420-SN1-RE Programming in Science - Lab Exercise 18

October 30, 2024

Introduction

In this lab you'll learn more about using `matplotlib` and `pandas` together. Goals for this lab:

- Create a file summarizing `pandas` data.
- Create and use histograms.
- Plot data with `matplotlib`
- Detect outliers.

Files distributed with this lab:

- `lab18.pdf` - These lab instructions.
- `abalone.csv` - Data for this lab.

For this exercise you'll use a new CSV data file, `abalone.csv`. This file contains 4177 rows, each representing measurements of the shellfish abalone. The data includes seven columns:

Name	Type	Comments
sex	`str`	M, F, or I (infant)
length_mm	`int`	Shell length in millimeters
diameter_mm	`int`	Shell diameter in millimeters
height_mm	`int`	Shell height in millimeters
whole_weight_gm	`float`	Total weight in grams
shell_weight_gm	`float`	Shell weight in grams
rings	`int`	Number of growth rings in shell, indicates age

You can get more information at this website: https://archive.ics.uci.edu/dataset/1/abalone.

Exercise 0

For this part, you do *not* have to submit anything. You can do all of the work in the IDLE shell, but be careful that you are working in the correct folder!

1. Import the `pandas` package and read the `abalone.csv` file into a `DataFrame`.

   ```
   import pandas as pd
   df = pd.read_csv('abalone.csv')
   ```

2. Print the `columns` attribute of the `DataFrame`. Note the names of the columns.

   ```
   print(df.columns)
   ```

3. You can print a brief statistical summary of your numeric data with the `describe()` function:

   ```
   print(df.describe())
   ```

What happens to the string data?

4. Remember that a row or column of a `DataFrame` is called `Series`. A `Series` is sort of like a standard Python `list`, but with many additional functions. You'll explore some of the details of the `Series` object now. A `Series` has a length just like a list and can use many of the same

```
print(len(df.length_mm))
print(min(df.length_mm))
print(max(df.length_mm))
print(sum(df.length_mm))
```

5. Remember the `unique()` function that you used previously. Recall what it does:

```
print(df.sex.unique())
print(df.rings.unique())
```

6. While you now know how to import the `mean()` function from the `statistics` module, much of the same statistical functionality is available directly in `pandas`. You can compute the mean of the `length_mm` as follows:

```
print(df.length_mm.mean())
```

7. There are many other similar functions available:

```
print(df.length_mm.min())
print(df.length_mm.max())
print(df.length_mm.median())
print(df.length_mm.std()) # standard deviation
```

8. One useful feature of the `pandas Series` type is the support of *vectorized* operations. This gives a very intuitive way to work with lists:

```
x = df.height_mm
z = (x - x.mean()) / x.std()
```

   Print the value `z`. Notice what has happened. You have computed the *z-score* for each value in the `Series`. Vectorized operations are not possible with *standard* Python lists. It can be very useful when it is made available through packages like `pandas`, as long as you understand the difference.

9. Now that you've computed the z-scores for the height of the abalone shells, you can use the z-score to test for *outliers* on a data set. A common definition of an outlier is any point whose value is more than three standard deviations from the mean, which translates to a z-score with an absolute value greater than three:

```
print(x[z.abs() > 3])
print(df[z.abs() > 3])
```

   How many outliers do you identify with respect to the height measurement?

10. `pandas` has some useful features that can automatically create charts and graphs using `matplotlib`. For example you can create a histogram of the entire `DataFrame`:

```
import matplotlib.pyplot as plt
df.hist()
plt.show()
```

This automatically produces a histogram that divides the measurements into ten "bins" to illustrate the distribution of the values of each numeric column.

Can you use the histogram to help identify the outliers of the dataset?

Exercise 1

For this exercise you'll use the new CSV data file, `abalone.csv`. to explore a few additional features of `pandas`. Your goal will be to create a small text file that reports some basic statistics about the data.

You'll want to create a Python file, `lab18ex1.py` to contain the code to implement this part of the lab. Your program should write a text file, `report.txt`, which will contain the following summary of the data:

```
sex count length height weight rings
M    1528  112.3   30.3  198.3  10.7
F    1307  115.8   31.6  209.3  11.1
I    1342   85.5   21.6   86.3   7.9
```

You want to subdivide the data by sex, and use those divisions to compute the number of records of each sex, the mean shell length, the mean shell height, the mean whole weight, and the mean number of rings.

Be sure to format the data neatly using f-strings.

Exercise 2

The original purpose of the abalone data was to test whether some measurement could predict the age of the animal as effectively as counting the growth rings of the shell. Counting the growth rings is labour-intensive and tedious, but it gives a reliable estimate of the animals age. Scientists were trying to determine if another quantity, or mixture of quantities, could be used to predict age more easily.

To explore how easy this might be, create a single `matplotlib` figure using scatter plots to show the relationship between these measurements and the number of shell rings. the shell length, the shell height, the shell diameter, and the shell weight, and the number of rings. Create another Python file, `lab18ex2.py` to contain the code to implement this part of the lab.

Note that, with over 4000 data points, your scatter plot will be a little "crowded". It might be easier to see detail if you use smaller markers for each data item. You can do this by specifying an optional third argument to `plt.scatter()` that specifies the size of the marker.

You can also specify a keyword argument to give the desired colour. Your call might look like this:

```
plt.scatter(x, y, 2.0, color='violet')
```

Try to make creative use of loops and other tricks to avoid repeating code more than necessary to make your four subplots.

Your final chart should closely resemble Figure 1

Once again, you can use your scatter plots to look for obvious outliers in your data set.

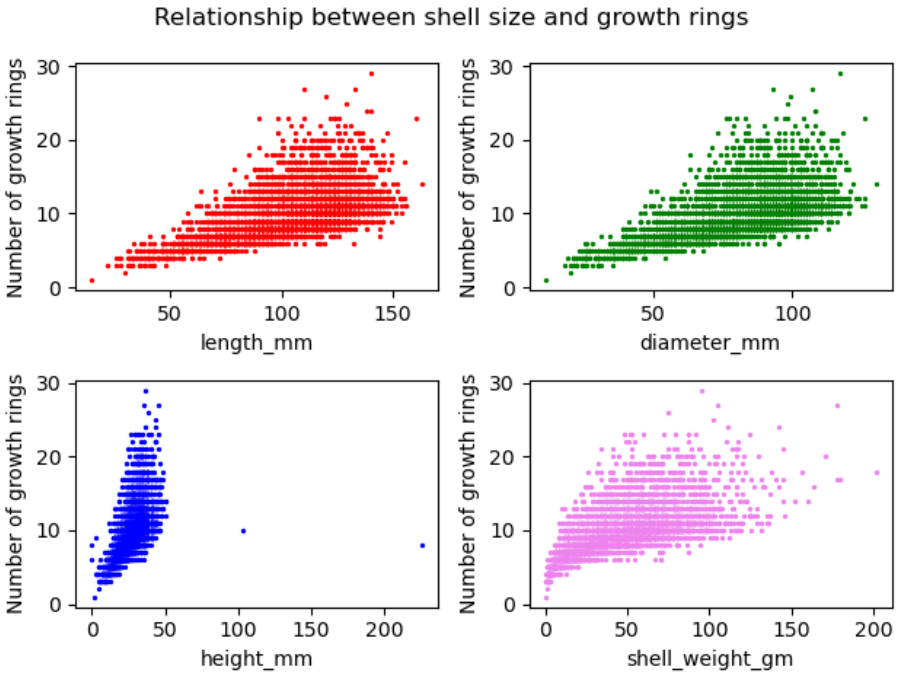

Figure 1: The finished scatter plot

Submission requirements

Combine your Python files into a single zip file and upload them to Omnivox. Be sure to submit a *single* file containing all of your work.