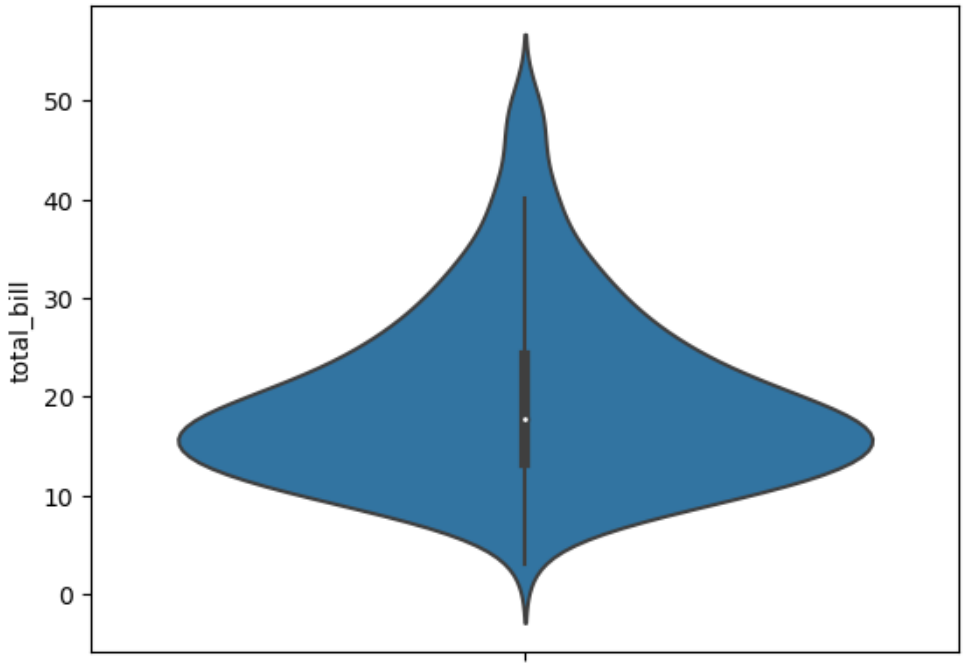
#only possible for columns with numerical values

#please identify which columns are possible



Violin plot using Seaborn (single plot)

```
#single plot vertically

sns.violinplot(y=var['total_bill'])

plt.show()
```



Pairplot using Seaborn

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

var = sns.load_dataset("tips")

var
```

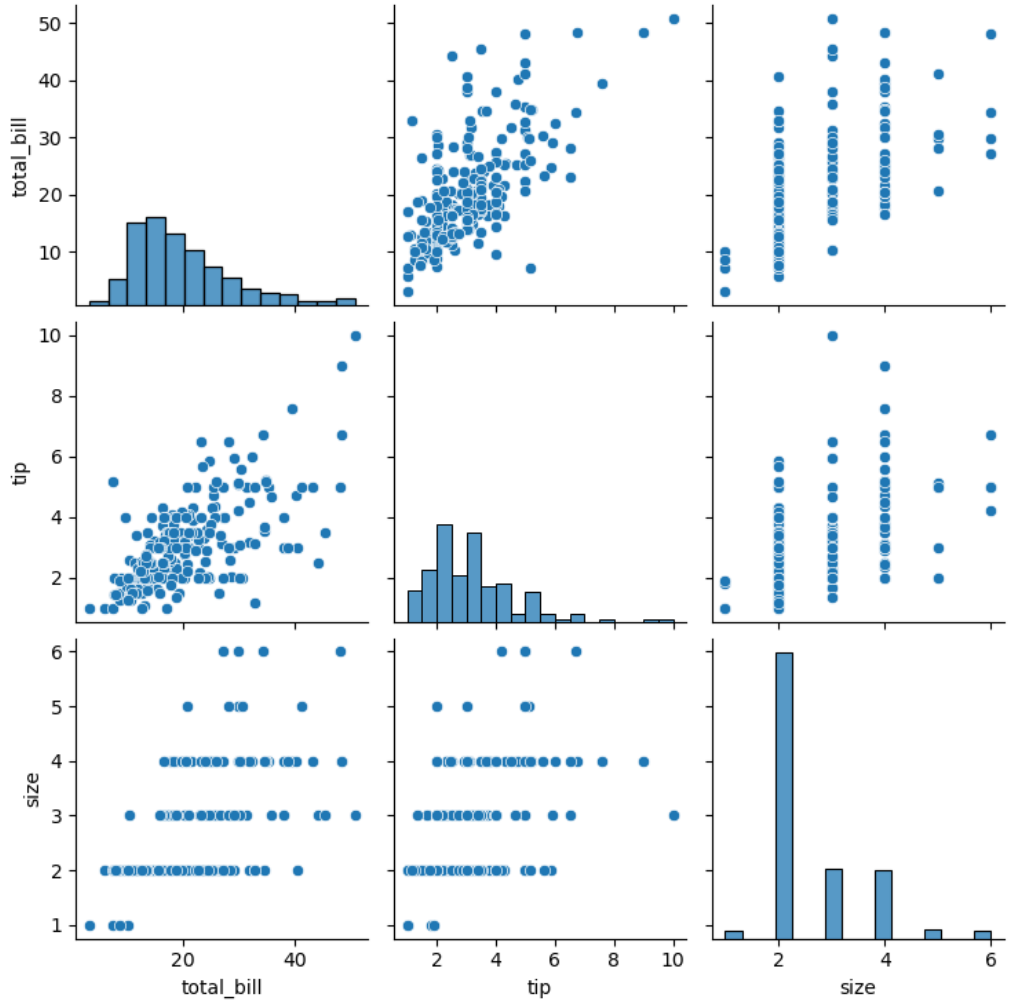
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	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

Pairplot using Seaborn

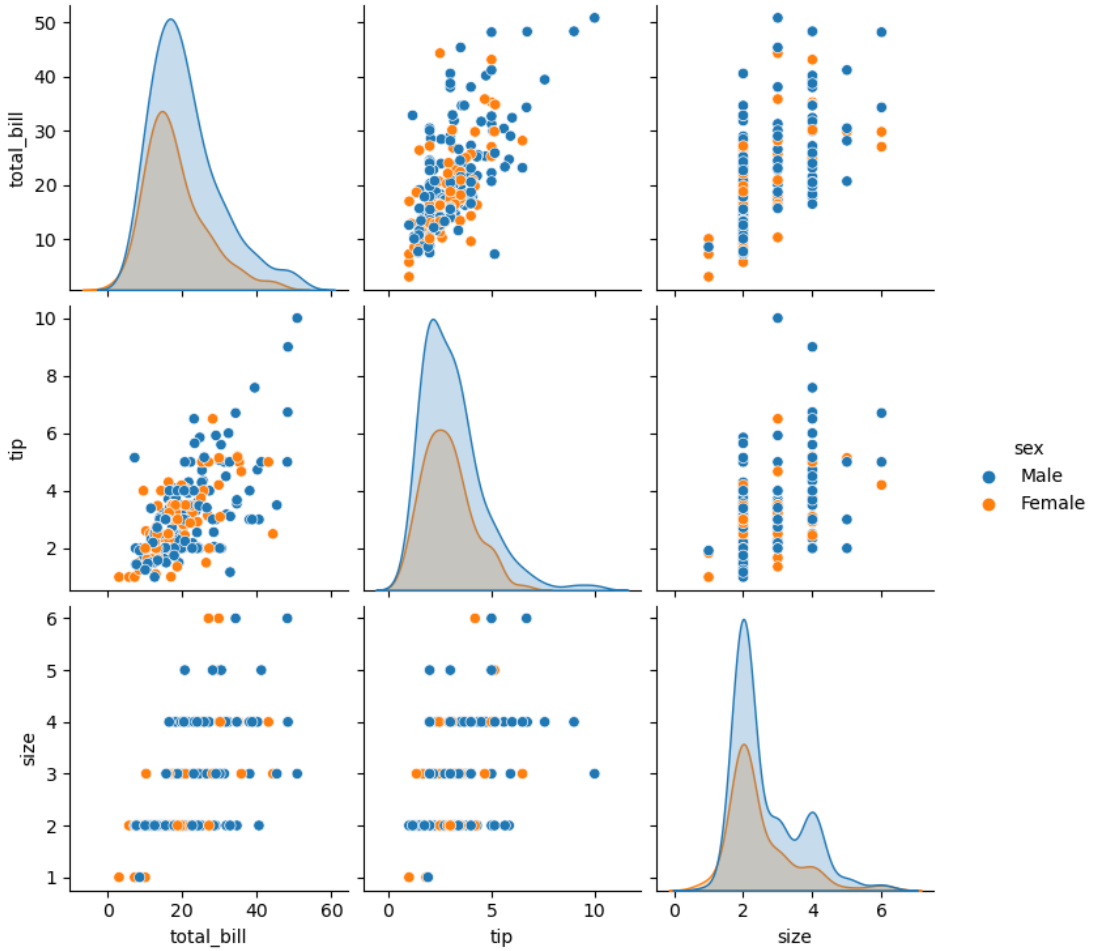
sns.pairplot(var)

plt.show()



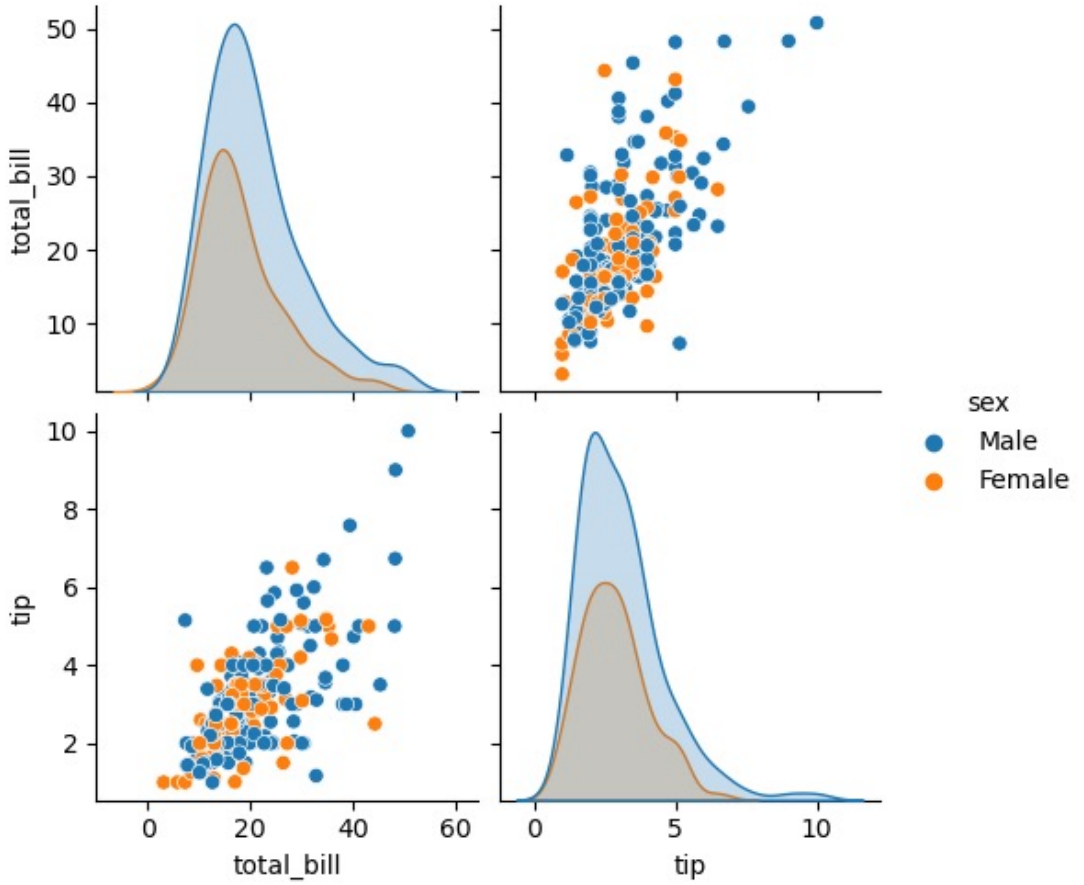
Pairplot using Seaborn (hue)

```
sns.pairplot(var, hue='sex')

plt.show()
```



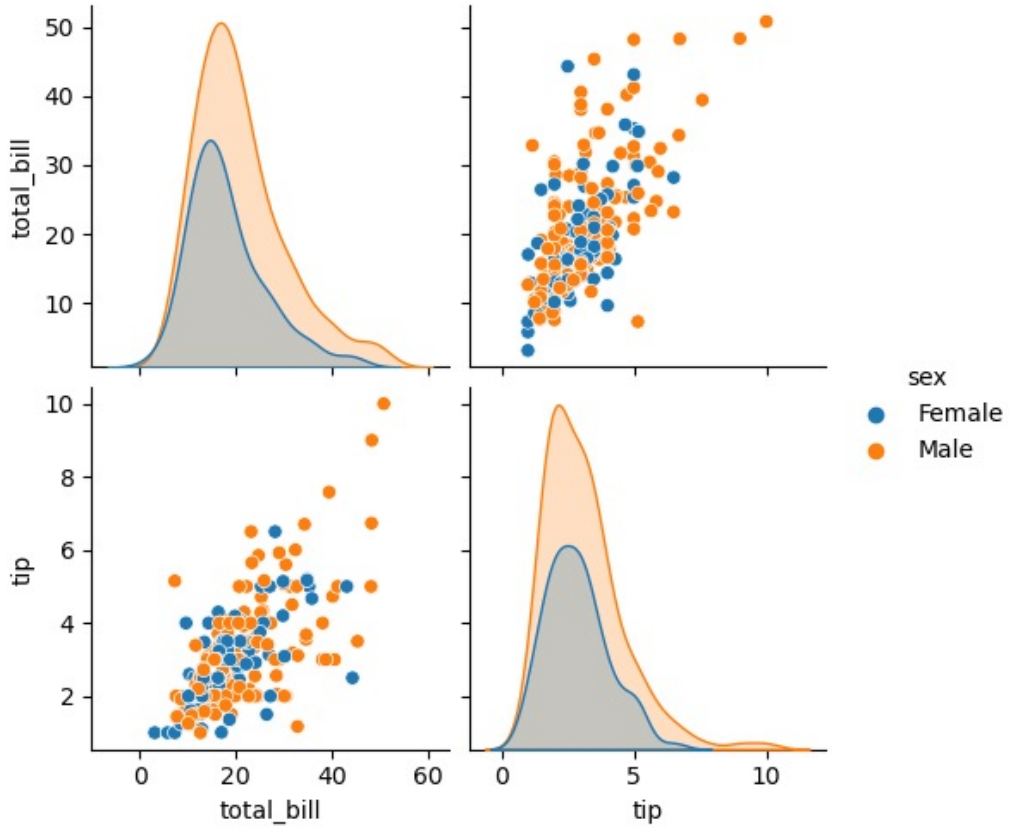
Pairplot using Seaborn (vars)

```
sns.pairplot(var,
vars=["total_bill", "tip"], hue='sex')

plt.show()
```



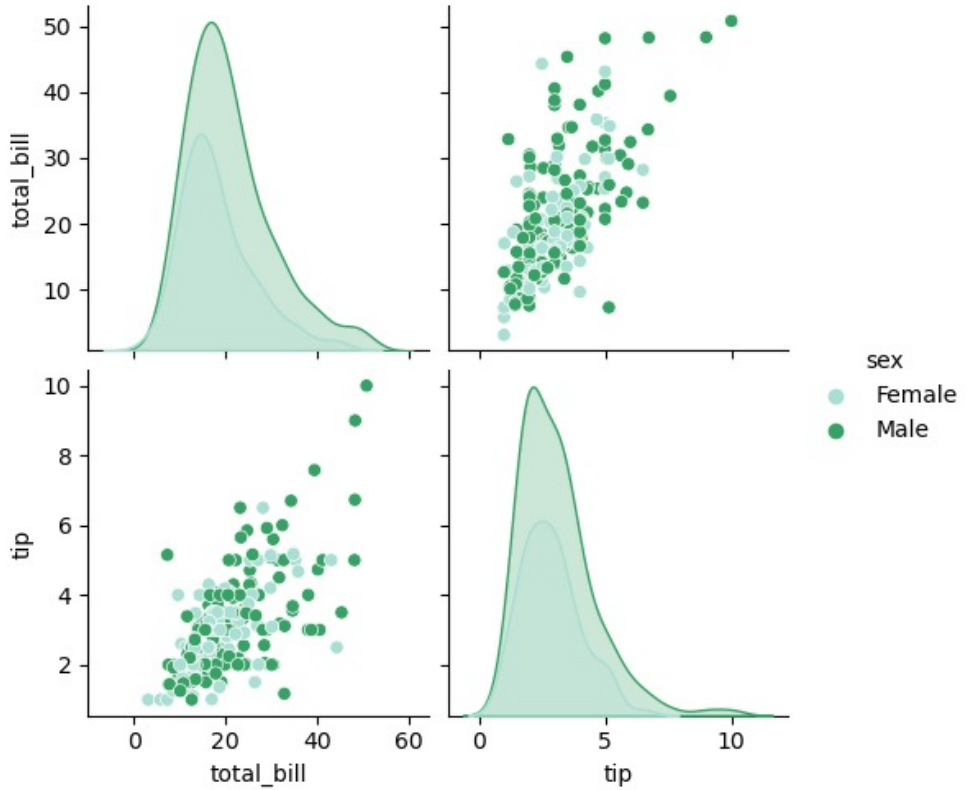
Pairplot using Seaborn (order)

```
sns.pairplot(var,
vars=["total_bill", "tip"], hue='sex’,
hue_order=["Female", "Male"])

plt.show()
```



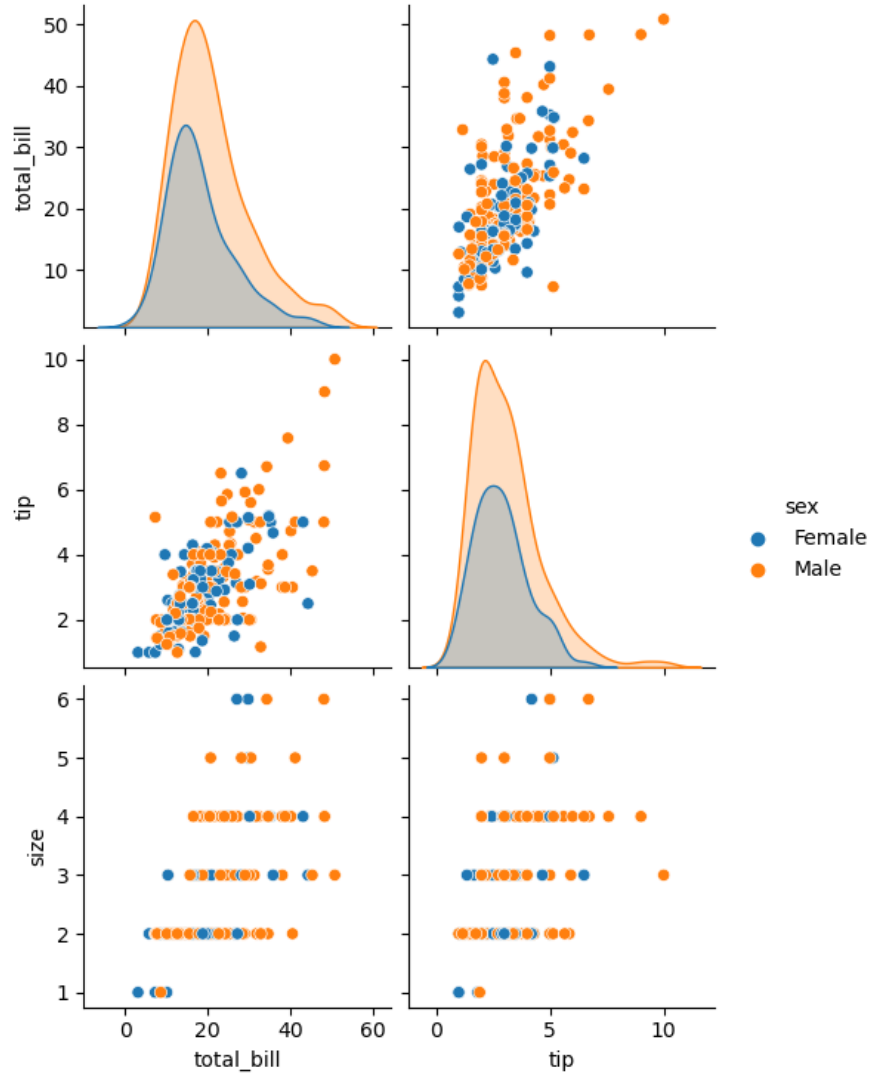
Pairplot using Seaborn (palette)

```
sns.pairplot(var, vars=["total_bill", "tip"],
hue='sex', hue_order=["Female", "Male"],
palette='BuGn')

plt.show()
```



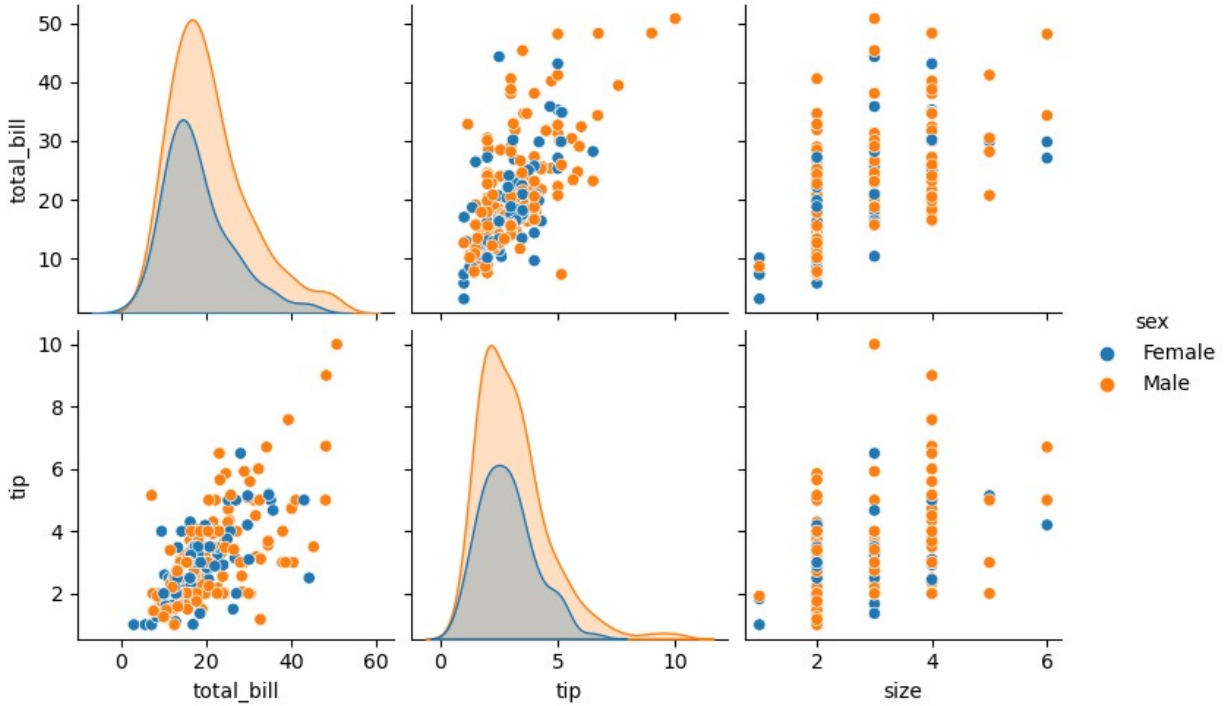
Pairplot using Seaborn (x-axis)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
x_vars=["total_bill", "tip"])

plt.show()
```



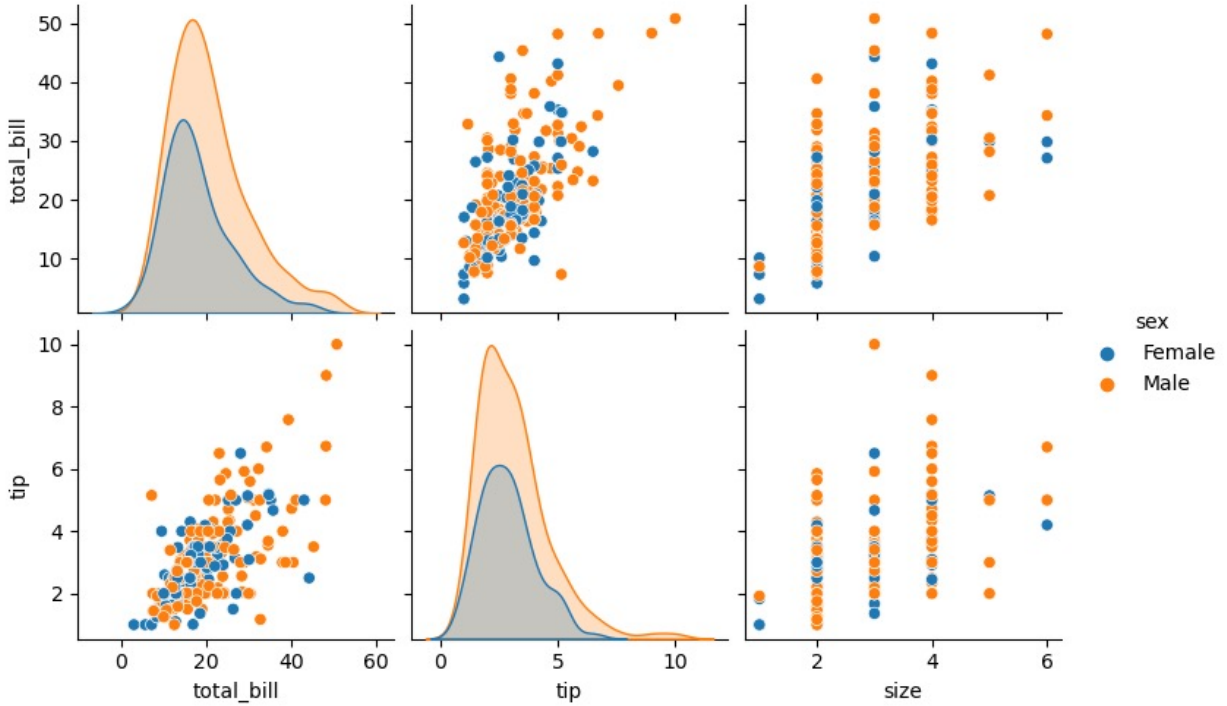
Pairplot using Seaborn (y-axis)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
y_vars=["total_bill", "tip"])

plt.show()
```



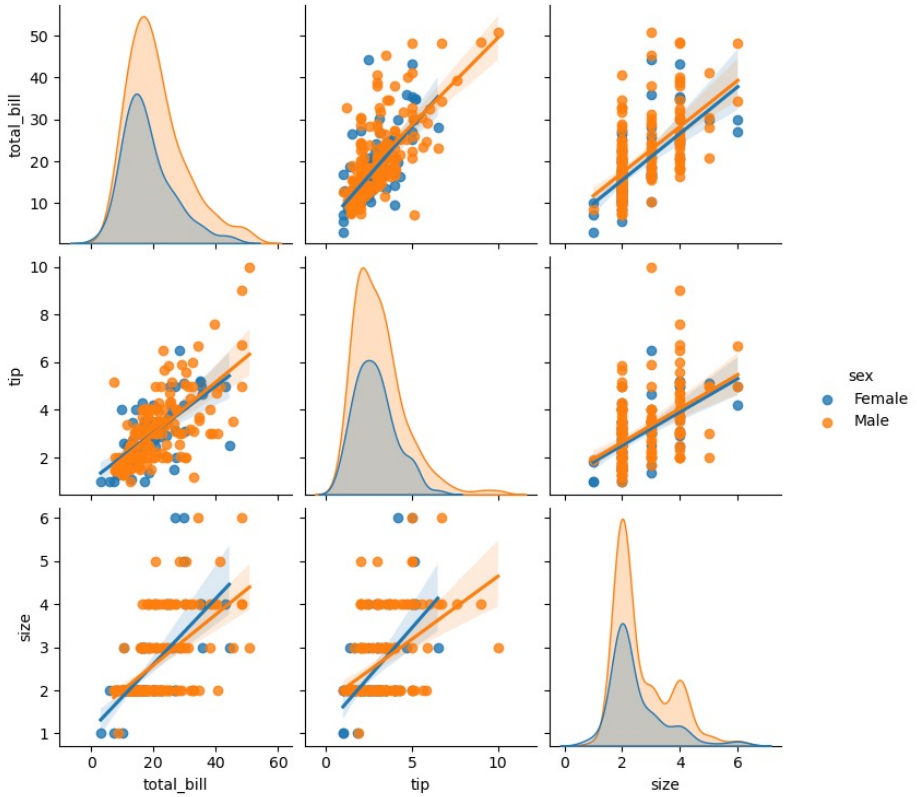
Pairplot using Seaborn (y-axis)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
y_vars=["total_bill", "tip"])

plt.show()
```



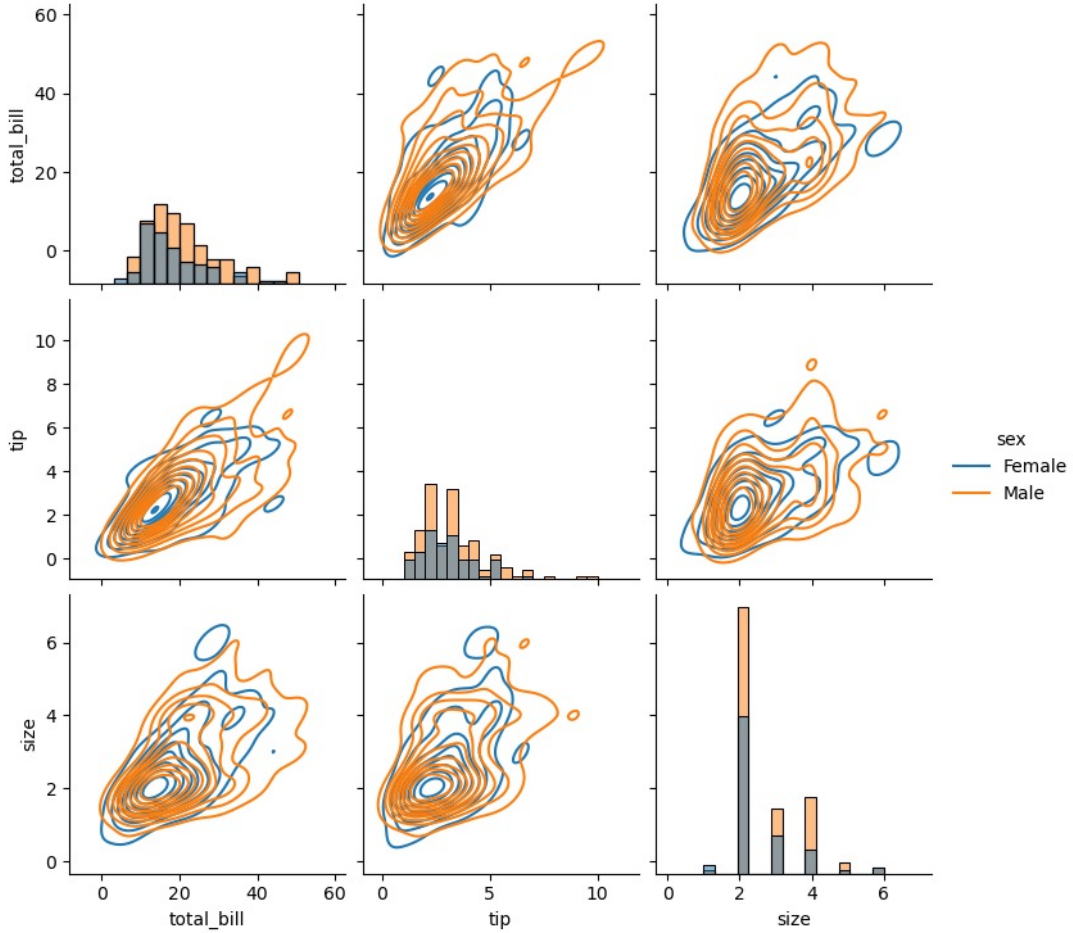
Pairplot using Seaborn (kind)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
kind="reg")

plt.show()

#We can also pass 'scatter', 'kde', 'hist' in kind
```



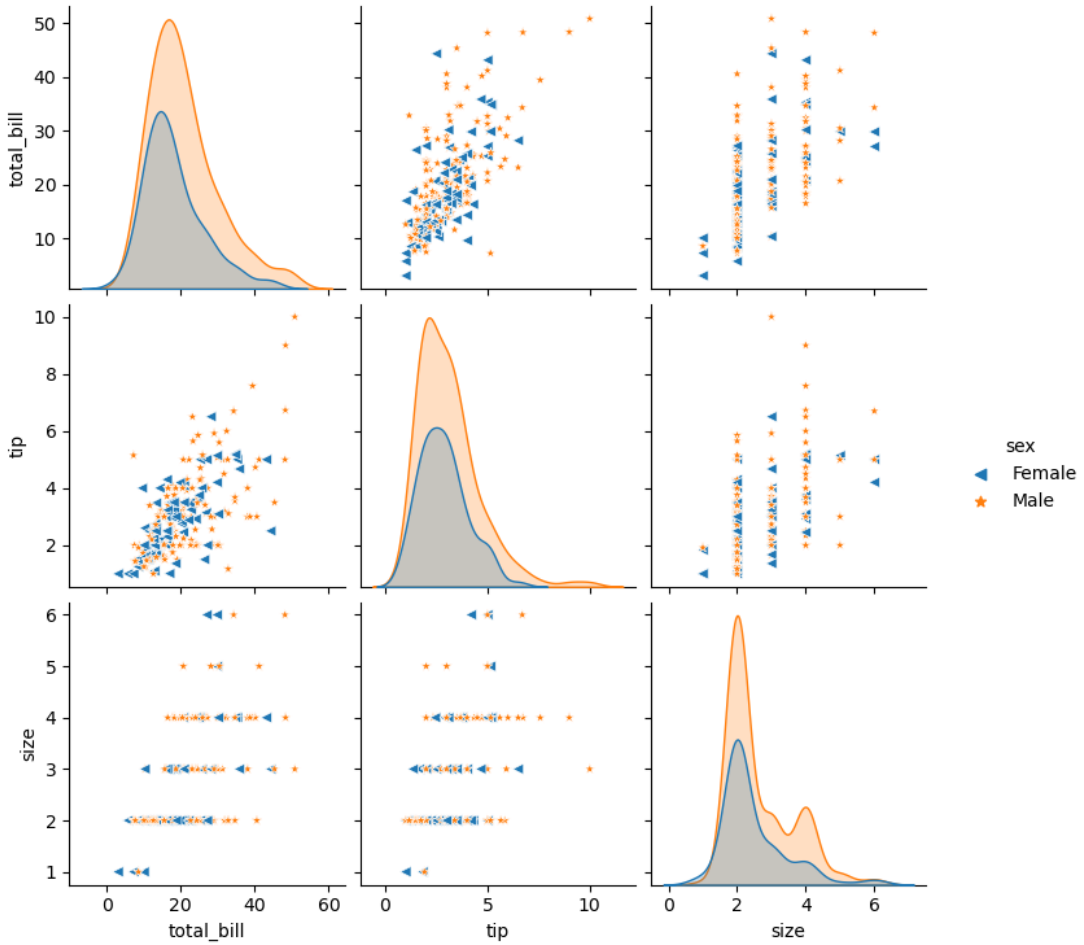
Pairplot using Seaborn (diag_kind)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
kind="kde", diag_kind='hist')

plt.show()

#We can also pass 'scatter', 'kde', 'hist' in kind
```



Pairplot using Seaborn (markers)

```
sns.pairplot(var, hue='sex’,
hue_order=["Female", "Male"],
markers=['*', '<'])

plt.show()
```



Strip using Seaborn (markers)

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

var = sns.load_dataset("tips")

var
```

	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

Strip using Seaborn (markers)

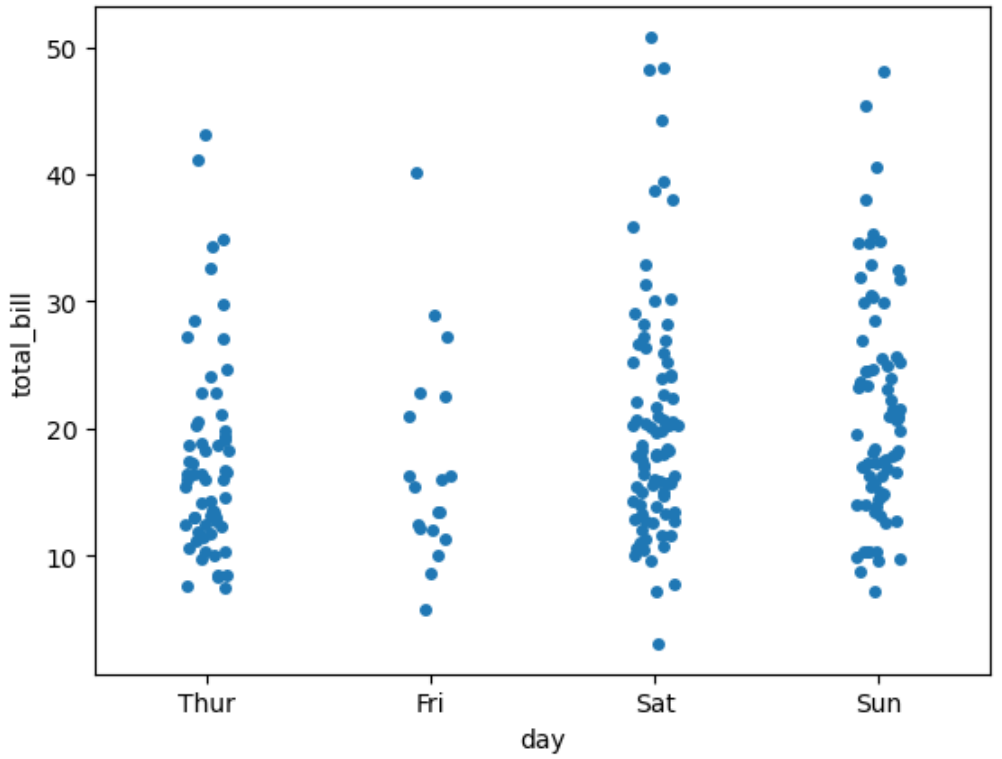
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

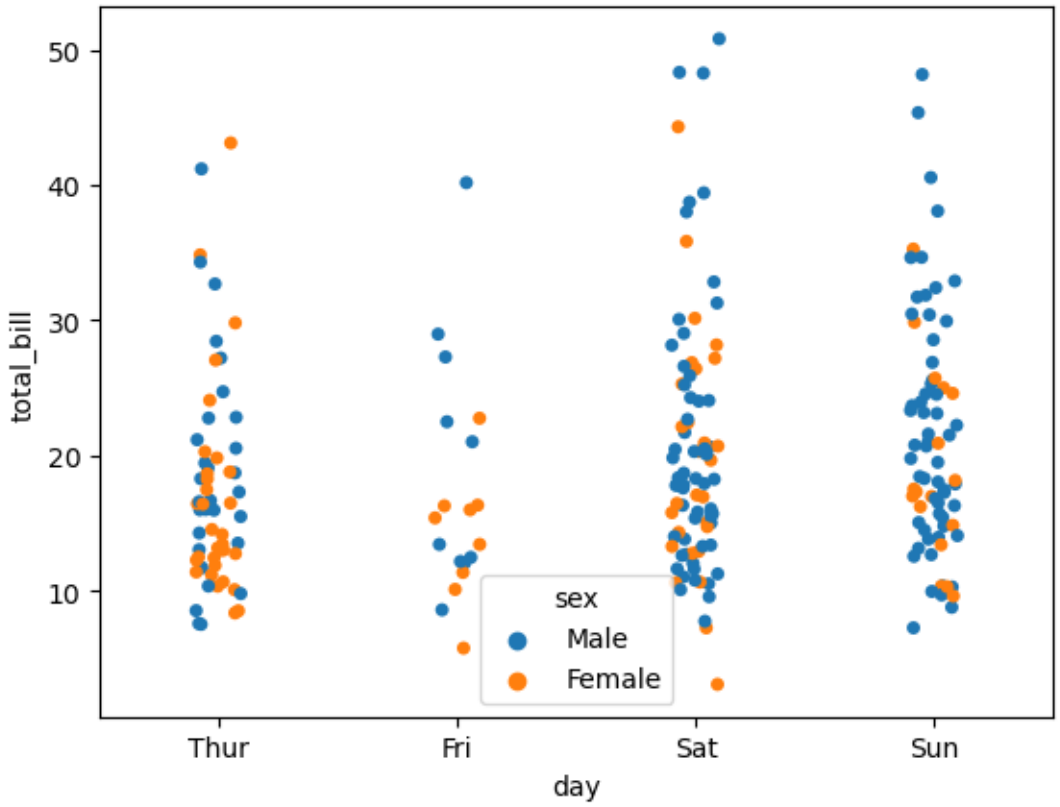
Strip using Seaborn

```
sns.stripplot(x='day', y='total_bill', data=var)

plt.show()
```



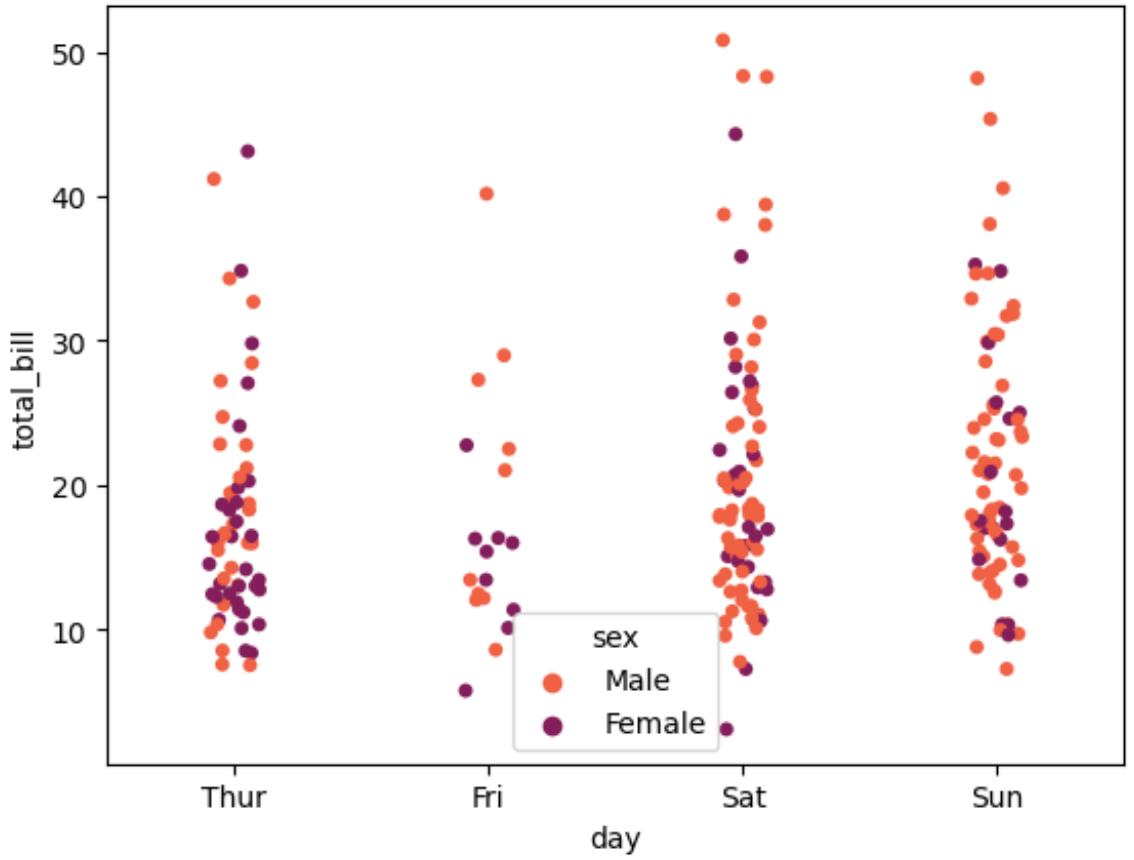
Strip using Seaborn (hue)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex')

plt.show()
```



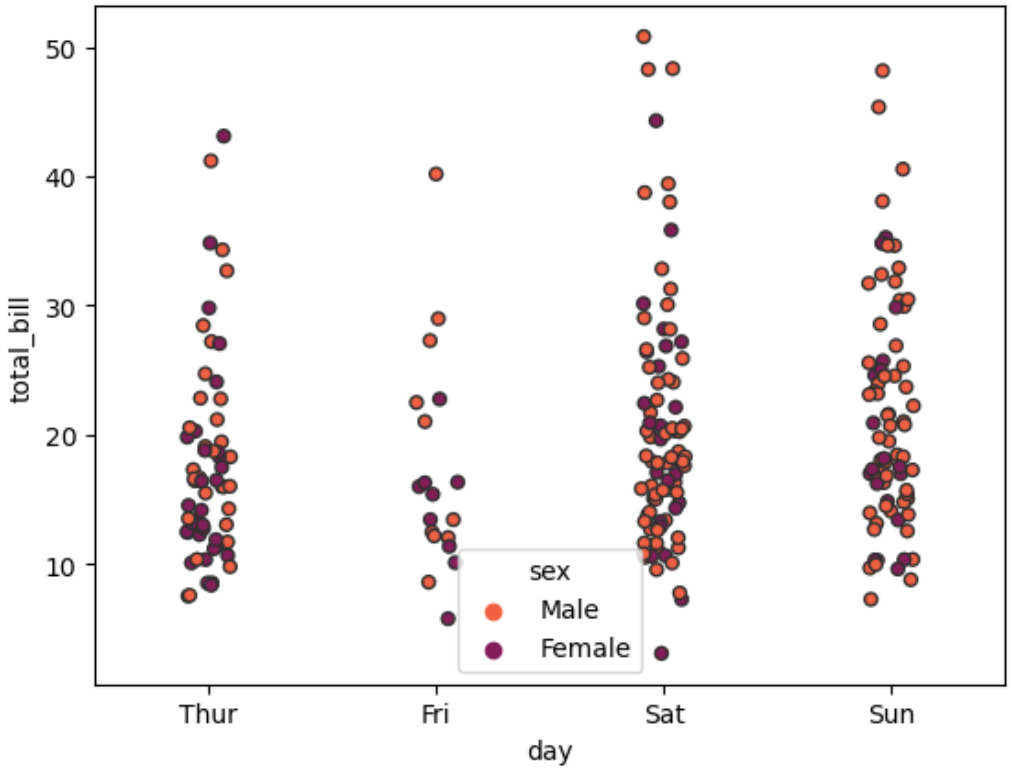
Strip using Seaborn (palette)

```
sns.stripplot(x='day', y='total_bill’,
data=var, hue='sex', palette='rocket_r')

plt.show()
```



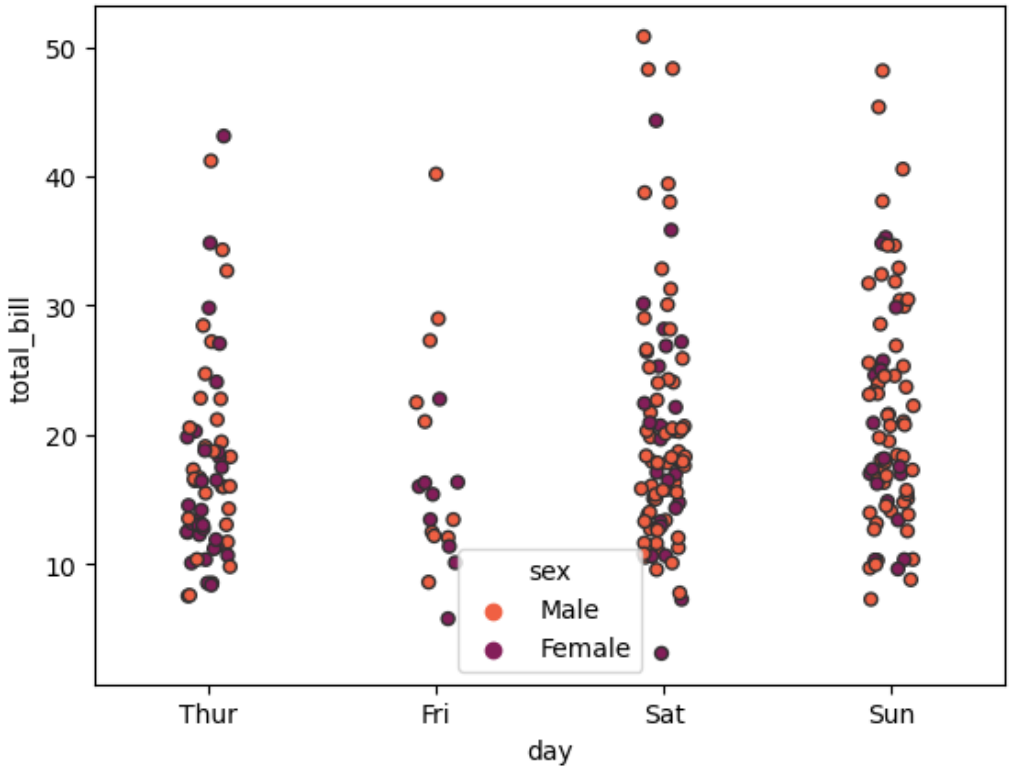
Strip using Seaborn (linewidth)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1)

plt.show()
```



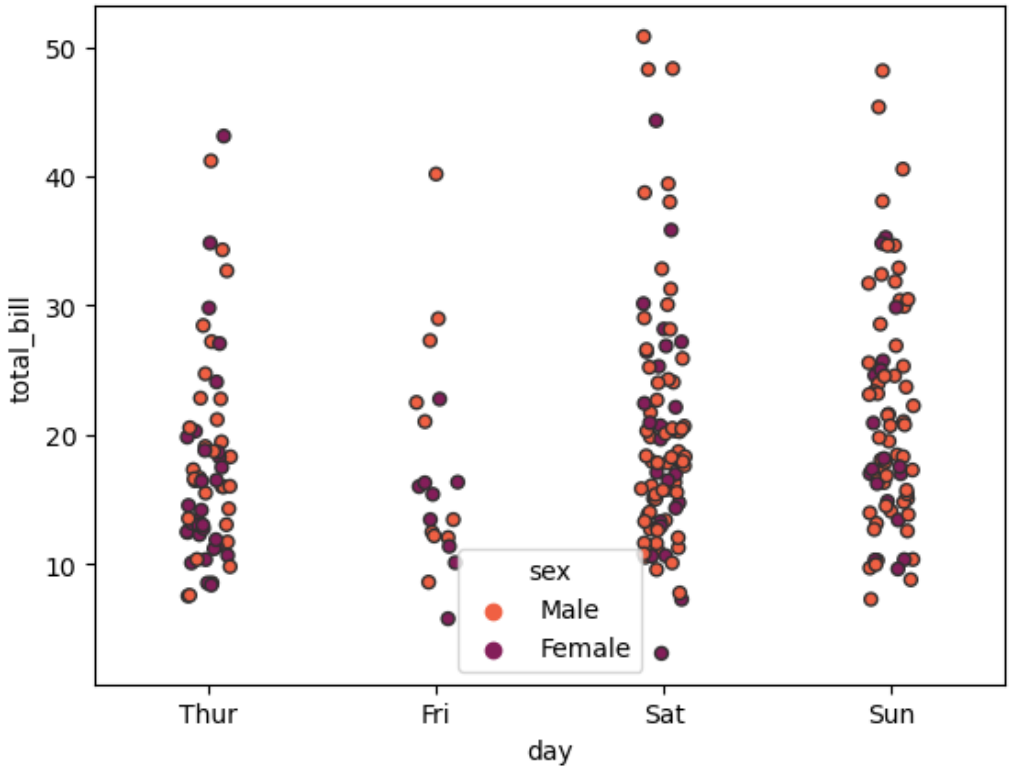
Strip using Seaborn (edgecolor)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1,
edgecolor='r')

plt.show()
```



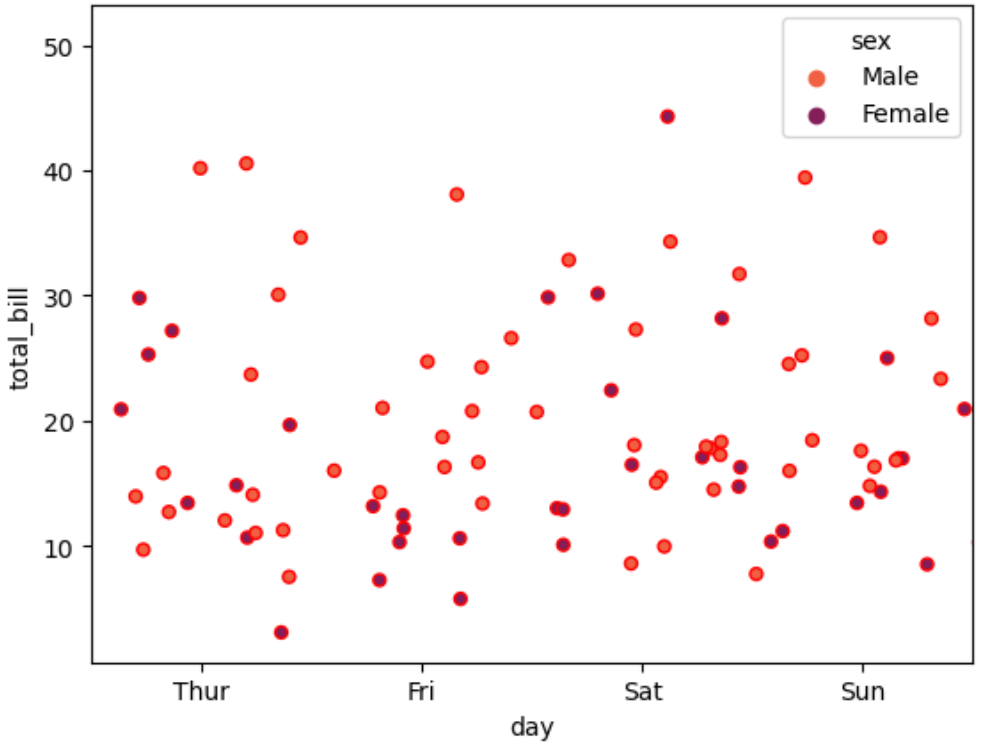
Strip using Seaborn (edgecolor)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1,
edgecolor='r')

plt.show()
```



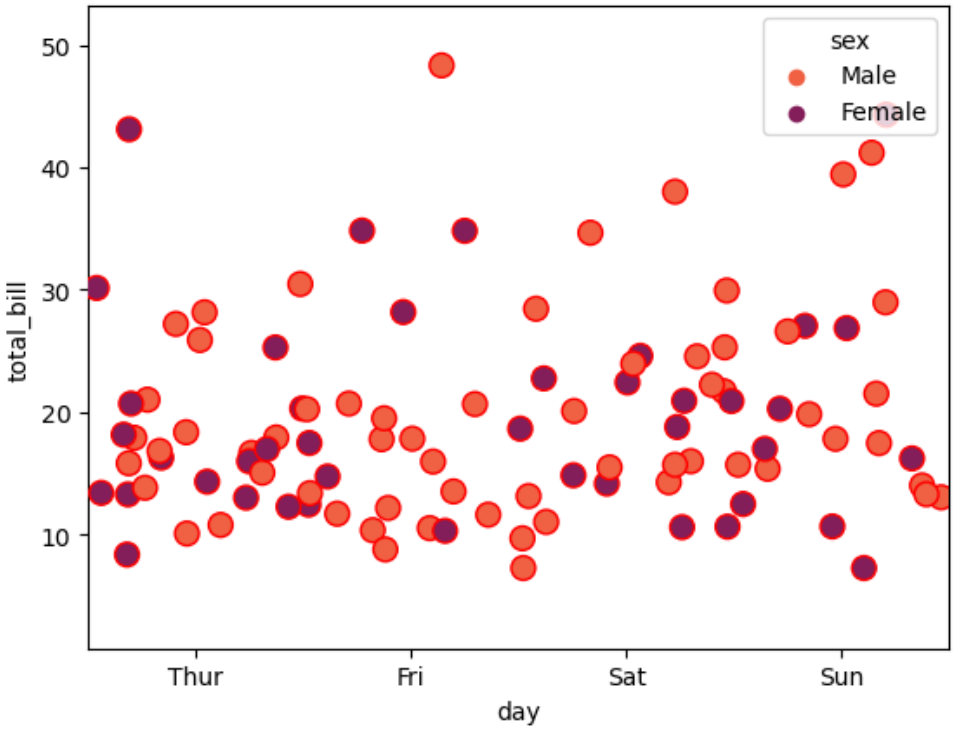
Strip using Seaborn (jitter)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1,
edgecolor='r', jitter=5)

plt.show()
```



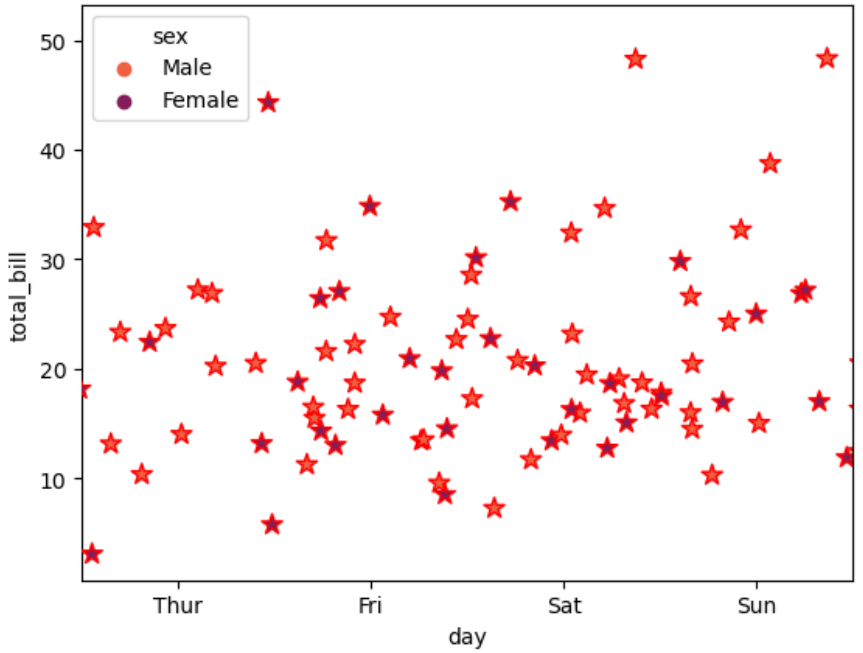
Strip using Seaborn (size)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1,
edgecolor='r', jitter=5, size=10)

plt.show()
```



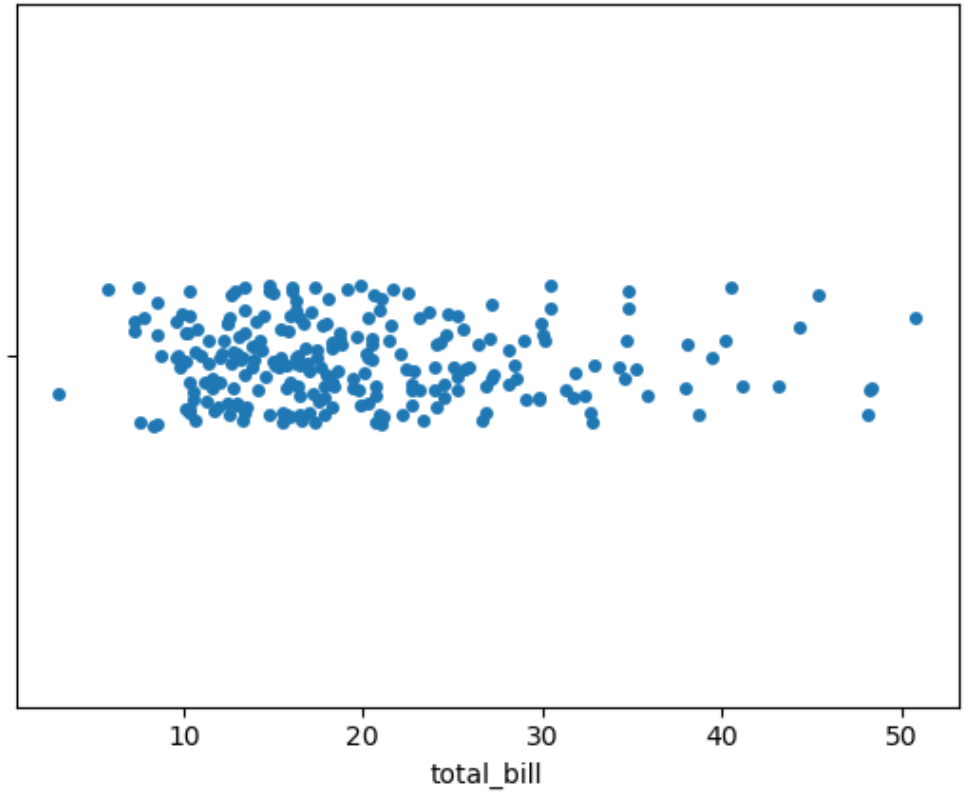
Strip using Seaborn (marker)

```
sns.stripplot(x='day', y='total_bill', data=var,
hue='sex', palette='rocket_r', linewidth=1,
edgecolor='r', jitter=5, size=10, marker='*')

plt.show()
```



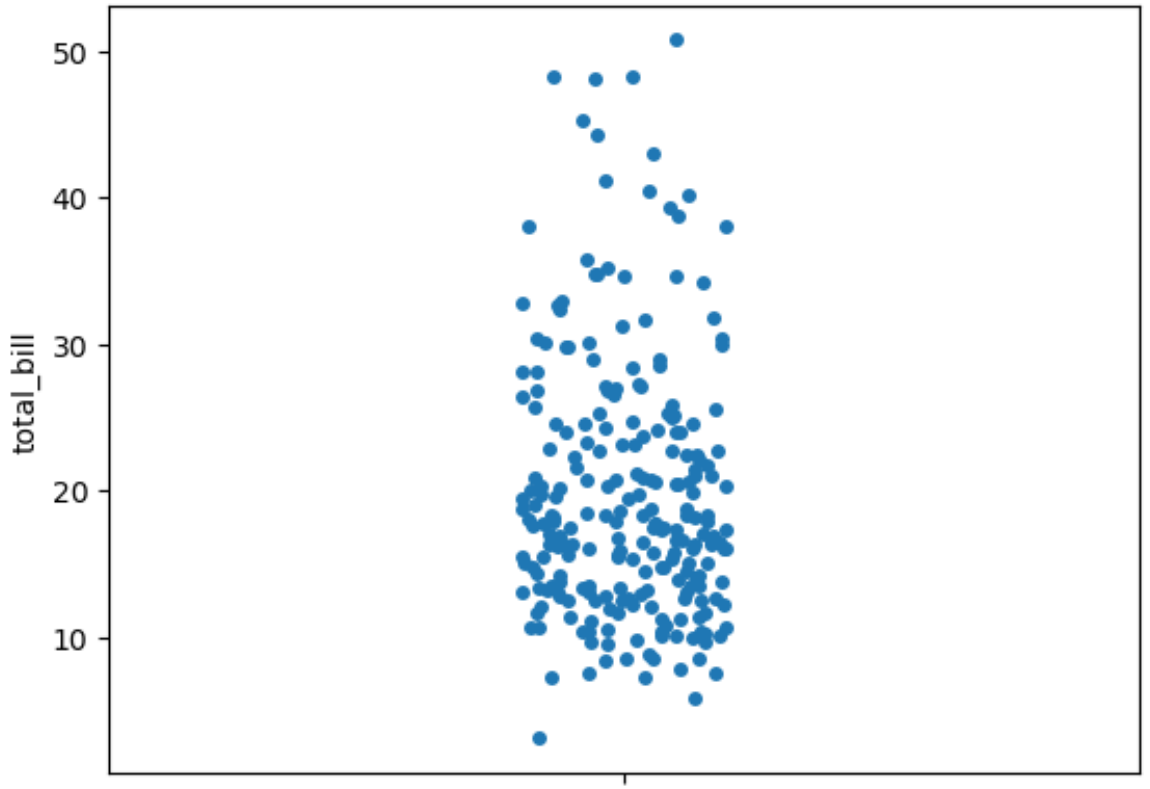
Strip using Seaborn (single plot)

```
sns.stripplot(x=var['total_bill'])

plt.show()
```



Strip using Seaborn (single plot)

```
sns.stripplot(y=var['total_bill'])

plt.show()
```



Boxplot using Seaborn

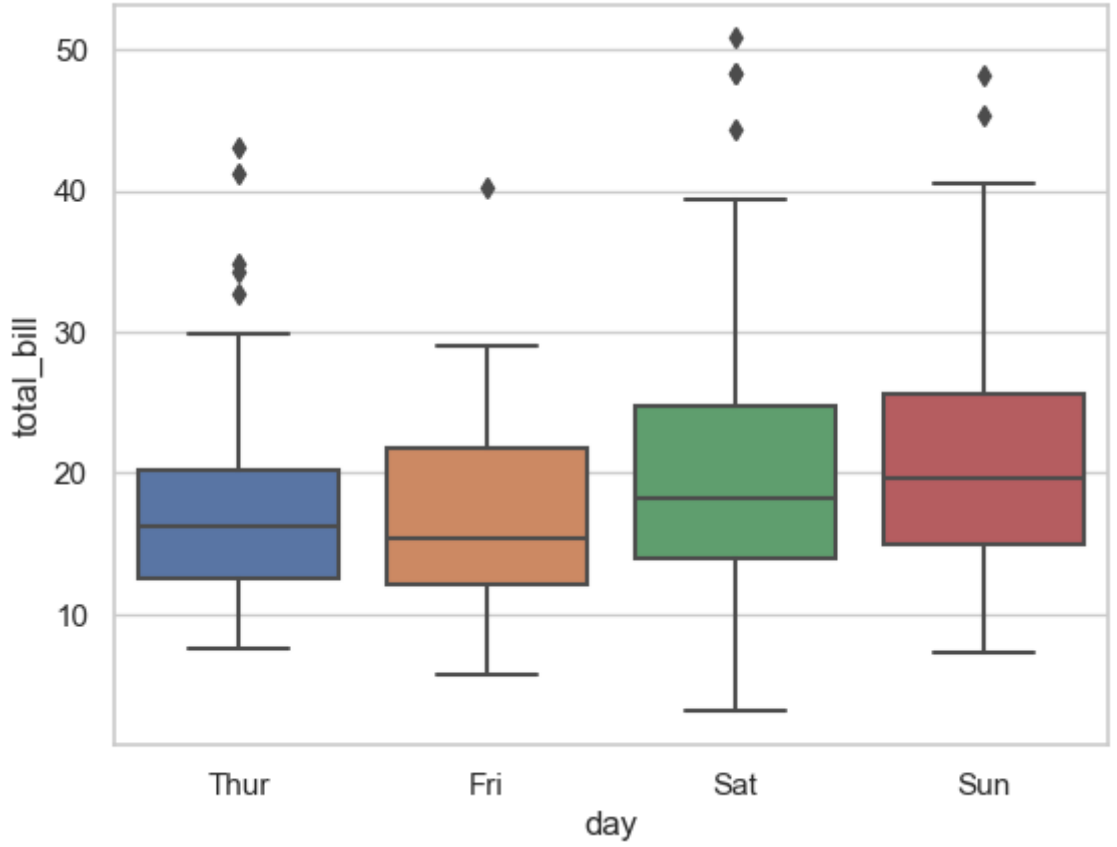
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

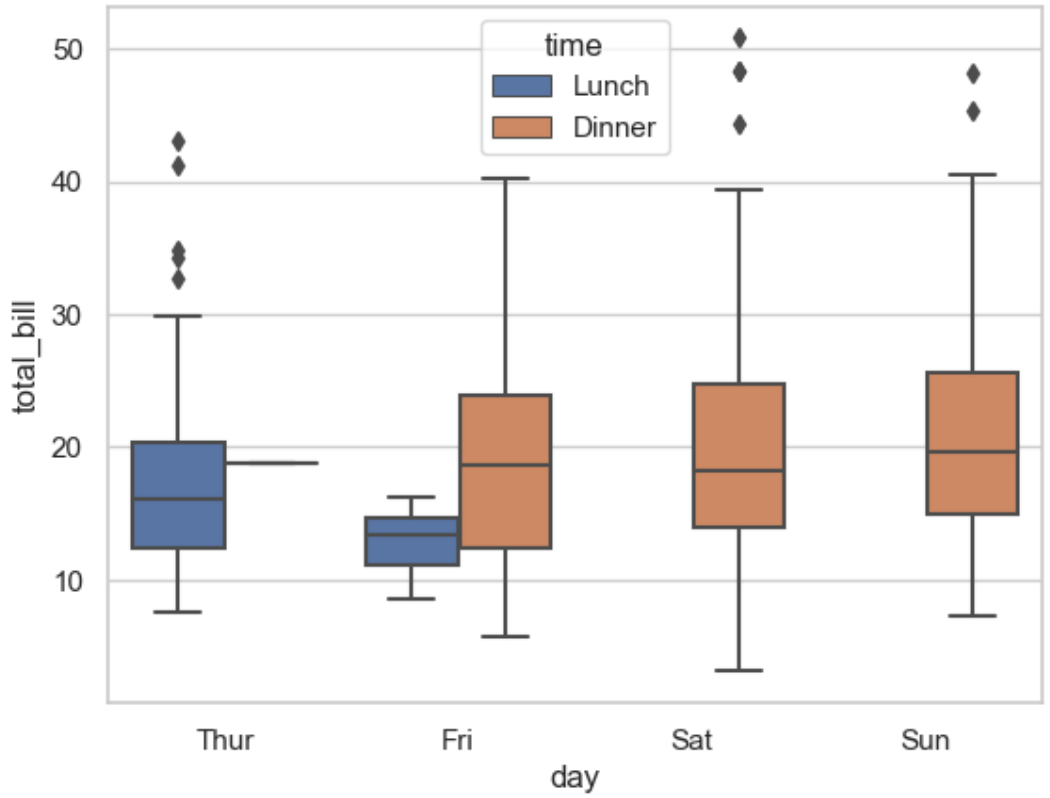
Boxplot using Seaborn (whitegrid)

```
sns.set(style="whitegrid")

sns.boxplot(x='day', y='total_bill',
data=var)

plt.show()
```



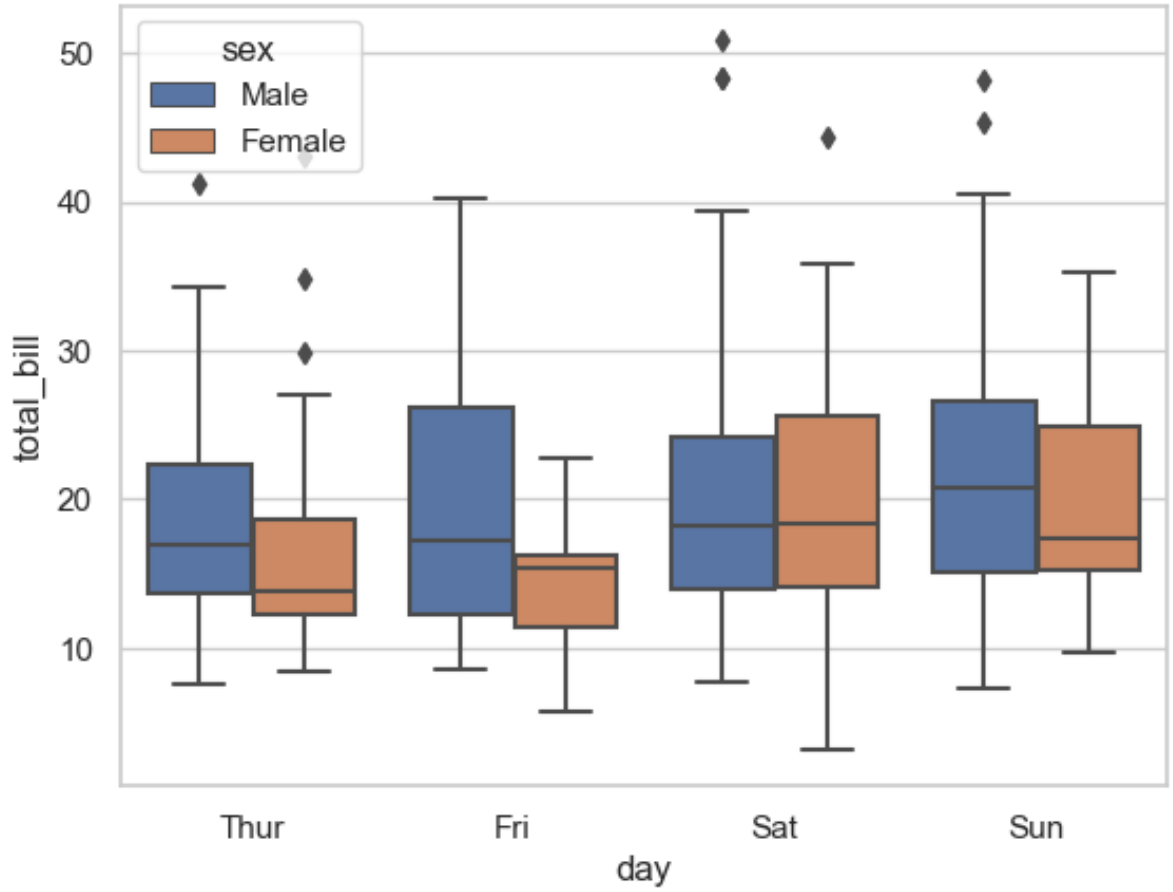
Boxplot using Seaborn (hue='time')

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='time')

plt.show()
```



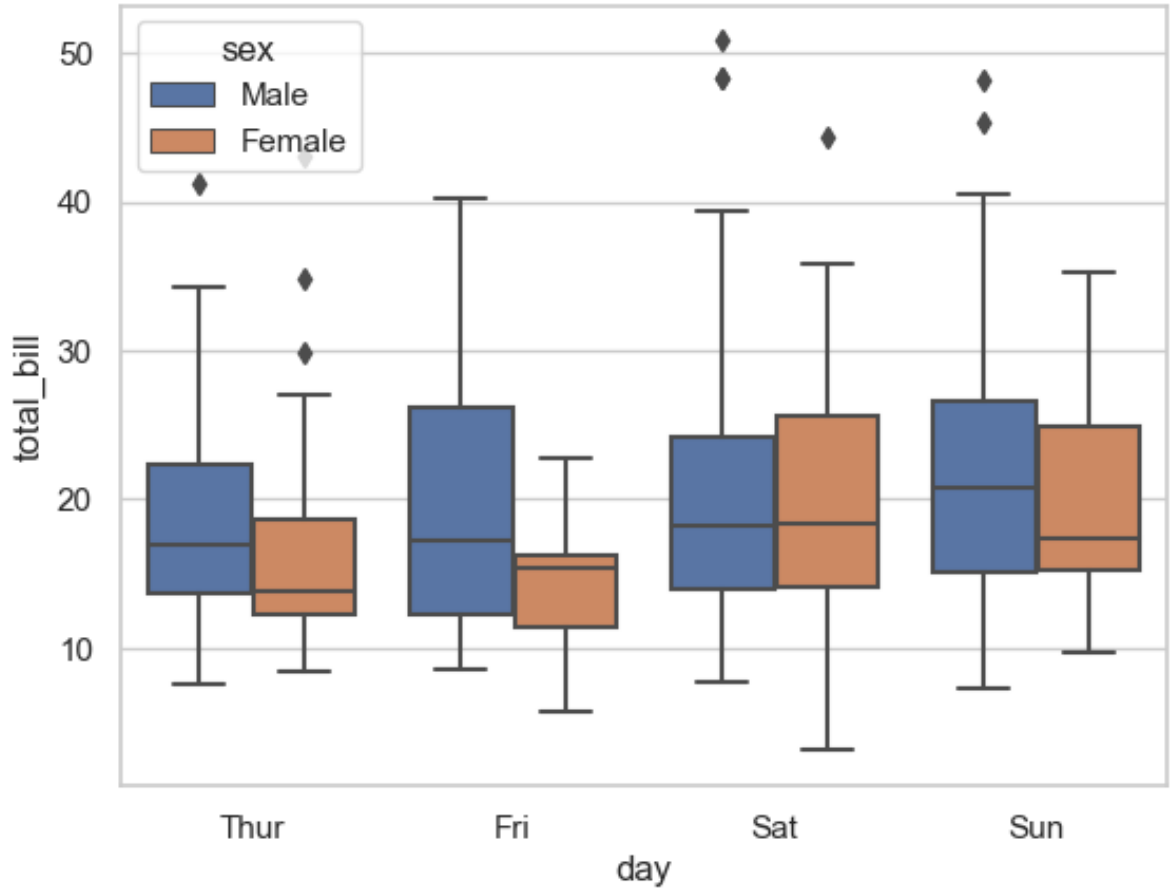
Boxplot using Seaborn (hue='sex')

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex')

plt.show()
```



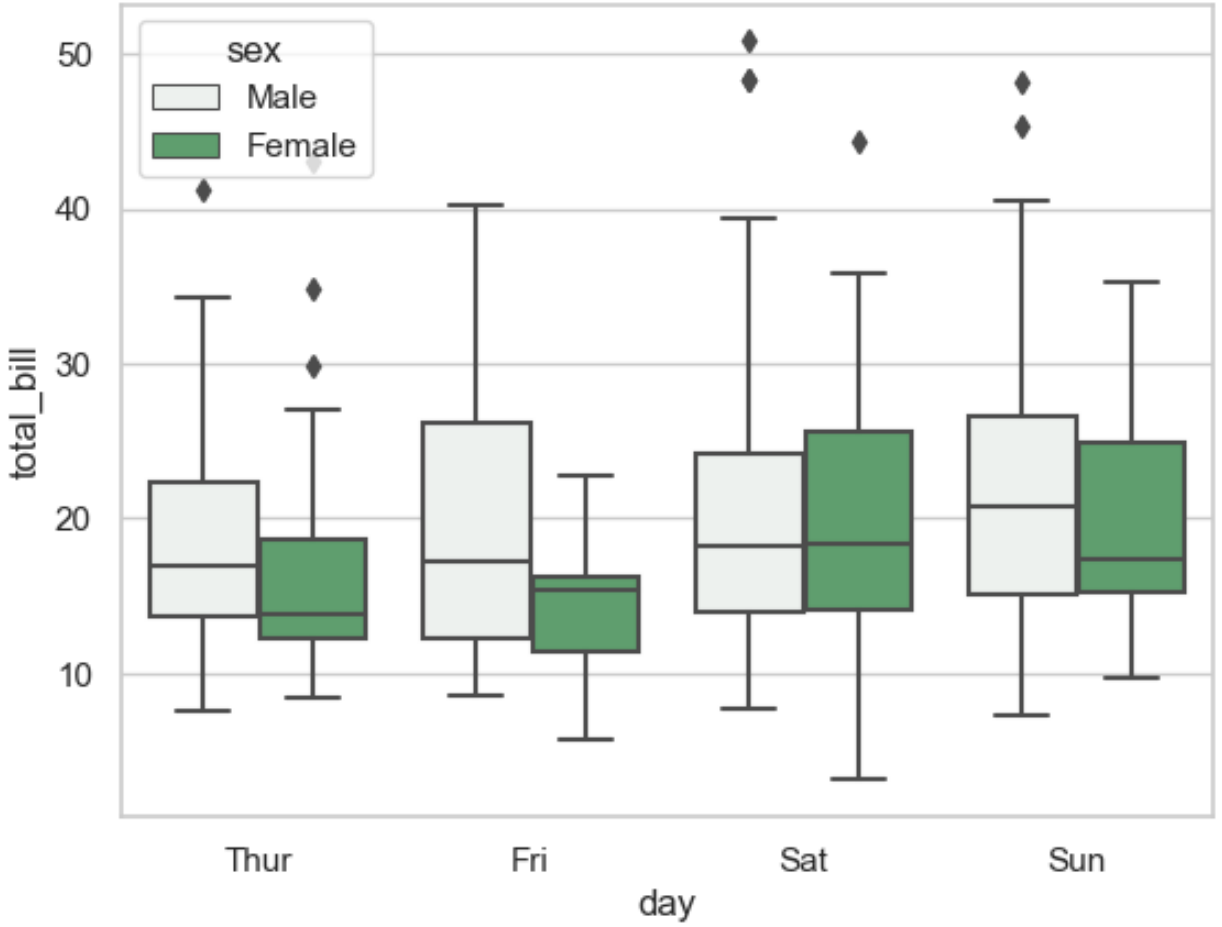
Boxplot using Seaborn (hue='sex')

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex')

plt.show()
```



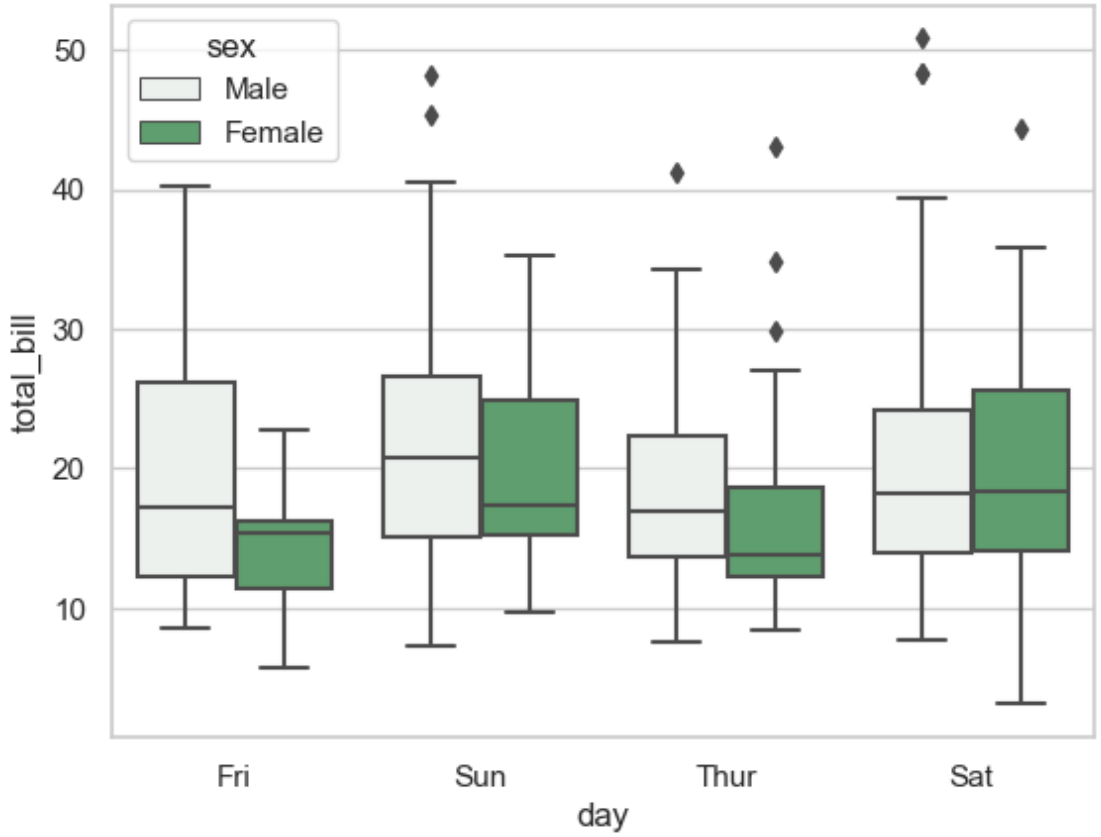
Boxplot using Seaborn (color)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='g')

plt.show()
```



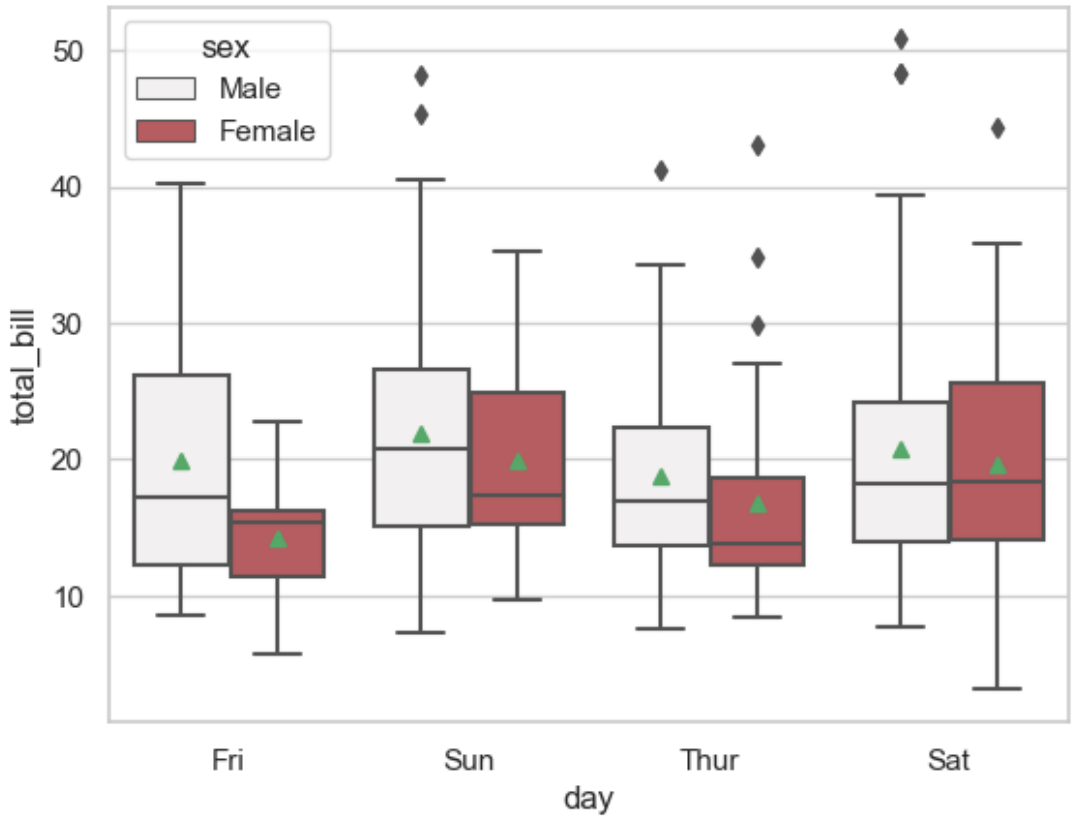
Boxplot using Seaborn (order)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='g',
order=['Fri', 'Sun', 'Thur', 'Sat'])

plt.show()
```



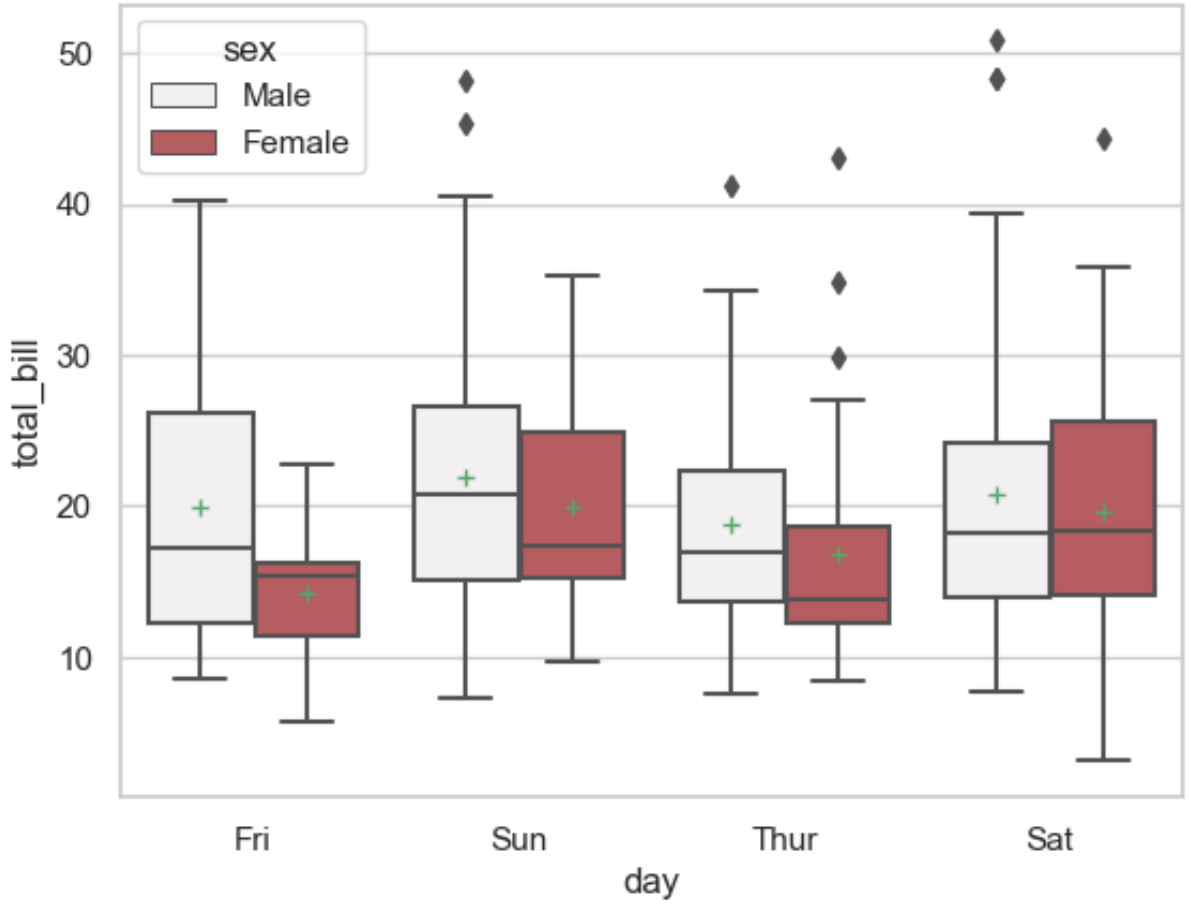
Boxplot using Seaborn (showmeans)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='r',
order=['Fri', 'Sun', 'Thur', 'Sat'],
 showmeans=True)

plt.show()
```



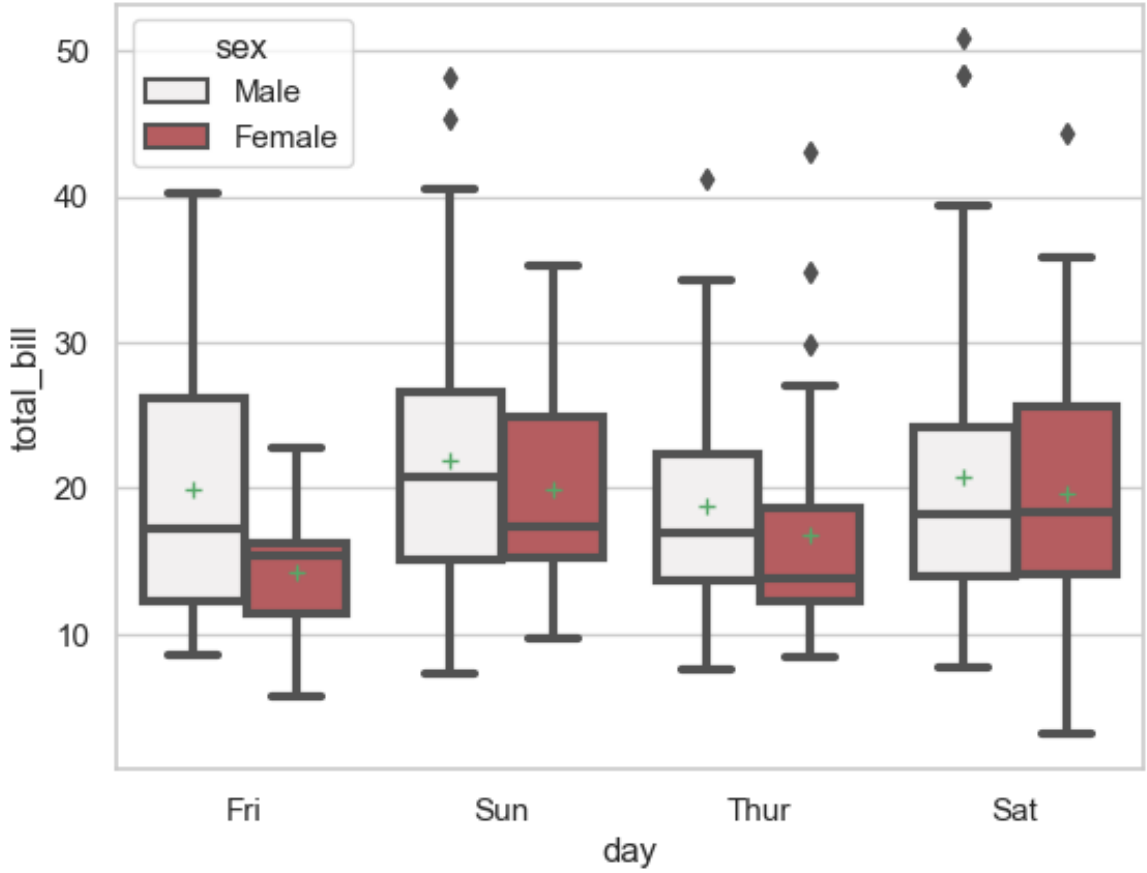
Boxplot using Seaborn (marker)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='r',
order=['Fri', 'Sun', 'Thur', 'Sat'],

        showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'})


plt.show()
```



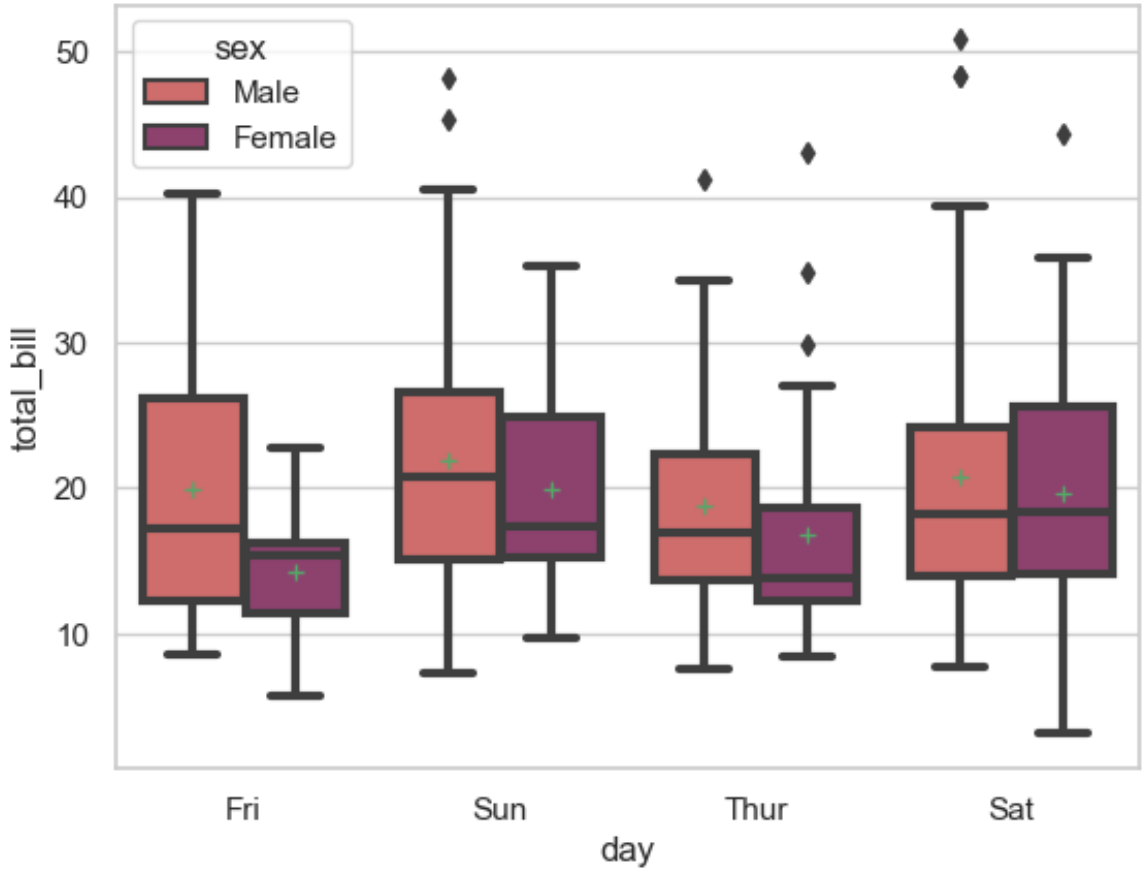
Boxplot using Seaborn (linewidth)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='r',
order=['Fri', 'Sun', 'Thur', 'Sat'],

        showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'}, linewidth=3)


plt.show()
```



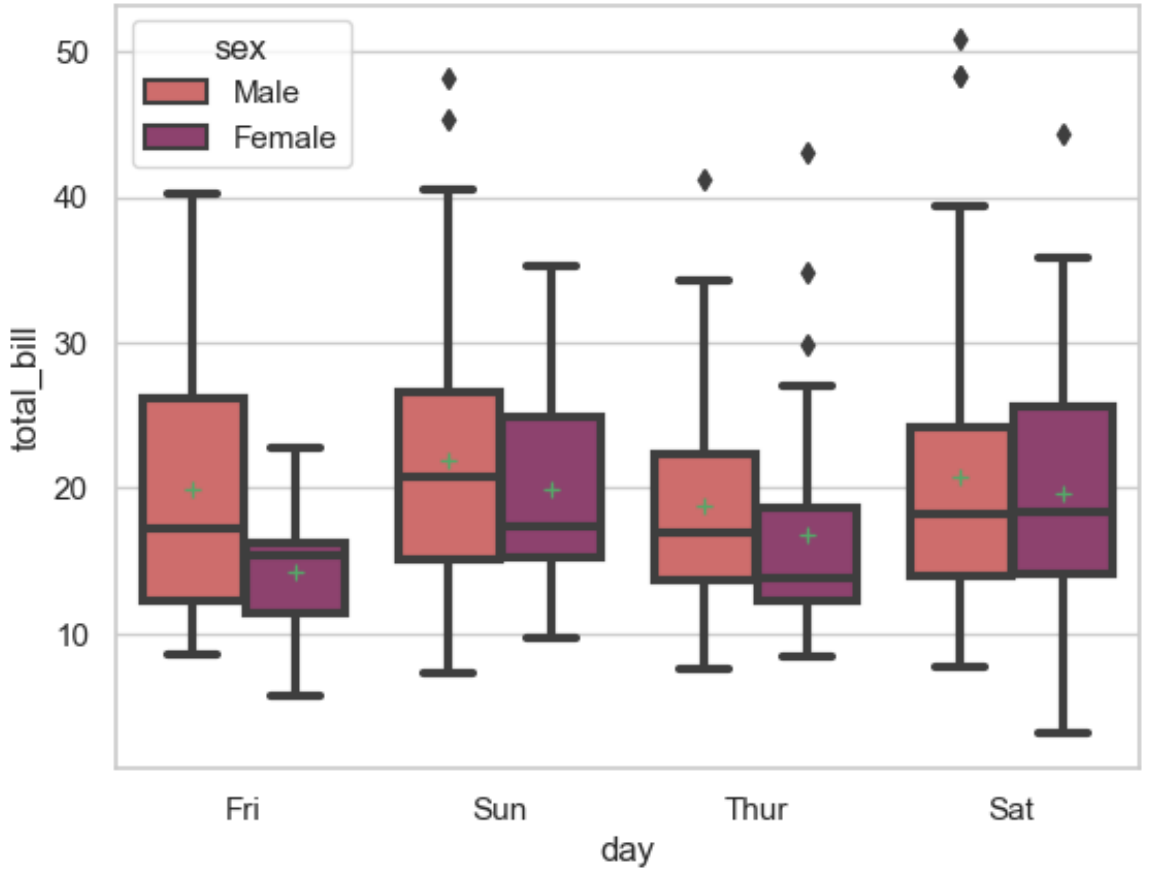
Boxplot using Seaborn (palette)

```
sns.boxplot(x='day', y='total_bill', data=var,
hue='sex', color='r', order=['Fri', 'Sun',
'Thur', 'Sat'],
        showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'}, linewidth=3,
        palette='flare')


#try plasma


plt.show()
```



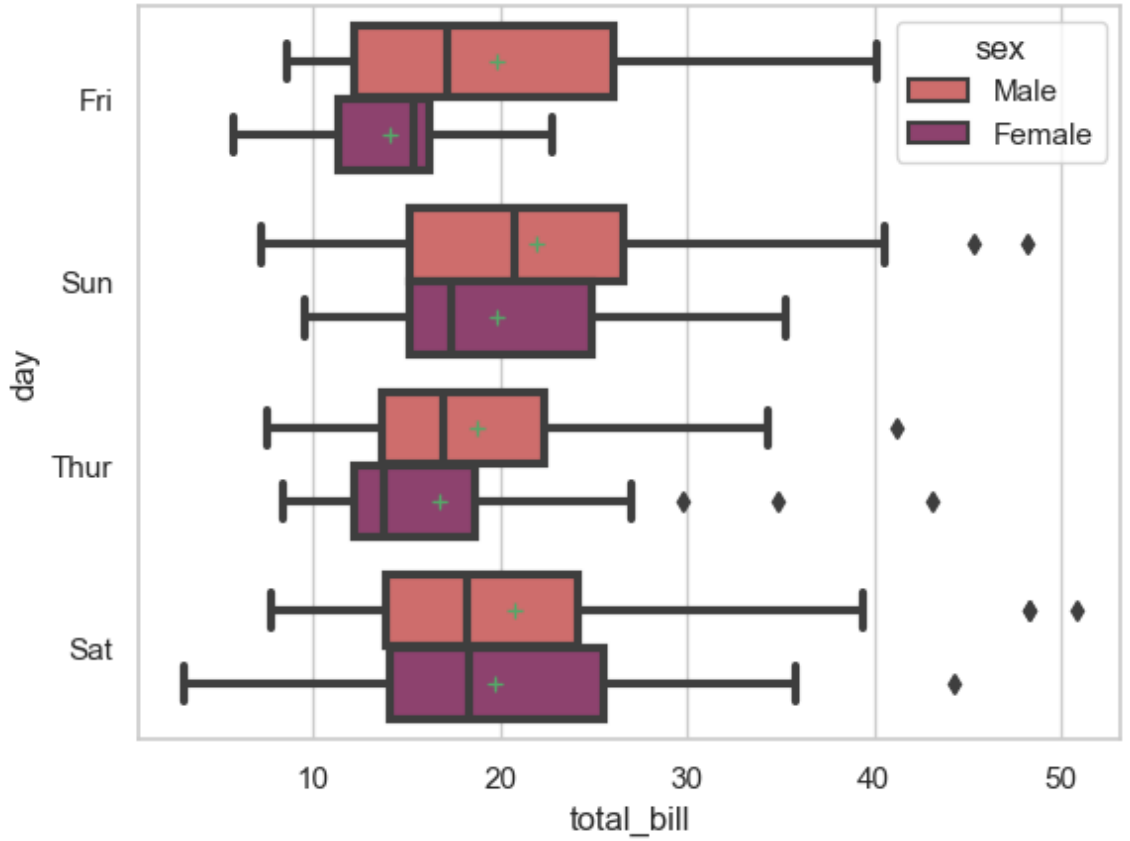
Boxplot using Seaborn (orient)

```
sns.boxplot(x='day', y='total_bill',
data=var, hue='sex', color='r',
order=['Fri', 'Sun', 'Thur', 'Sat'],

        showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'}, linewidth=3,

        palette='flare', orient='v')


plt.show()
```



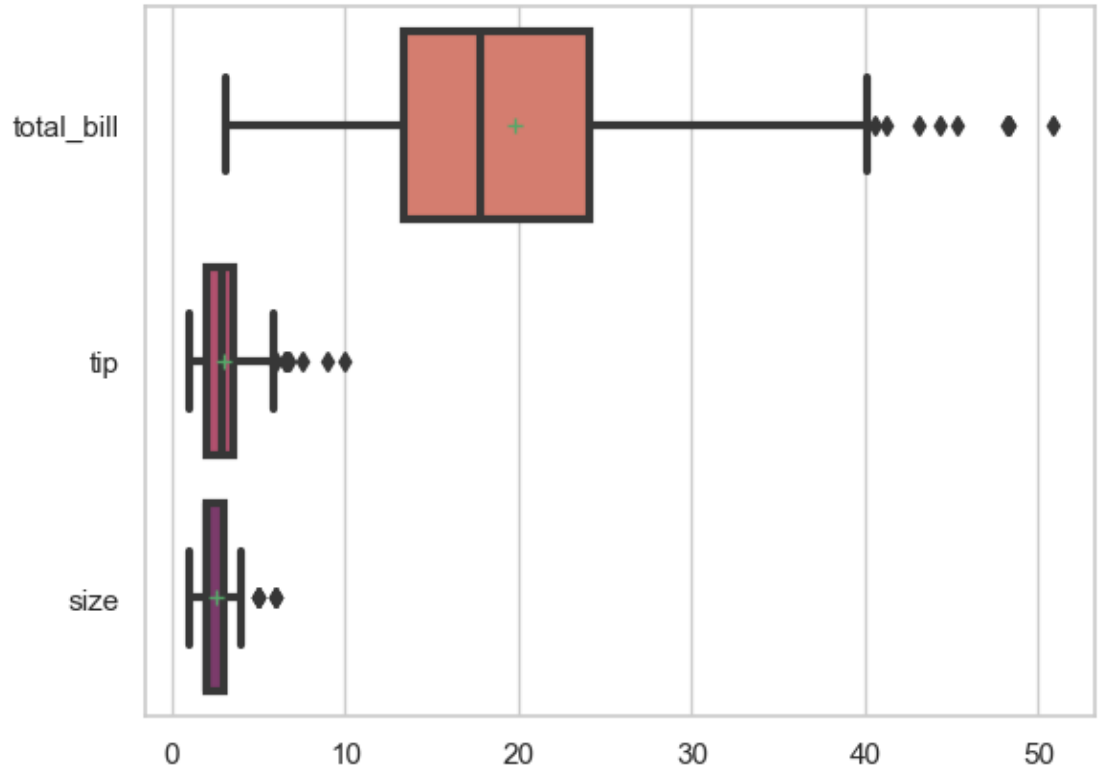
Boxplot using Seaborn (without_orient)

```
sns.boxplot(x='total_bill', y='day',
data=var, hue='sex', color='r',
order=['Fri', 'Sun', 'Thur', 'Sat'],

        showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'}, linewidth=3,

        palette='flare')


plt.show()
```



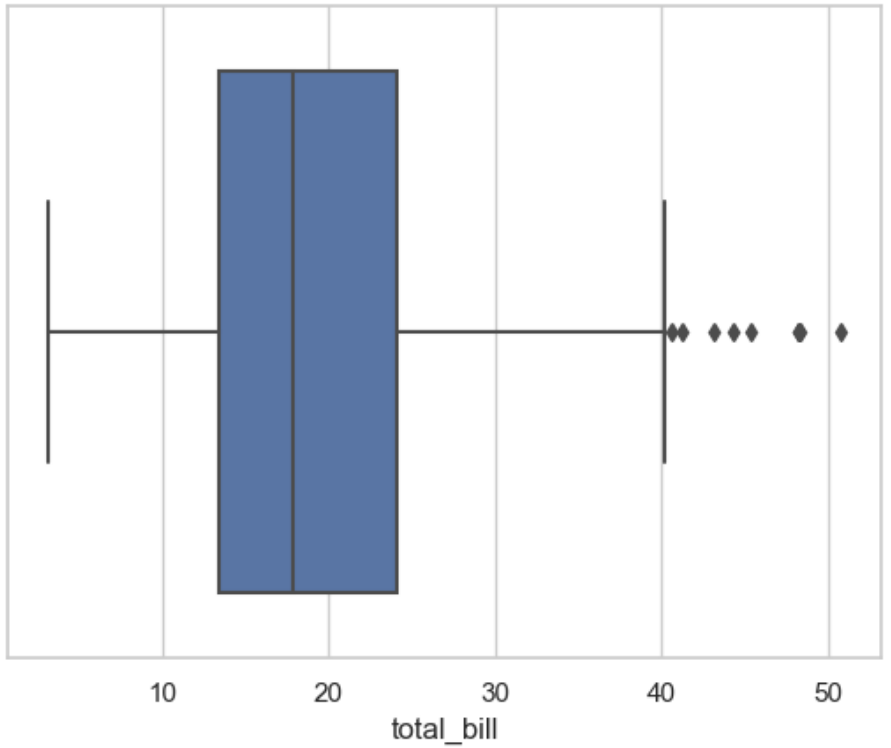
Boxplot using Seaborn (orient='h')

```
sns.boxplot(data=var,
showmeans=True,
meanprops={"marker":"+",
"markeredgecolor":'g'}, linewidth=3,
        palette='flare', orient='h')


plt.show()
```



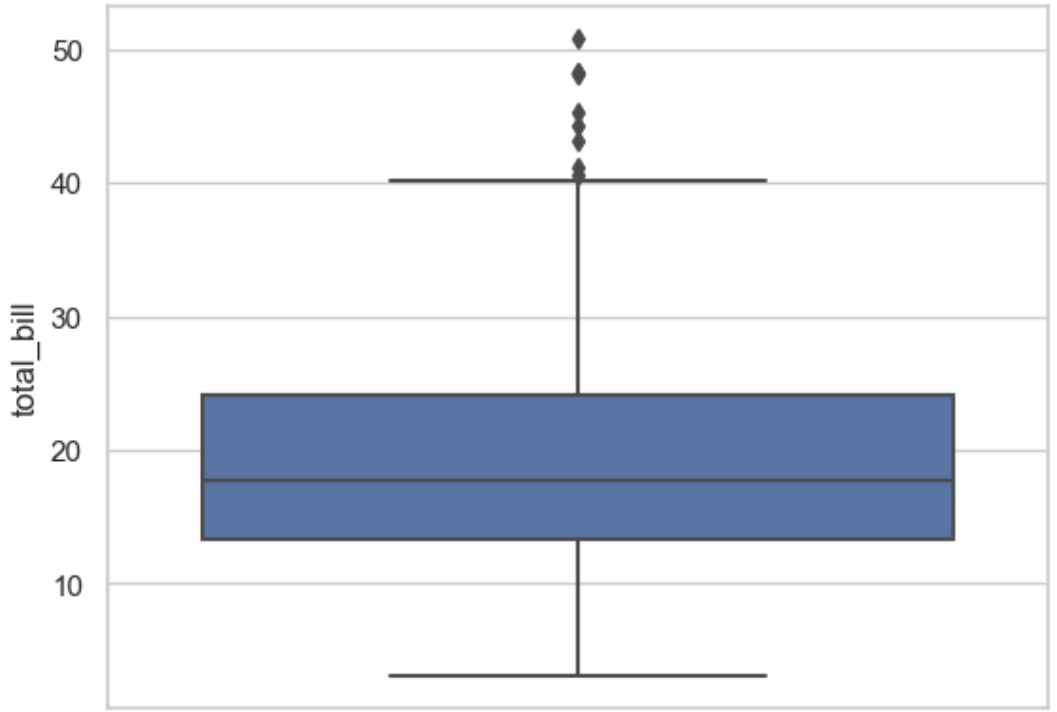
Boxplot using Seaborn (single plot, h)

```
sns.boxplot(x=var["total_bill"] )

plt.show()
```



Boxplot using Seaborn (single plot, v)

```
sns.boxplot(y=var["total_bill"] )

plt.show()
```



Cat plot using Seaborn

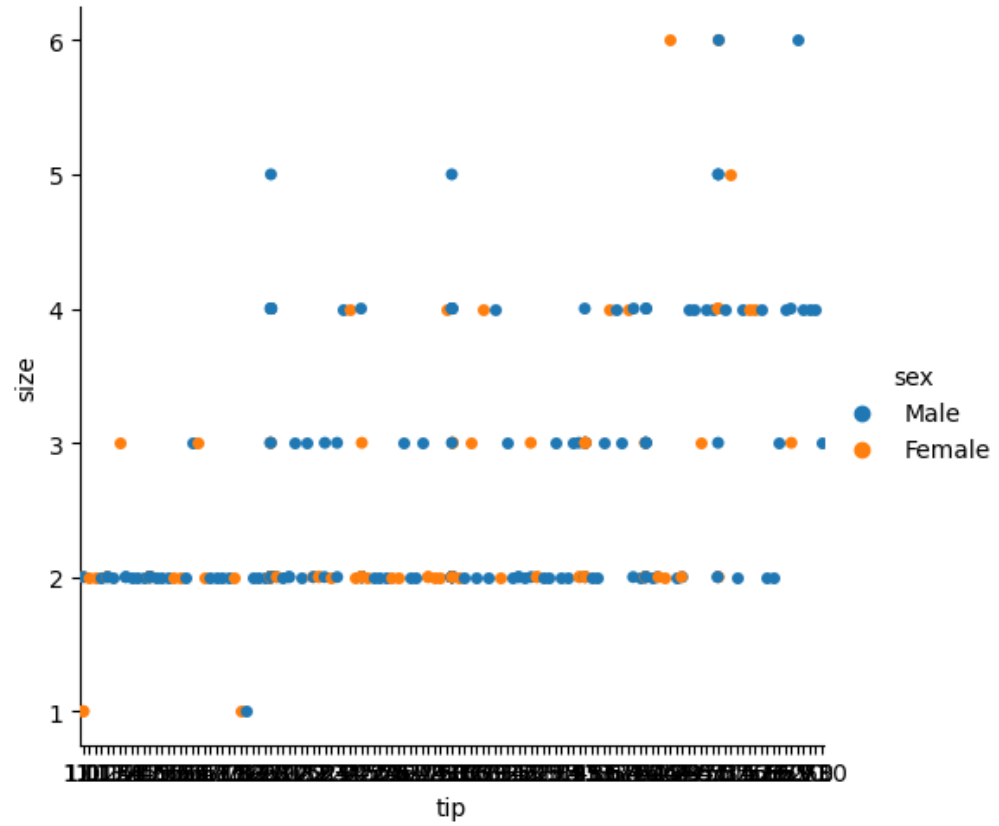
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

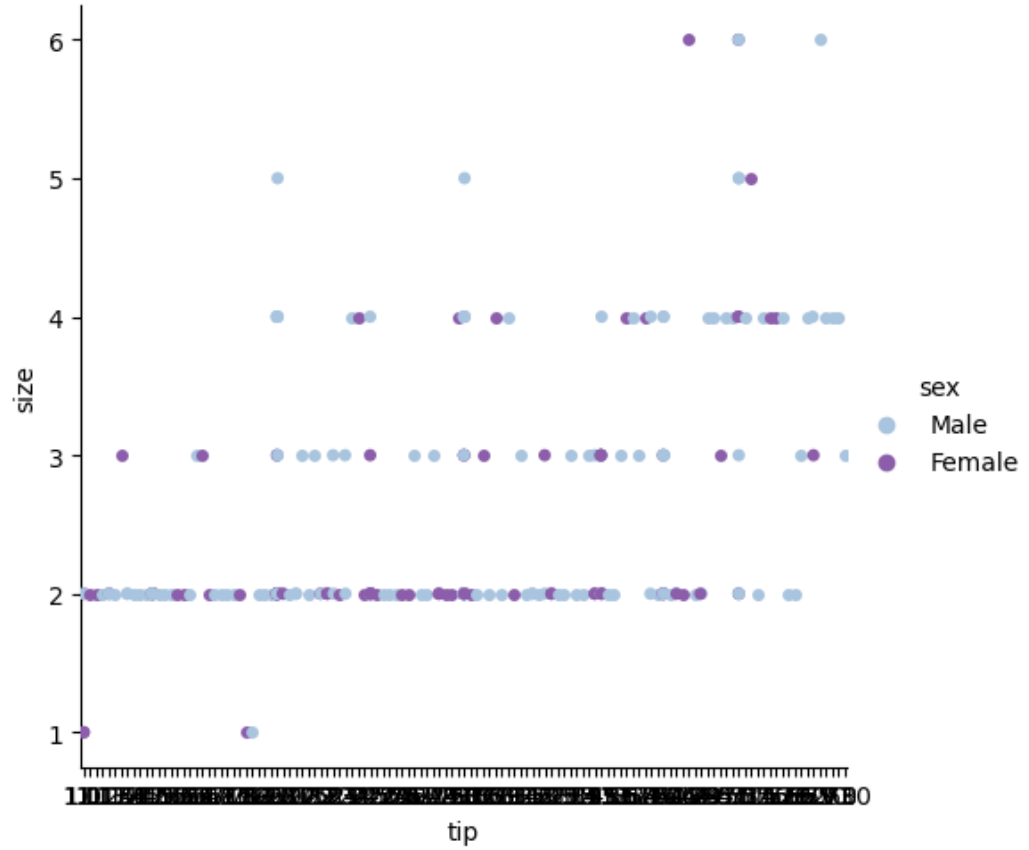
Cat plot using Seaborn

```
sns.catplot(x='tip', y='size', data=var,
hue='sex')

plt.show()
```



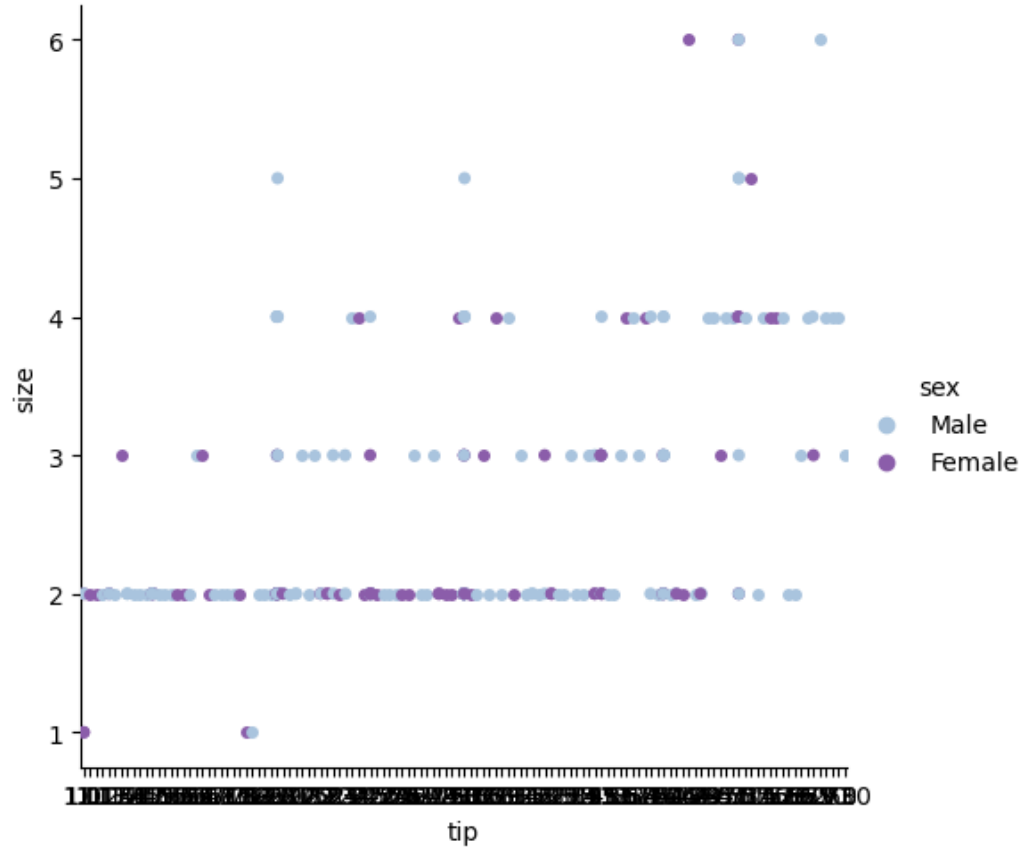
Cat plot using Seaborn (palette)

```
sns.catplot(x='tip', y='size', data=var,
hue='sex', palette='BuPu')

plt.show()
```



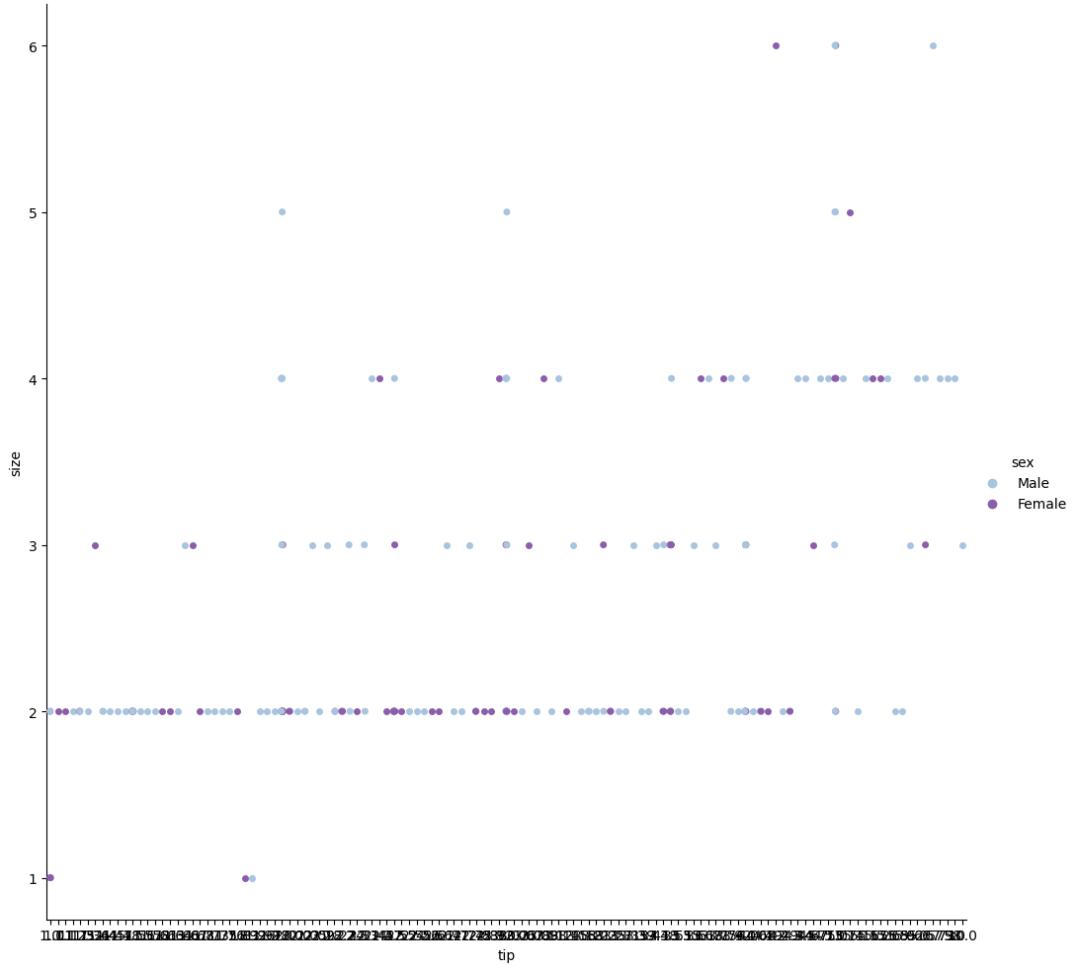
Cat plot using Seaborn (palette)

```
sns.catplot(x='tip', y='size', data=var,
hue='sex', palette='BuPu')

plt.show()
```



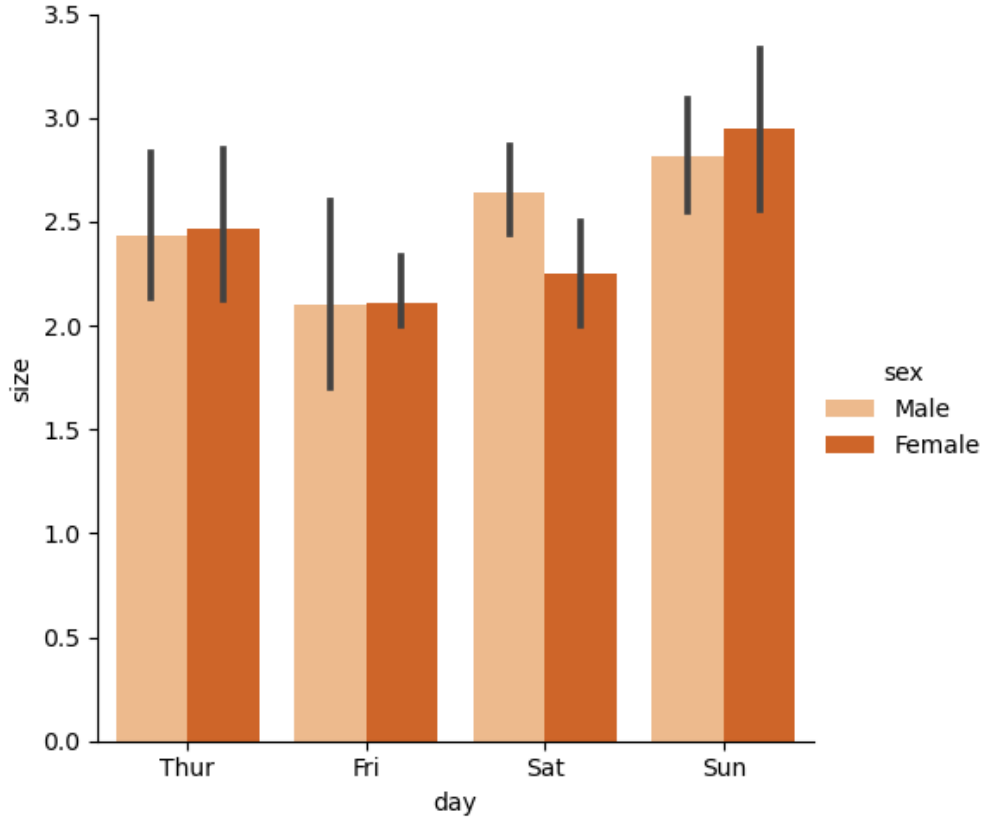
Cat plot using Seaborn (height)

```
sns.catplot(x='tip', y='size', data=var,
hue='sex', palette='BuPu', height=10)

plt.show()
```



Cat plot using Seaborn (kind='bar')

```
sns.catplot(x='day', y='size', data=var,
hue='sex', palette='Oranges',
kind='bar')

plt.show()
```



Cat plot using Seaborn (using kind)

You can make the following plots using the kind function.

Categorical scatterplots:

- `stripplot()` (with `kind="strip"`; the default)
- `swarmplot()` (with `kind="swarm"`)

Categorical distribution plots:

- `boxplot()` (with `kind="box"`)
- `violinplot()` (with `kind="violin"`)
- `boxenplot()` (with `kind="boxen"`)

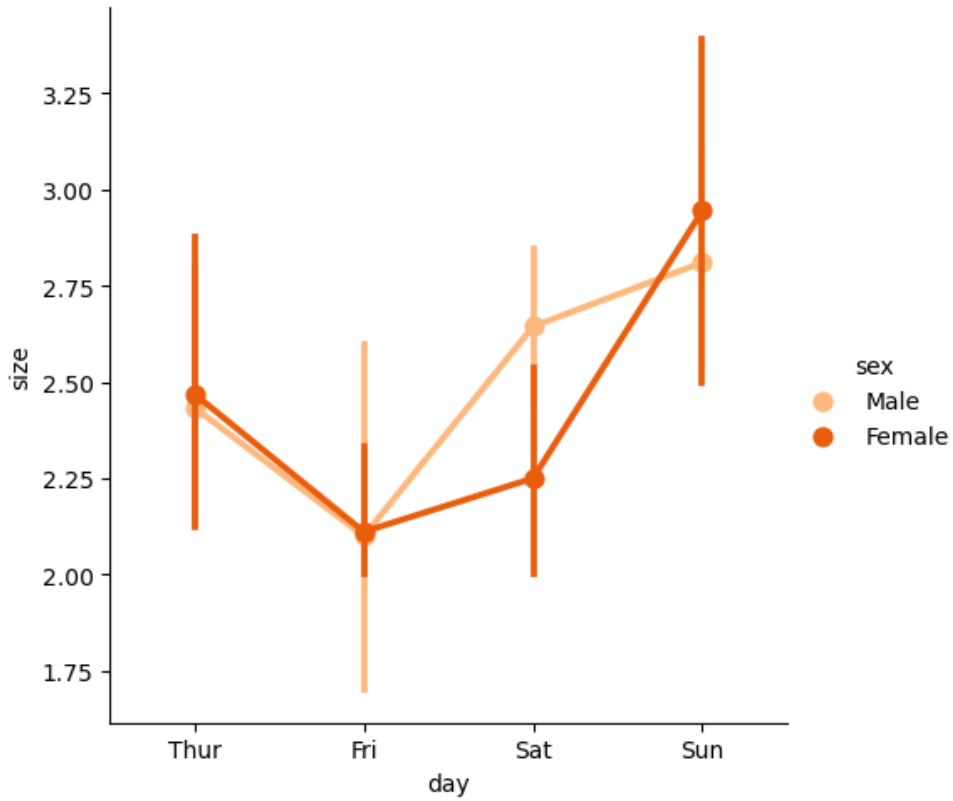
Categorical estimate plots:

- `pointplot()` (with `kind="point"`)
- `barplot()` (with `kind="bar"`)
- `countplot()` (with `kind="count"`)

Reference: https://seaborn.pydata.org/generated/seaborn.catplot.html

Cat plot using Seaborn (kind='point')

```
sns.catplot(x='day', y='size', data=var,
hue='sex', palette='Oranges',
kind='point')

plt.show()
```



Facet grid in Seaborn (multiple plots)

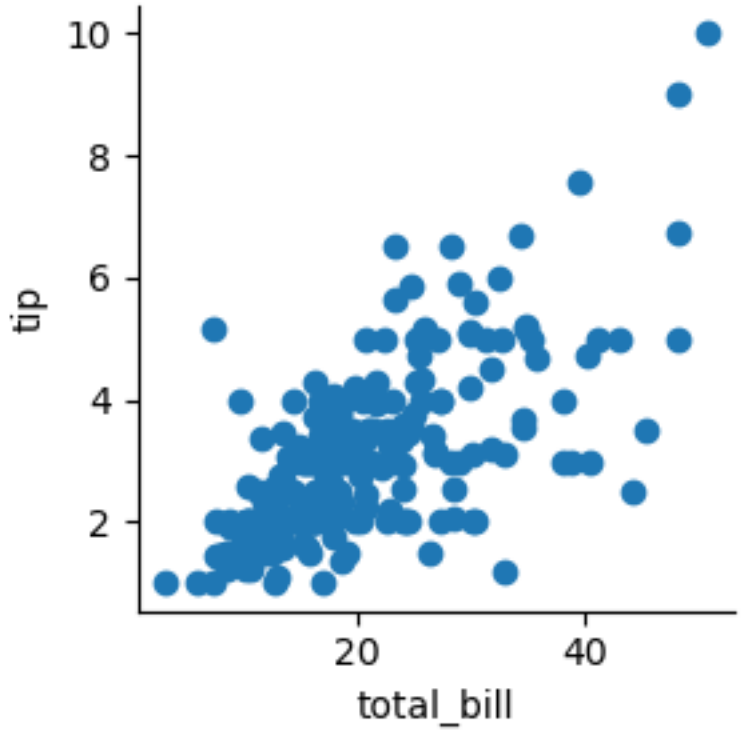
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

var = sns.load_dataset("tips")

var
```

	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

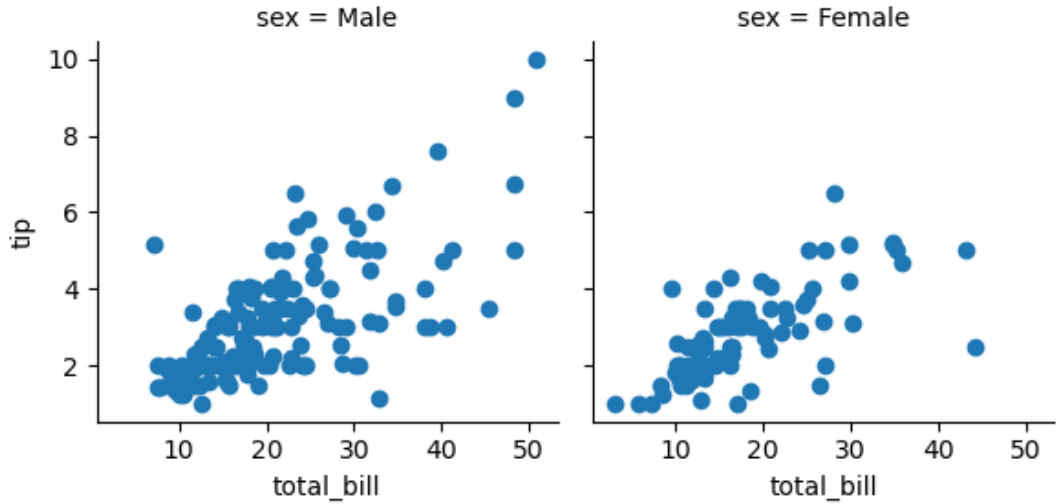
Facet grid in Seaborn (multiple plots)

```
fg = sns.FacetGrid(var)

fg.map(plt.scatter, "total_bill", "tip")

plt.show()
```



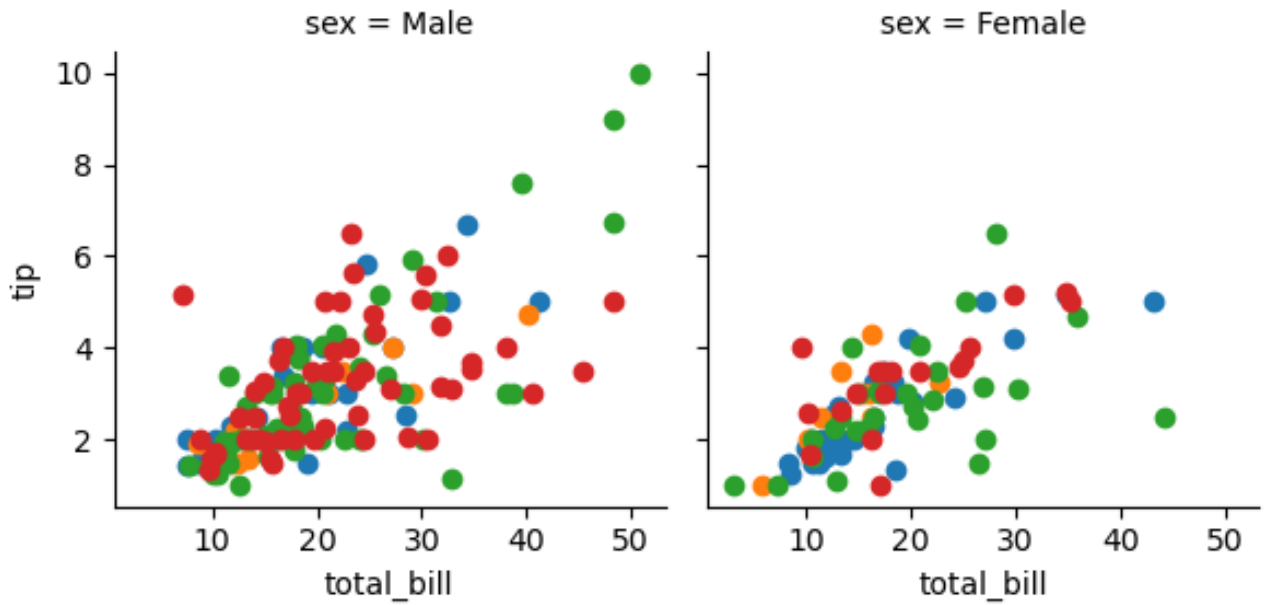
Facet grid in Seaborn (multiple plots)

```
fg = sns.FacetGrid(var, col='sex')

fg.map(plt.scatter, "total_bill", "tip")

plt.show()
```



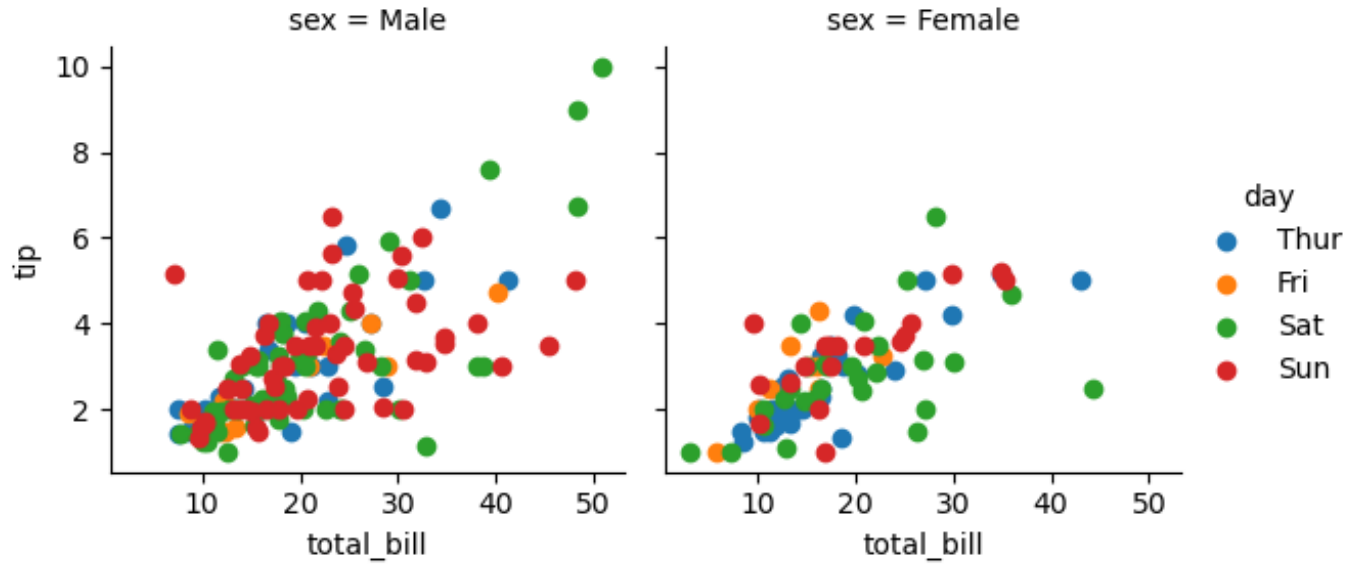
Facet grid in Seaborn (multiple plots)

```
fg = sns.FacetGrid(var, col='sex', hue='day')

fg.map(plt.scatter, "total_bill", "tip")

plt.show()
```



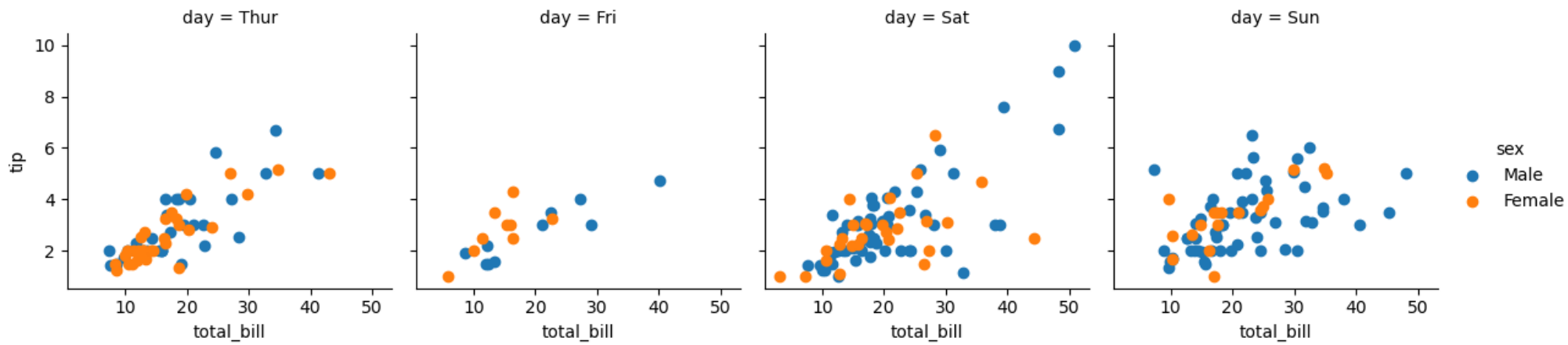
Facet grid in Seaborn (adding legend)

```
fg = sns.FacetGrid(var, col='sex', hue='day')

fg.map(plt.scatter, "total_bill", "tip").add_legend()

plt.show()
```



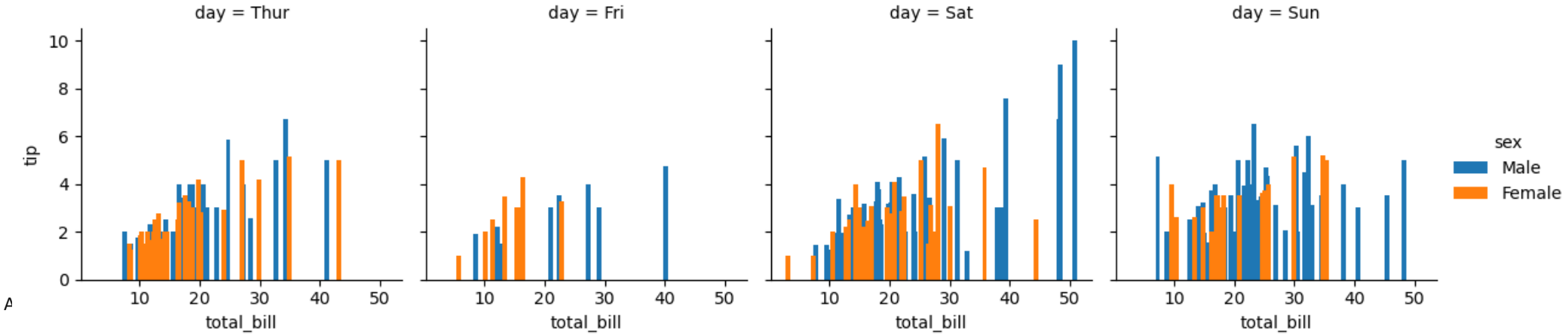
Facet grid in Seaborn (multiple plots)

```
fg = sns.FacetGrid(var, col='day', hue='sex')

fg.map(plt.scatter, "total_bill", "tip").add_legend()

plt.show()
```



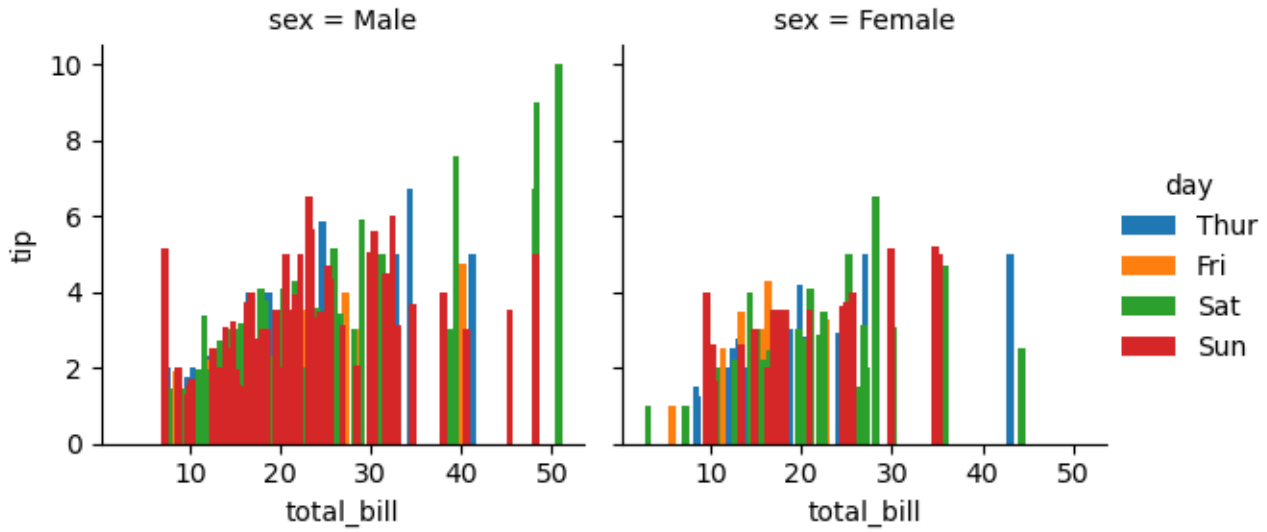
Facet grid in Seaborn (different plots)

```
fg = sns.FacetGrid(var, col='day', hue='sex')

fg.map(plt.bar, "total_bill", "tip").add_legend()

plt.show()
```



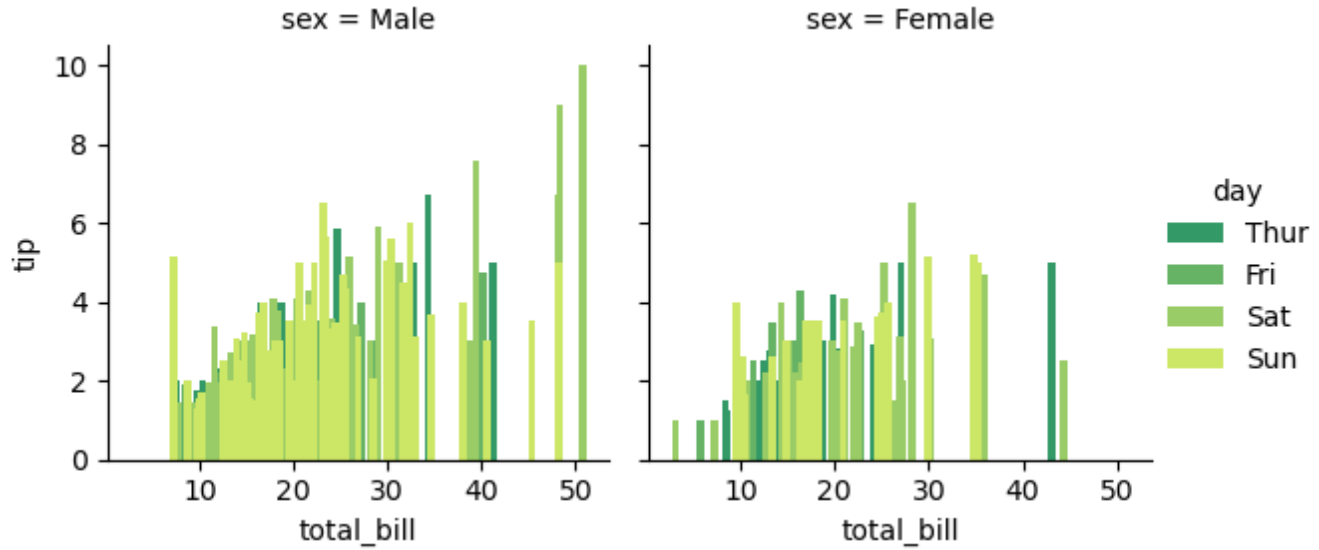
Facet grid in Seaborn (col, hue)

```
fg = sns.FacetGrid(var, col='sex', hue='day')

fg.map(plt.bar, "total_bill", "tip").add_legend()

plt.show()
```



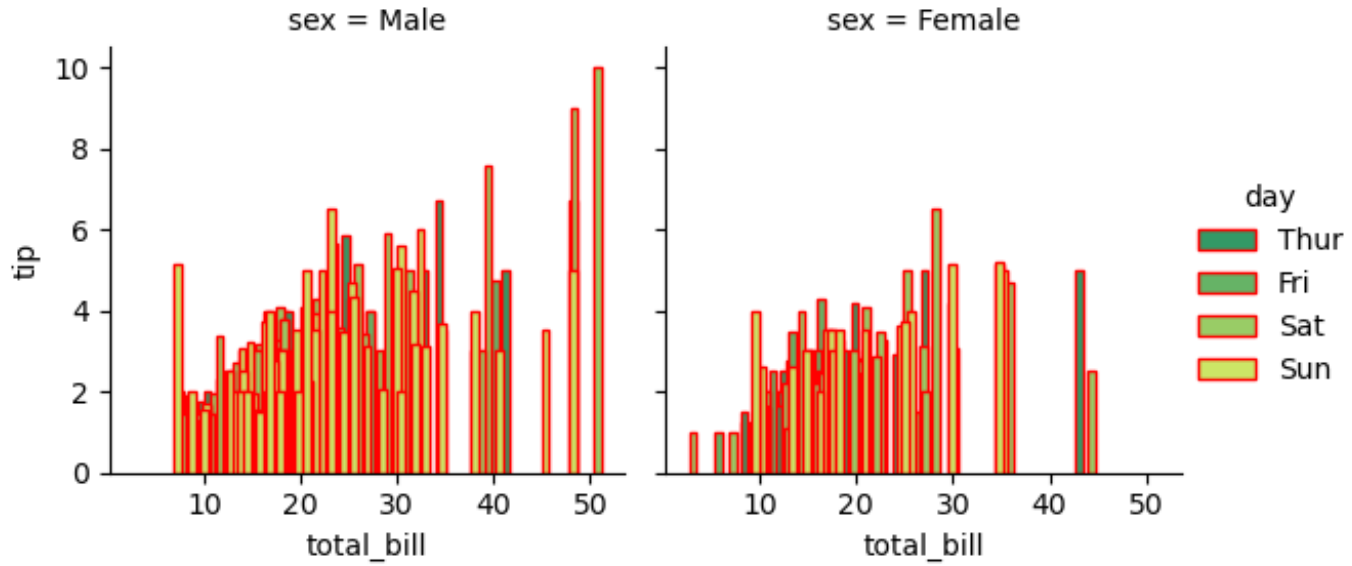
Facet grid in Seaborn (palette)

```
fg = sns.FacetGrid(var, col='sex', hue='day', palette='summer')

fg.map(plt.bar, "total_bill", "tip").add_legend()

plt.show()
```



Facet grid in Seaborn (edgecolor)

```
fg = sns.FacetGrid(var, col='sex', hue='day', palette='summer')

fg.map(plt.bar, "total_bill", "tip", edgecolor='r').add_legend()

plt.show()
```



Important links:

Documentation in Seaborn:

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

Seaborn datasets:

https://github.com/mwaskom/seaborn-data

Thank you all!
See you in next class… :)