INFO5002: Intro to Python for Info Sys

Week 10

Northeastern University

Slides created by: Zachary Doucet

Week 10

I. Numpy

II. Pandas

III. Visualization

Numpy

numpy.org

Python lists are limiting

- Limited operations, heterogeneous data storage--operation that works at one index may not work at another.
- We will be using Numpy.

```
pip install numpy
```

```python
import numpy as np
```

Array as primitive

- Numpy uses the array as a primitive—kind of like a Python list.

```
x = np.array([1,2,3,4,5,6])
```

- Arrays can be indexed and are mutable like python.

```
x[0] = 4
```

- Can also do slicing but be careful—returns a view instead of a copy. Mutating the view mutates the original!

```
y = x[:3]
y[0] = 6
```

Array Attributes

- arr.ndim: number of dimensions: 1 – vector, 2 – matrix, 3+ – tensor.
- arr.shape: returns the number of dimensions and data as tuple.
- arr.size: returns total number of elements.
- arr.dtype: returns the data type.

Functional array creation

- np.zeros(shape): create array of *shape* filled with 0’s.
- np.ones(shape): create array of *shape* filled with 1’s.
- np.empty(shape): create array of *shape* filled with random numbers (faster!).
- np.arange(start, stop, step): create vector of range from *start* to *stop* incrementing by *step*.
- np.linspace(start, stop, num=50): create vector of *num* evenly spaced numbers from *start* to *stop*.

Useful functions

- np.sort(arr): sorts in ascending order.
- np.concatenate((arr1, arr2, ...), axis=0): concatenates two or more arrays together along axis *axis*.
- np.reshape(arr, shape=shape): will reshape the array according to what you specify—must keep same # elems.
  - (12,) → (3,4) Good
  - (2, 4) → (3, 3) Bad

Conditional selection

- Can pass a conditional within brackets:

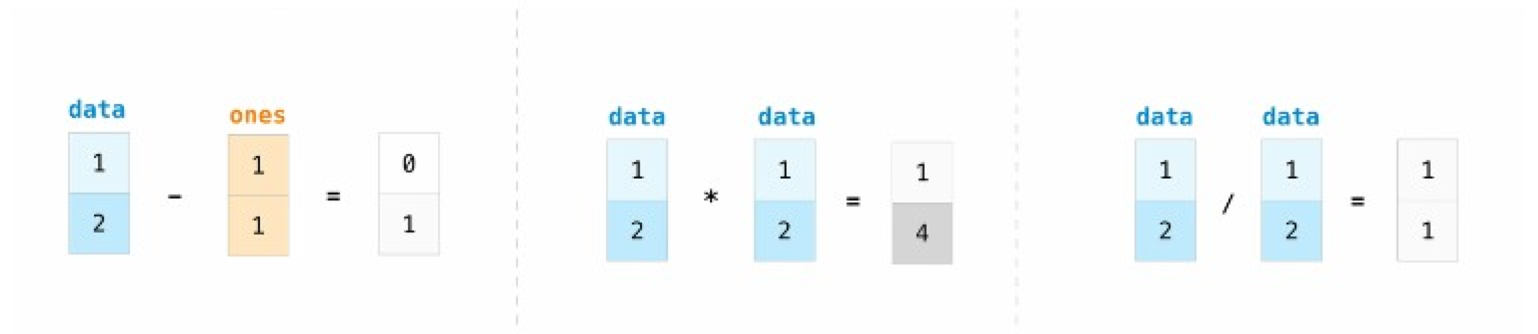
```
b = a[(a > 18) & (a < 25)]
```

- & is same as python's and and | is same as python's or.

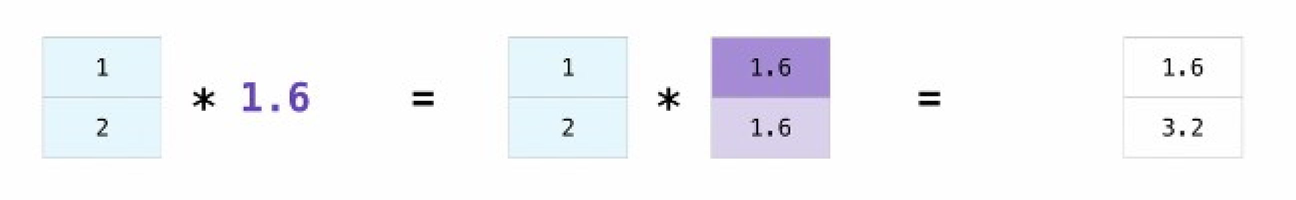
```
divisible_2_or_3 = a[(a%2==0)|(a%3==0)]
```

Operations

- +, -, /, * all do the operation element wise.



- If you try to perform an operation with two different shapes, it will attempt to broadcast to make it work.



Useful operator functions

- arr.sum(axis=-1): returns the sum of all the elements together along axis *axis*.
- arr.max(axis=-1): returns the largest along axis *axis*.
- arr.min(axis=-1): returns the smallest along axis *axis*.

Pandas

pandas.pydata.org

Even more power

- Pandas adds more data structures and power over Numpy.

```
pip install pandas
```

```python
import pandas as pd
```

Series and DataFrame as primitive

- Series: labelled one-dimensional array holding data of any type; integers, strings, Python object's.

```
0     4.0
1     5.0
2    12.0
3     NaN
4    32.0
5    18.0
dtype: float64
```

```
s = pd.Series([4, 5, 12, np.nan, 32, 18])
```

- DataFrame: two-dimensional data structure that acts like a table with rows and columns.

```
df = pd.DataFrame(np.array, index=row_names, columns=colum_names)
```

	A	B	C	D
2013-01-01	-0.365031	0.701977	0.381228	1.564787
2013-01-02	0.345290	-0.739007	-0.178305	1.069980
2013-01-03	0.675831	0.886833	-1.258269	1.183045
2013-01-04	0.448686	-0.578519	0.427933	-0.159340
2013-01-05	1.464388	2.221634	1.273367	1.046402
2013-01-06	-1.091492	0.479029	-2.464129	-2.946307

- Can also pass as a dictionary.

```
df2 = pd.DataFrame(
    {
        "A": 1.0,
        "B": pd.Timestamp("20130102"),
        "C": pd.Series(1, index=list(range(4)), dtype="float32"),
        "D": np.array([3] * 4, dtype="int32"),
        "E": pd.Categorical(["test", "train", "test", "train"]),
        "F": "foo",
    }
)
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

Useful functions and attributes

- df.head(n=5): return the *n* first rows.
- df.tail(n=5): return the *n* last rows.
- df.index: return the row labels.
- df.columns: return the col labels.
- df.to_numpy(): return a Numpy representation.

- df.describe(): shows quick statistical summary.

	A	B	C
count	100.000000	100.000000	100.000000
mean	102.366338	100.033401	99.982868
std	26.092785	25.816671	26.665102
min	31.517570	21.936161	22.466281
25%	84.552389	82.723932	80.393764
50%	103.406103	100.215987	101.635768
75%	119.650715	114.003662	117.620123
max	171.914097	170.454752	168.754208

- df.T: transpose

- df.sort_index(axis=0, ascending=True)

- df.sort_values(by, axis=0, ascending=True)

Indexing

- Pass in a single argument in brackets to get a series of the corresponding column.

```
df["C"]
```

- Pass in a slice to get matching rows.

```
     row      col
df.iloc[7:10, 1:3]
```

- If you want to work with direct indexing can use **iloc**.

Conditional selection

- Similar to numpy.

```
df[df["A"]>18]
```

- Can use **isin()** for non-number data.

```
df[df2["C"].isin(["rain", "storm"])]
```

Missing data

- df.dropna(): drop any row with missing data.
- df.fillna(value=None): fill any missing data with *value*.
- df.isna(): returns a new DataFrame where all cells with missing values are set to False; otherwise, True.

	0	1	2
0	False	True	False
1	True	False	False
2	True	False	True

Importing/Exporting

- pd.read_csv(“filepath.csv”): to read a csv and load as DataFrame.
- df.to_csv(“filepath.csv”): save DataFrame to csv.
- pd.read_parquet(“filepath.parquet”)
- df.to_parquet(“filepath.parquet”)
- pd.read_excel(“filepath.xlsx”)
- df.to_excel(“filepath.xlsx”)

Visualisation

DSfS 43-53

We want to see

- Currently your data is a bunch of numbers and strings which is not easily communicable.
- Data visualisation is the study of communicating data across visually.
  - Charts, tables, graphs, etc.

We will be using matplotlib

- Python library for creating visualisations.

```
pip install matplotlib
```

- Can then import it.

```
import matplotlib.pyplot as plt
```

General commands

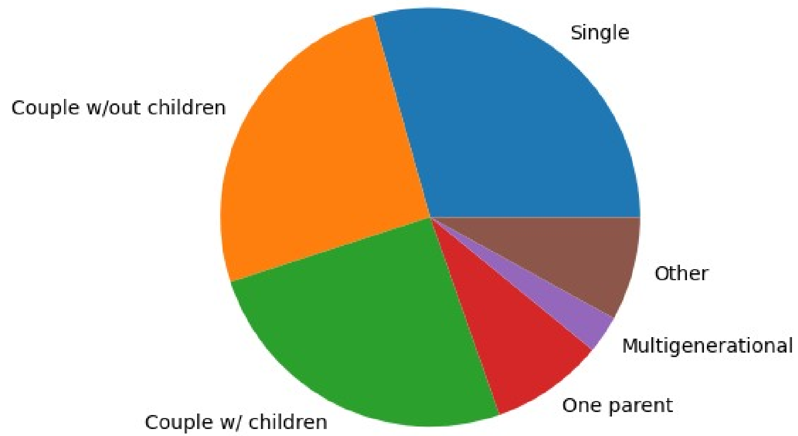
- Title: `plt.title("Title")`
- X-label: `plt.xlabel("Time")`
- Y-label: `plt.ylabel("Time")`
- Super title: `plt.suptitle("Super title")`
- Tight-layout: `plt.tight_layout()`
- Legend: `plt.legend()`

- Add grid lines: `plt.grid()`
- Set figure size: `plt.figure(figsize=(width, height))`
- X-axis range: `plt.xlim(min, max)`
- Y-axis range: `plt.ylim(min, max)`
- X-ticks: `plt.xticks()`
- Y-ticks: `plt.yticks()`
- Display: `plt.show()`
- Save to disk: `plt.savefig("filename.png")`

Pie Chart

- You give a 1D collection (x) where the proportion of each is computed as: `x / sum(x)`
- Can optional give a string list of labels.

```
plt.pie(x, labels=labels])
```



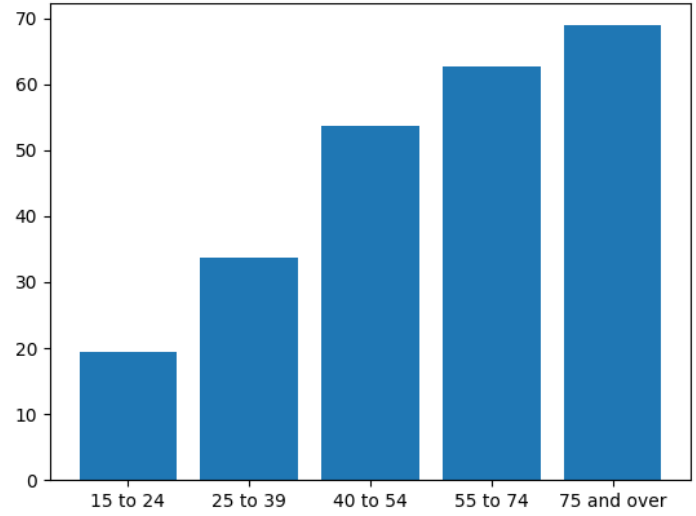
StatCan21 Houshold Disttribution

Bar Charts

- You give the bar labels (x) and the height of each (height)
- Can optionally specify:
  - Width of each bar (width)
  - Bar alignment:

    “center” or “edge”

```
plt.bar(x, height,
   width=0.8, align="center")
```

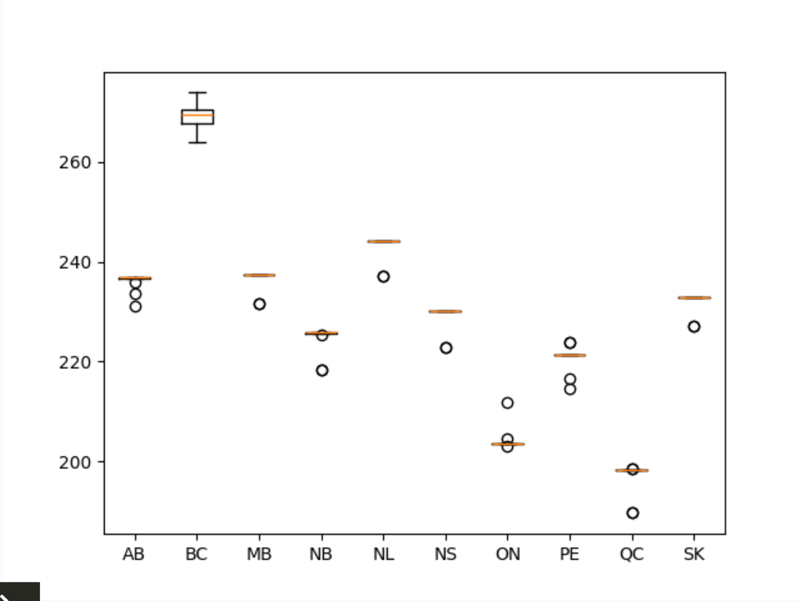


StatCan21 Toronto Household ownership rate by age.

Box Plot

- Give data as a 2D collection where each entry is a column and for each column you give all the raw data.
- Can optionally specify:
  - Labels (tick_labels)

```
plt.boxplot(x, tick_labels=labels)
```

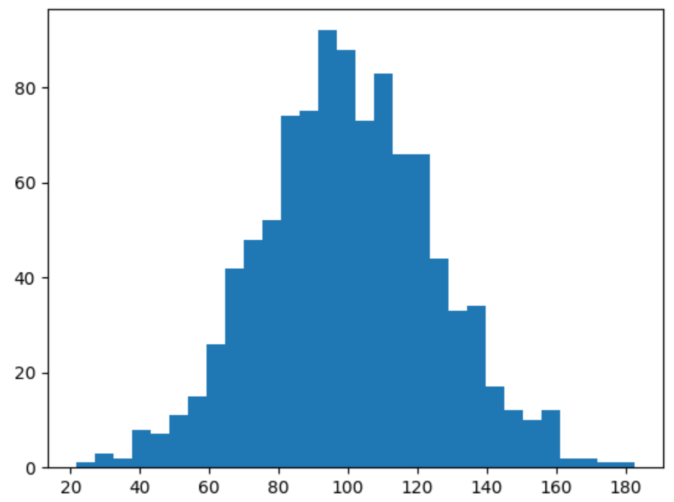


StatCan24 Average monthly egg prices per province for 2024.

Histogram

- Simply pass all of your data (x).
- Can optionally specify:
  - Number subdivisions (bins)

```
plt.hist(x, bins=10)
```

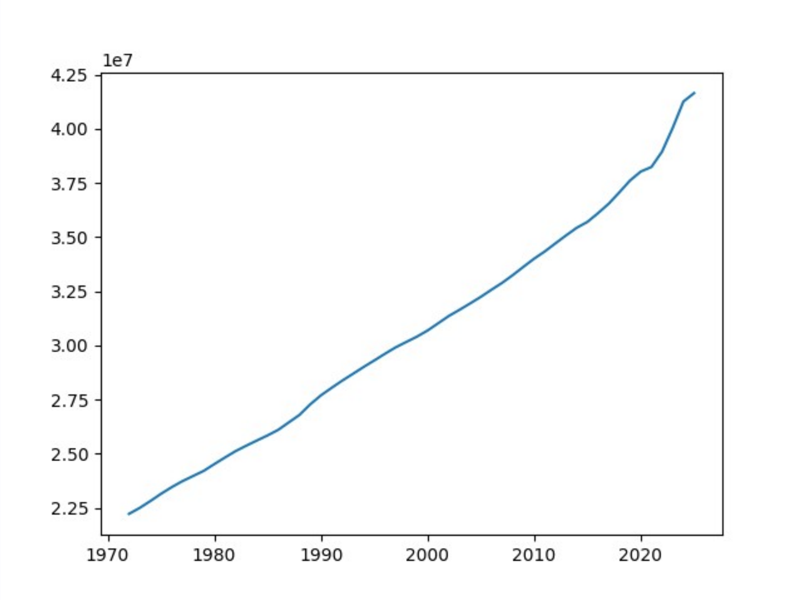


Histogram of a random distribution at mean 100 and std div of 25 with 30 subdivisions.

Line Charts

- Give your numerical x and y data.
- Can optionally pass:
  - Basic formating
  - To scale x or y (scalex, scaley)

```
plt.plot(x, y, [fmt]
   scalex=True, scaley=True)
```



Canada yearly population since StatCan.

[fmt]

- You first specify the colour, then the marker shape, then the line style.

r	red
g	green
b	blue
c	cyan
m	magenta
y	yellow
k	black
w	white

o	circle
*	star
.	point
,	pixel
x	x
+	plus
s	square
d	diamond

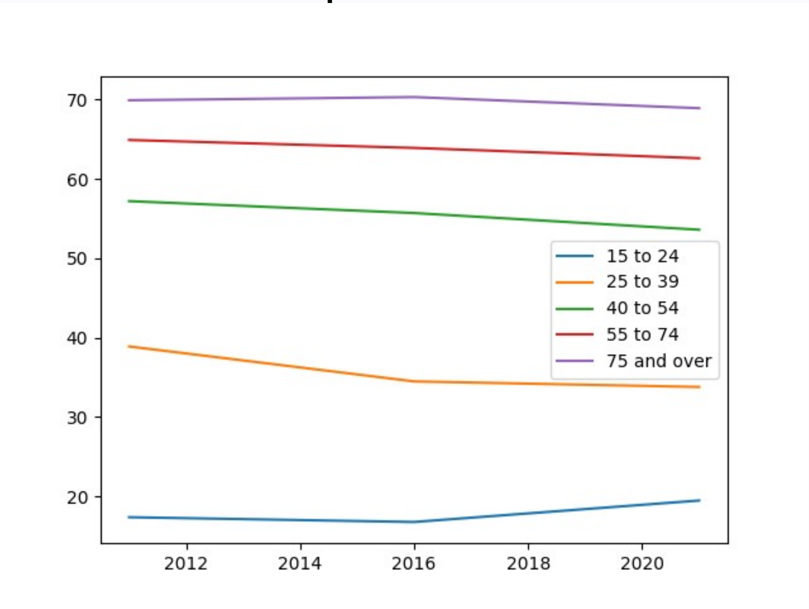
-	solid
:	dotted
--	dashed
-.	dotted-dashed

Examples:
- “bo”
- “c*--”

Multi-Line Charts

- Can call plot multiple times to place multiple lines on same chart.
- You can specify legend's label with label.

```
plt.plot(x1, y1, label="ax1")
plt.plot(x2, y2, label="ax2")
plt.plot(x3, y3, label="ax3")
```

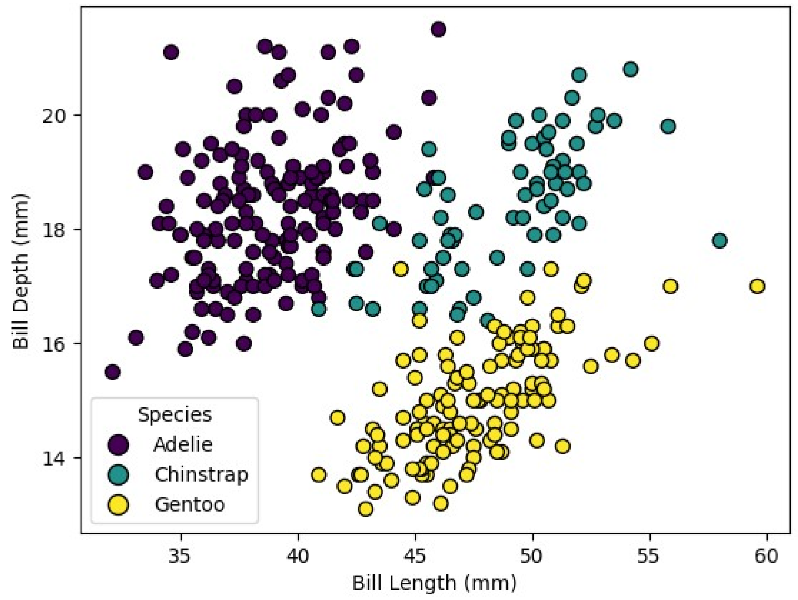


StatCan houshold ownership rates by age group 2011, 2016, and 2021.

Scatterplot

- Pass in the x and y.
- Can optionally specify:
  - Colour each gets (c) as a 1D collection for each entry.

```
plt.scatter(x, y, c=colours)
```



3 different peinguin samples Bill Length vs Bill Depth, Palmer Peinguins.

Use seaborn for complex viz

- Built on-top of matplotlib and allows for complex and prettier visualisations.

```
pip install seaborn
```

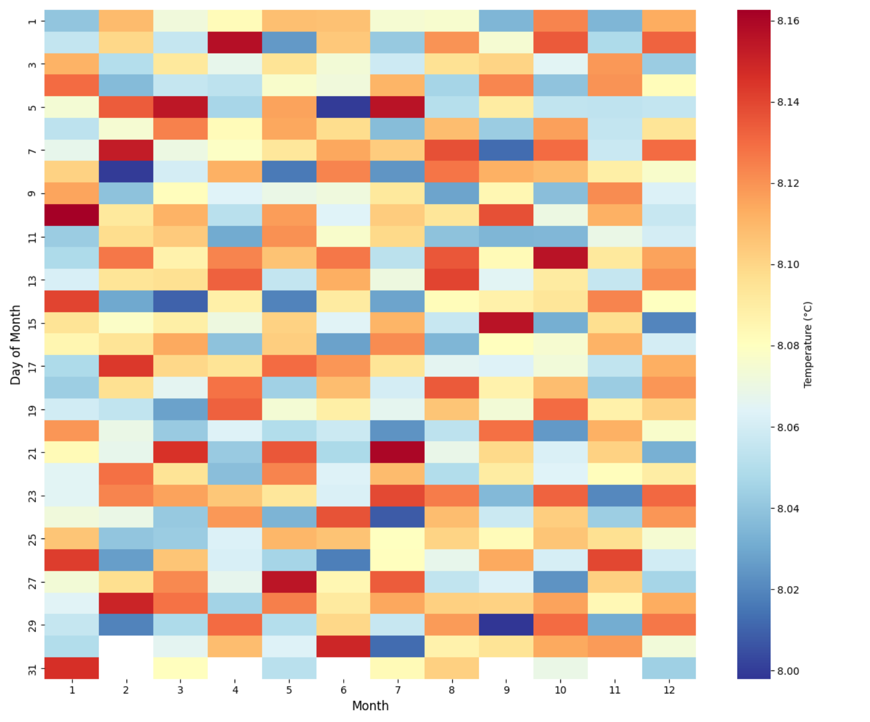
- Can then import it.

```
import seaborn as sns
```

Heatmap

- You pass a pandas DataFrame.
- Many optional arguments.

```
sns.heatmap(dataframe)
```

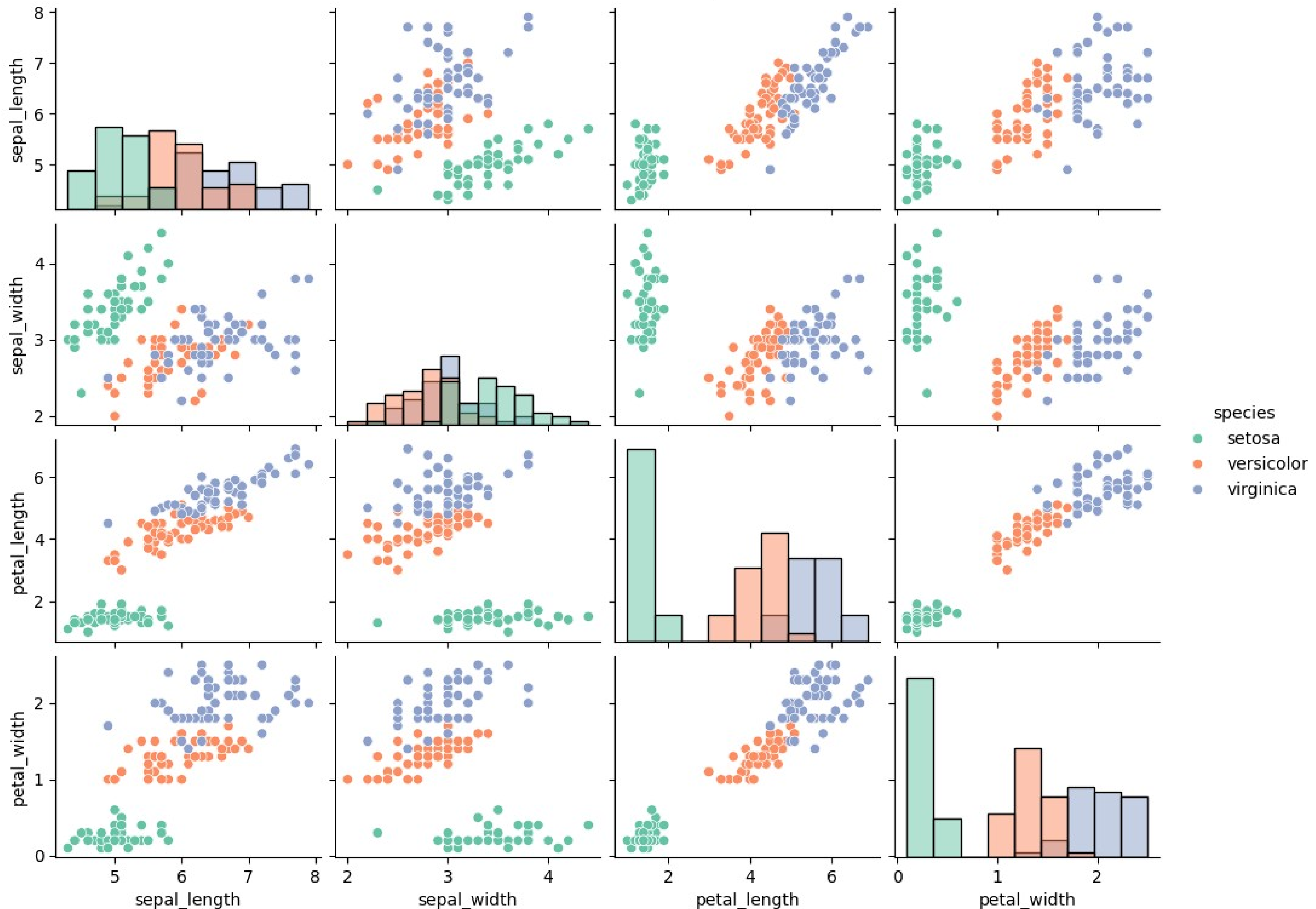


Average daily sea temperatures for each month sector 11417 off coast Vancouver 1.0m depth, ERDDAP

Pair Plot

- Pass in pandas DataFrame as first argument.
- Specify which variable should change colours (hue).
- Can optionally specify:
  - palette: seaborn colour pallete
  - vars: limit which variables to use (otherwise all)
  - diag_kind: “auto”, “hist”, “kde”, None

```
sns.pairplot(data,
    hue=hue,
    palette=palette,
    vars=vars,
    diag_kind="auto")
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

Let’s practice

In Canvas you will find a zipped file which has the energy production data of **each** of Canada’s 10 provinces per month for the year 2024. Your job is to ***tell a story***.