Python-workshop

February 27, 2024

1 Python data wrangling workshop

An introduction to the pandas package Bailie Wynbelt / Jeff Oliver 3 February, 2023

1.0.1 Objectives / Learning Outcomes

1. Manipulate data and extract information from datasets using pandas
2. Create descriptive summary statistics
3. Output data visualizations with (plot9)

1.1 Data Science: more fun, less pain

But wait…you might be wondering what is pandas and how can we use it in Data Science?

Python is a dynamic language that can be used for a variety of problems these ranging from Software Enginnering to Data Science. Today, we are going to focus on how to use Python, specifically the pandas package, to wrangle, analyze, and visualize data. The pandas package is a collection of functions that allows us to easily manipulate, summarize, and visualize data. In this lesson, we will use the pandas package and the iris dataset to wrangle data, create summary statistics, and develop appealing visualizations.

1.2 Let's get started!

First we need to setup our environment in Jyupter notebook

We are using the pandas library, which contains a collection of functions that allows for efficient data manipulation and analysis. To use this package we need to import the package.

To get started we can create a markdown box at the beginner of our file that states the goal of the document, name, email, and date. After that we can import the package.

Descriptive statisitcs and visualization with pandas Bailie Wynbelt wynbeltb@arizona.edu 2024-02-04

[1]:
```
#import statements
import pandas as pd

import numpy as np

from plotnine import *
```

```
c:\Users\wynbe\anaconda3\lib\site-
packages\pandas\core\computation\expressions.py:21: UserWarning: Pandas requires
version '2.8.4' or newer of 'numexpr' (version '2.8.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
c:\Users\wynbe\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60:
UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version
'1.3.4' currently installed).
  from pandas.core import (
C:\Users\wynbe\AppData\Local\Temp\ipykernel_22880\235706412.py:2:
DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

Now that we have imported all the required packages, we can read in the data. To do this we will use the read_csv() function from pandas

[20]:
```
#import iris data set
#This is a dataframe, meaning it is a two-dimentional structure.
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.
 ↪data' #get csv url

iris =  pd.read_csv(csv_url, names =␣
 ↪['sepal_length','sepal_width','petal_length','petal_width','species'])

iris
```

[20]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
..	…	…	…	…	…
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

[150 rows x 5 columns]

1.3 Explore the Data

We have imported the pandas package and the dataset! Now we can start exploring the dataset and working on summarizing the data.

Some of my favorite ways of exploring the dataset include the following:

[12]:
```
#Gives the number of rows and columns in the format (rows, columns)
iris.shape
```

[12]:
```
(150, 5)
```

[13]:
```
#Shows the name ofthe column, number of non-null values, and the datatype.
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[14]:
```
#Returns the first 5 rows
iris.head()
```

[14]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

[15]:
```
#Returns the last 5 rows
iris.tail()
```

[15]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

We have imported the pandas package/dataset and have explored the data! Now we can start

working on summarizing the data.

1.4 Summarizing the data

How can we create descriptive statistics for the iris dataset? -Means -Standard errors -For each trait -For each species

[16]:
```
#Quick statisitic summary of data
iris.describe()
```

[16]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[42]:
```
#Find the mean of sepal length for all species
iris_mean = iris["sepal_length"].mean()

iris_mean
```

[42]:
```
5.843333333333334
```

[43]:
```
#Find the standard deviation of sepal length for all species
iris_std = iris["sepal_length"].std()

iris_std
```

[43]:
```
0.828066127977863
```

We have found the mean and standard deviation for sepal_length. However, there are also different species, so what if there are differences in traits between species?

To explore this question we can use the groupby function.

[26]:
```
#find the mean for each species for each trait (column)
iris_mean = iris.groupby('species').mean()

iris_mean
```

[26]:

species	sepal_length	sepal_width	petal_length	petal_width
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.770	4.260	1.326
Iris-virginica	6.588	2.974	5.552	2.026

What if we also want to find the standard deviation for each trait for each species? We can use the agg function alongside the groupby function.

[24]:
```
#find the mean and standard deviation for each species for each trait (column)
iris_stats = iris.groupby('species').agg(['mean', 'std'])

iris_stats
```

[24]:
```
                 sepal_length           sepal_width           petal_length  \
                         mean       std        mean       std         mean
species
Iris-setosa             5.006  0.352490       3.418  0.381024        1.464
Iris-versicolor         5.936  0.516171       2.770  0.313798        4.260
Iris-virginica          6.588  0.635880       2.974  0.322497        5.552

                          petal_width
                       std        mean       std
species
Iris-setosa       0.173511       0.244  0.107210
Iris-versicolor   0.469911       1.326  0.197753
Iris-virginica    0.551895       2.026  0.274650
```

Great! Now we have a descriptive statisitcs for all species and traits. Similar steps can be taken if you just want to find descriptive statisitics for one column or trait. Lets explore how to do this below.

[25]:
```
#find the mean per species for sepal length
iris_mean = iris.groupby('species')['sepal_length'].mean()

iris_mean
```

[25]:
```
species
Iris-setosa        5.006
Iris-versicolor    5.936
Iris-virginica     6.588
Name: sepal_length, dtype: float64
```

[ ]:
```
#find the mean and standard deviation for sepal length of each species
iris_stats = iris.groupby('species')['sepal_length'].agg(['mean', 'std'])

iris_stats
```

[ ]:
```
                  mean       std
species
Iris-setosa      5.006  0.352490
Iris-versicolor  5.936  0.516171
Iris-virginica   6.588  0.635880
```

1.5 Plotting Data

So far we have

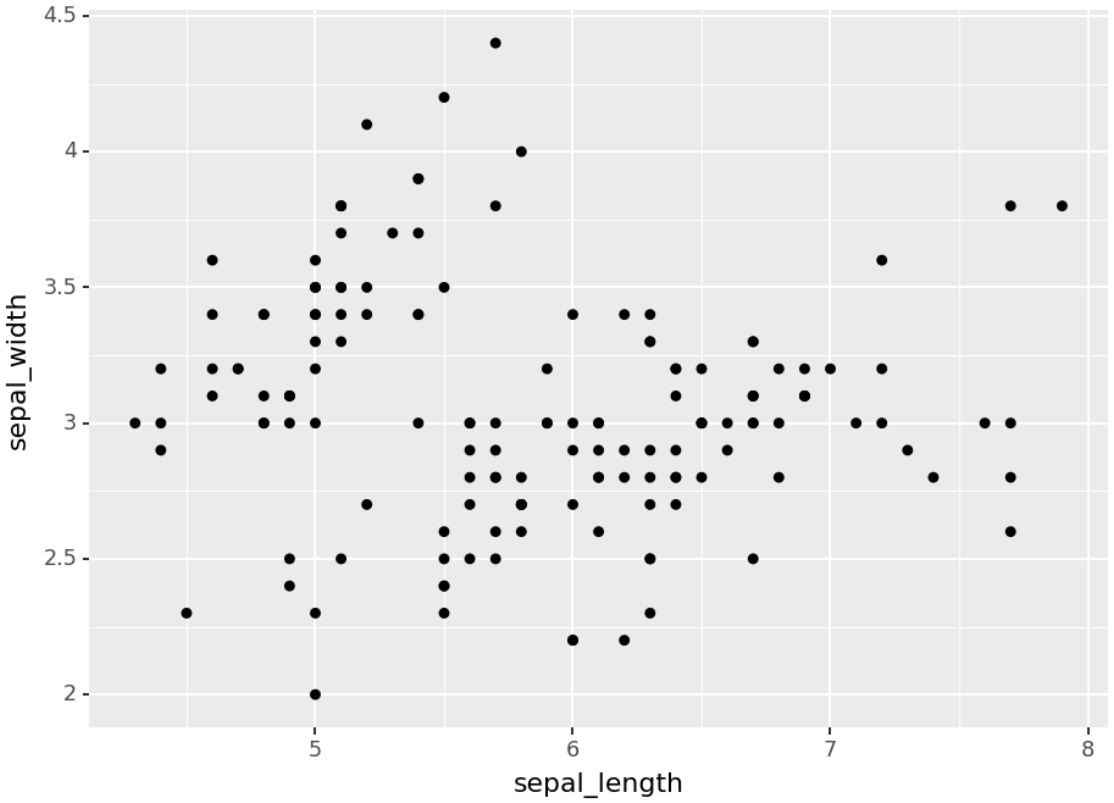
-read the data -explored the data -created descriptive statistics

We are now read to visualize the data! In R, people commonly use ggplot2 to visualize datasets. In python, we will use plot9 which essentially functions similarly to ggplot2 and outputs similar visualizations

First, we need to create a general plot and then state what type of plot we want.

We are going to create a scatterplot displaying sepal length on the x-axis and sepal width on the y-axis.

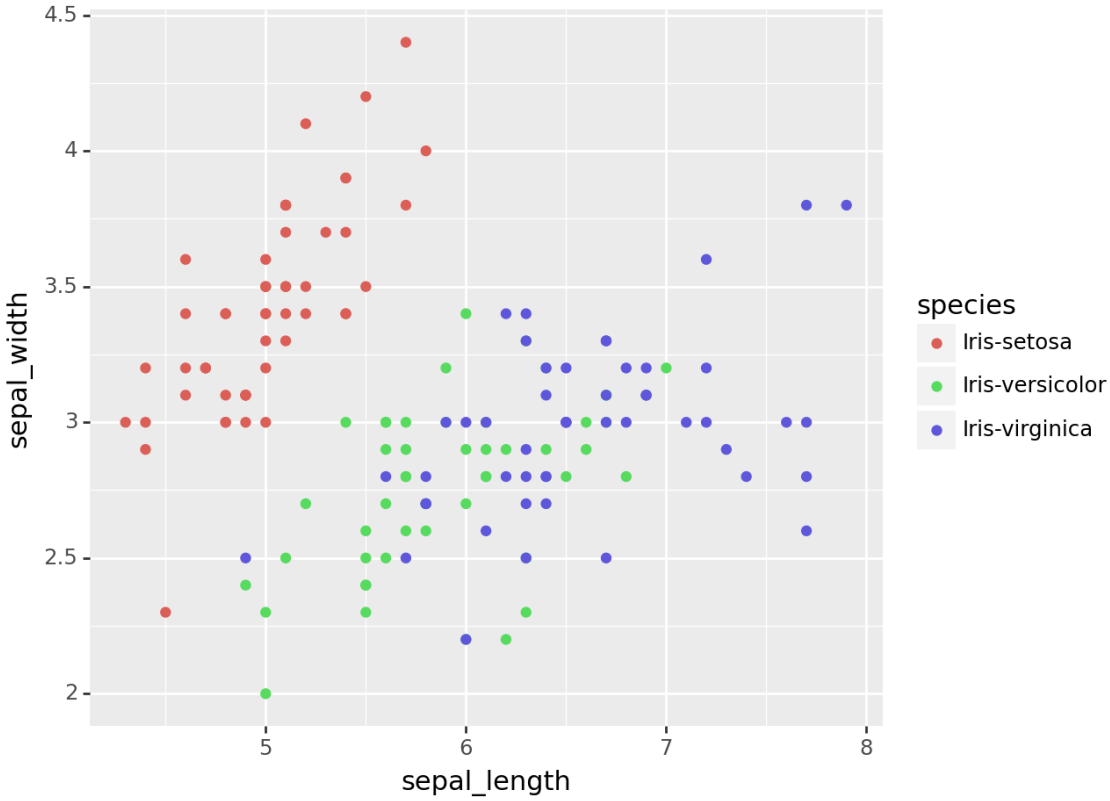
[27]:
```
(ggplot(iris, aes(x = "sepal_length", y = "sepal_width"))
+ geom_point())
```



[27]:
```
<Figure Size: (460 x 345)>
```

It looks like there is two distinct groupings within our scatterplot, lets explore this more by adding an extra argurment into the aes() function.

[28]:
```
(ggplot(iris, aes(x = "sepal_length", y = "sepal_width", color = "species"))
+ geom_point())
```



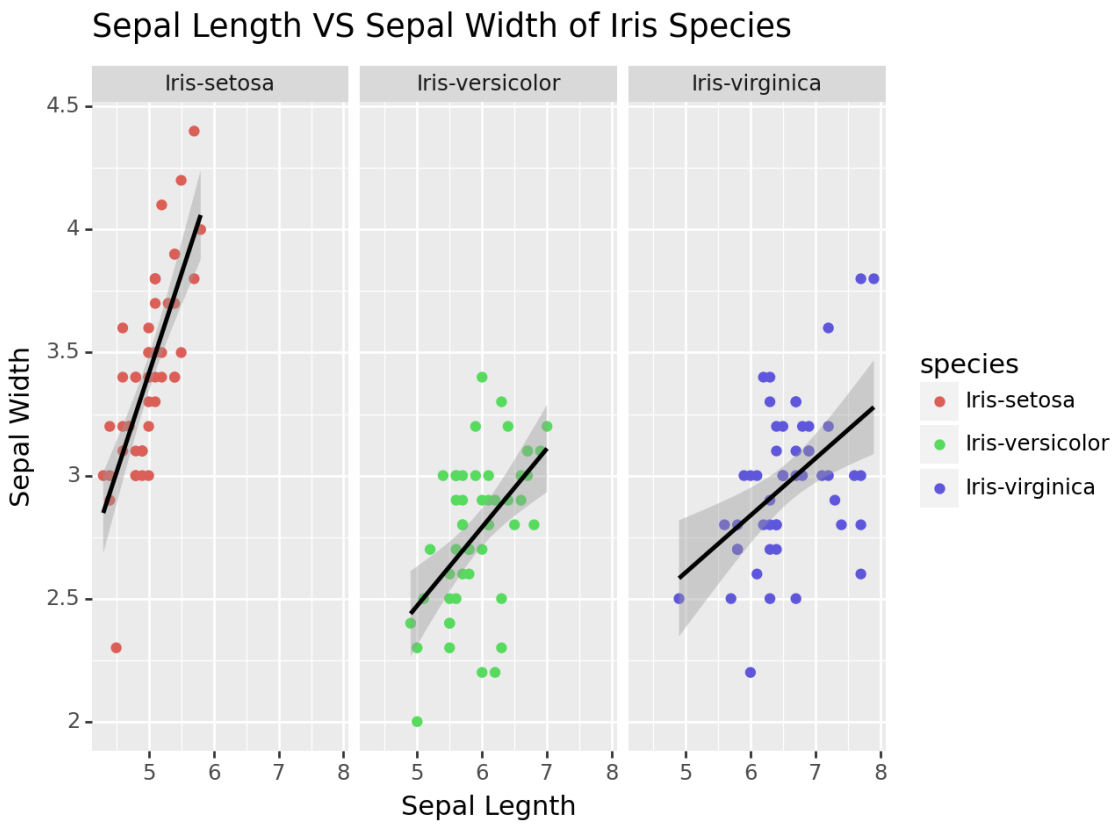
[28]:
```
<Figure Size: (640 x 480)>
```

Lastly, we can better visualize the differences between species by faceting and creating a linear model. This can be done by adding the facet_wrap and stat_smooth argument.

A facet wrap is dividing the graph into sections based on a particular variable. In our case, it is species.

Lastly, we can beautify the graph by adding labels and a title.

[29]:
```
(ggplot(iris, aes(x = "sepal_length",
                  y = "sepal_width",
                  color = "species")) +
        geom_point() +
        facet_wrap("~species") +
        stat_smooth(method = "lm", color = "black") +
        labs(x = "Sepal Legnth",
            y = "Sepal Width",
            title = "Sepal Length VS Sepal Width of Iris Species"))
```



[29]: <Figure Size: (640 x 480)>

2 Your turn!

Great! We were able to explore the data, find descriptive statisitcs, and create a visualization.

Now, it is your turn to work together and explore the palmers penguins dataset, with the end goal of finding descriptive statistics and a visualization.

First, lets start with loading in the data. We will do this together

[33]:
```
#load in the data
from palmerpenguins import load_penguins
penguins = load_penguins()
```

2.1 Explore the dataset

Employ one or two of the explorative functions we used before. Use which ever ones are your favorite!

[35]:
```
#explore the data
penguins.head()
```

[35]:
```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex  year
0       3750.0    male  2007
1       3800.0  female  2007
2       3250.0  female  2007
3          NaN     NaN  2007
4       3450.0  female  2007
```

[36]:
```
penguins.tail()
```

[36]:
```
       species island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
339  Chinstrap  Dream            55.8           19.8              207.0
340  Chinstrap  Dream            43.5           18.1              202.0
341  Chinstrap  Dream            49.6           18.2              193.0
342  Chinstrap  Dream            50.8           19.0              210.0
343  Chinstrap  Dream            50.2           18.7              198.0

     body_mass_g     sex  year
339       4000.0    male  2009
340       3400.0  female  2009
341       3775.0    male  2009
342       4100.0    male  2009
343       3775.0  female  2009
```

[37]:
```
penguins.describe()
```

[37]:
```
       bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g  \
count      342.000000     342.000000         342.000000   342.000000
mean        43.921930      17.151170         200.915205  4201.754386
std          5.459584       1.974793          14.061714   801.954536
min         32.100000      13.100000         172.000000  2700.000000
25%         39.225000      15.600000         190.000000  3550.000000
50%         44.450000      17.300000         197.000000  4050.000000
75%         48.500000      18.700000         213.000000  4750.000000
max         59.600000      21.500000         231.000000  6300.000000

              year
count   344.000000
mean   2008.029070
std       0.818356
min    2007.000000
```

```
25%     2007.000000
50%     2008.000000
75%     2009.000000
max     2009.000000
```

2.2 Find Descriptive Statisitcs

We want to find descriptive statistics for the dataset! For your challenge, find the mean and standard deviation of the "body_mass_g" column per species

[41]:
```
penguins_stats = penguins.groupby('species')['body_mass_g'].agg(['mean', 'std'])

penguins_stats
```

[41]:
```
                  mean         std
species
Adelie     3700.662252  458.566126
Chinstrap  3733.088235  384.335081
Gentoo     5076.016260  504.116237
```

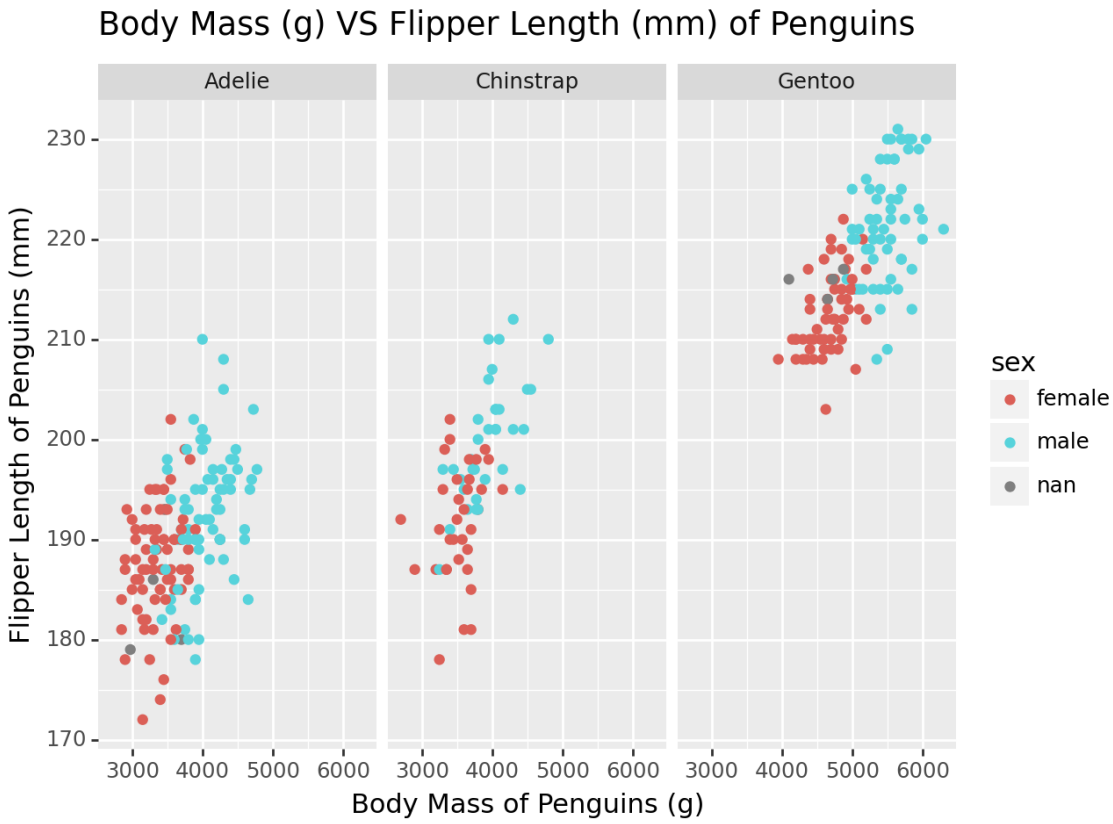
2.3 Visualization

For your next challenge... try creating a visualization that displays body mass on the x-axis vs flipper length on the y-axis with color differented by sex (color in the aes() argument) and the graph faceted by species. Add labels that you see fit to the graph.

Steps to be taken 1) Create a scatterplot (geom_point) with "body_mass_g" on the x-axis and "flipper_length_mm" on the y-axis. Dont forgot to add aes(color =) set to "sex" ! 2) Add a facet_wrap by "species" 3) Add labels with the labs() argument

[36]:
```
(ggplot(data = penguins,
       mapping = aes(x = "body_mass_g",
                     y = "flipper_length_mm",
                     color = "sex")) +
  geom_point() +
  facet_wrap("~species") +
  labs(x = "Body Mass of Penguins (g)",
       y = "Flipper Length of Penguins (mm)",
       title = "Body Mass (g) VS Flipper Length (mm) of Penguins"))
```

```
c:\Users\wynbe\anaconda3\lib\site-packages\plotnine\layer.py:364:
PlotnineWarning: geom_point : Removed 2 rows containing missing values.
```



[36]: <Figure Size: (640 x 480)>