Doctoral Academy Python — Worksheet 3

12th June 2018

Overview

This final lab provides complete code examples for generating a variety of plots. We will again be using the "*Road Safety - Digital Breath Test Data 2011*" dataset. This can be obtained directly from *data.gov.uk* (see here).

These examples summarise the plots you have covered in this course's lab sessions. These examples do not include advanced plotting functionality or detailed explanation of the code. Please refer to the previous worksheets where these plotting tools were introduced for further explanation.

Before trying each of these examples you should import the relevant modules and load the dataset:

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

breath_data = pd.read_csv('DigitalBreathTestData2011.csv')
```

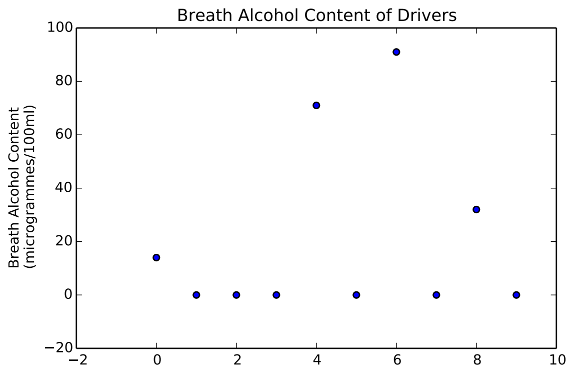
Aside from the above code, the plot examples on the following pages are self-contained and do not depend on any previous cells.

1 Plotting examples

Scatter plot

Build a scatter plot of breath-alcohol content for 10 breath tests:

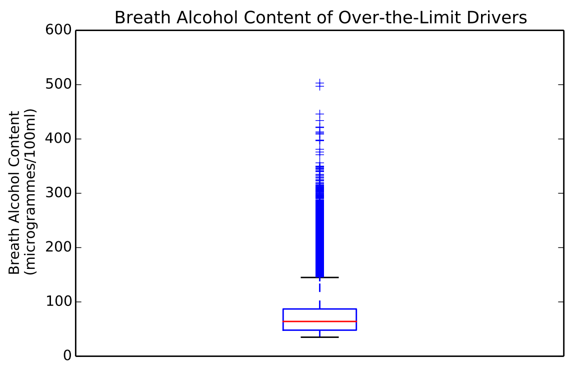
```
#
# Data processing...

breath_alcohol = breath_data['BreathAlcoholLevel(microg/100ml)']

# For this example, let's only plot the first 10 breath tests:
breath_alcohol = breath_alcohol[:10]

indx = np.arange(len(breath_alcohol))

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
ax.scatter(indx, breath_alcohol)

ax.set_ylabel('Breath Alcohol Content\n(microgrammes/100ml)')
ax.set_title('Breath Alcohol Content of Drivers')
plt.show()
```



Box plot

Build a box plot depicting the breath-alcohol content of drivers who were over the drink-drive limit:

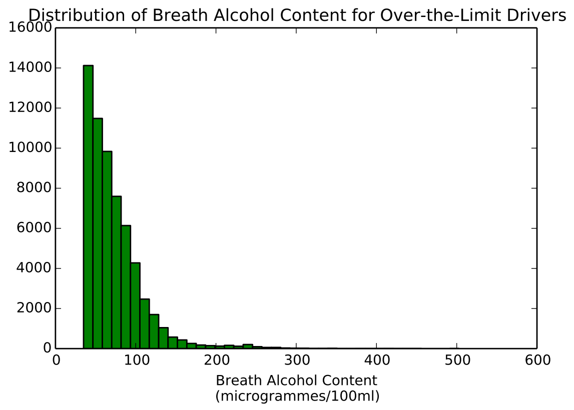
```
#
# Data processing...

breath_alcohol = breath_data['BreathAlcoholLevel(microg/100ml)']

# Let's only look at breath test results between 35 and 1000:
breath_alcohol = breath_alcohol[(35 <= breath_alcohol) & (breath_alcohol <= 1000)]

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
ax.boxplot([breath_alcohol])

ax.set_ylabel('Breath Alcohol Content\n(microgrammes/100ml)')
ax.set_title('Breath Alcohol Content of Over-the-Limit Drivers')
ax.set_xticks([])
plt.show()
```



Histogram

Build a histogram showing the distribution of drivers who are over the drink-drive limit:

```
#
# Data processing...

breath_alcohol = breath_data['BreathAlcoholLevel(microg/100ml)']

# Let's only look at breath test results between 35 and 1000:
breath_alcohol = breath_alcohol[(35 <= breath_alcohol) & (breath_alcohol <= 1000)]


#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])
ax.hist([breath_alcohol], bins=40, normed=False, facecolor='green')

ax.set_xlabel('Breath Alcohol Content\n(microgrammes/100ml)')
ax.set_title('Distribution of Breath Alcohol Content for Over-the-Limit Drivers')
plt.show()
```

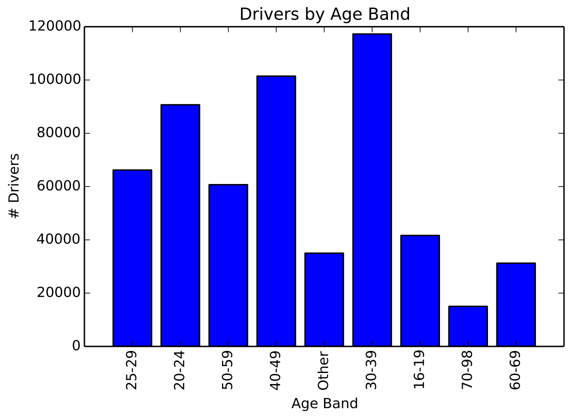


Bar chart

The two following examples demonstrate two different ways of building a bar chart representing the number of drivers in each age band.

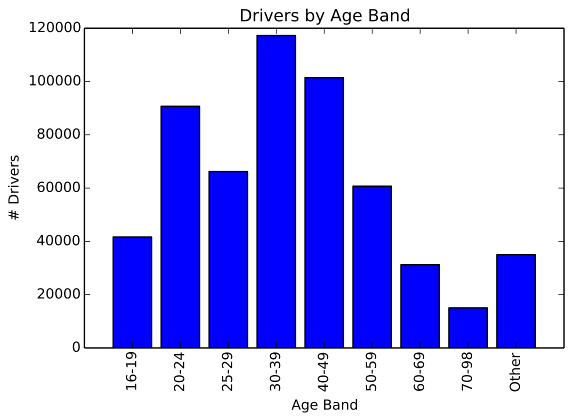
This example uses a `for` loop to count the number of drivers in each age band:

```
#
# Data processing...

ages = breath_data['AgeBand']

counts = []
labels = []
for age_band in ages.unique():
    rows = ages[ages == age_band]
    age_count = len(rows)  # number of people in this age band

    counts.append(age_count)
    labels.append(age_band)

indexes = np.arange(len(counts))

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.bar(indexes, counts, align='center')

ax.set_title('Drivers by Age Band')
ax.set_ylabel('# Drivers')
ax.set_xlabel('Age Band')
ax.set_xticks(indexes)
ax.set_xticklabels(labels, rotation='vertical')
plt.show()
```



This example uses a `pandas`'s `value_counts` method to count the number of drivers in each age band, and also re-orders the age bands so they appear in ascending order:

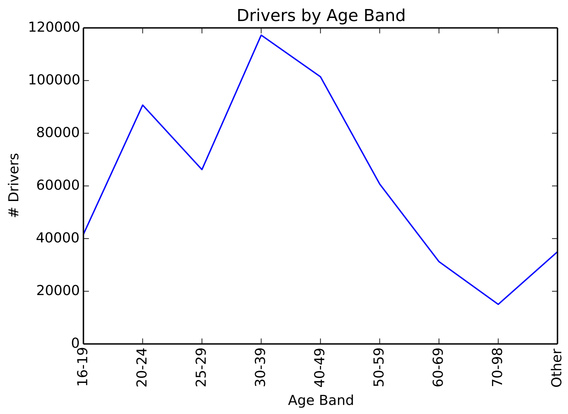
```
#
# Data processing...

ages = breath_data['AgeBand']

value_counts = ages.value_counts()

# Here we manually re-order the age bands so that they're ascending:
value_counts = value_counts.reindex(['16-19', '20-24', '25-29', '30-39', '40-49', '50-59',
    '60-69', '70-98', 'Other'])

counts = value_counts.values
labels = value_counts.keys()

indexes = np.arange(len(counts))

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.bar(indexes, counts, align='center')

ax.set_title('Drivers by Age Band')
ax.set_ylabel('# Drivers')
ax.set_xlabel('Age Band')
ax.set_xticks(indexes)
ax.set_xticklabels(labels, rotation='vertical')
plt.show()
```



Line plot

The following example presents the number of drivers in each age band as a line plot:

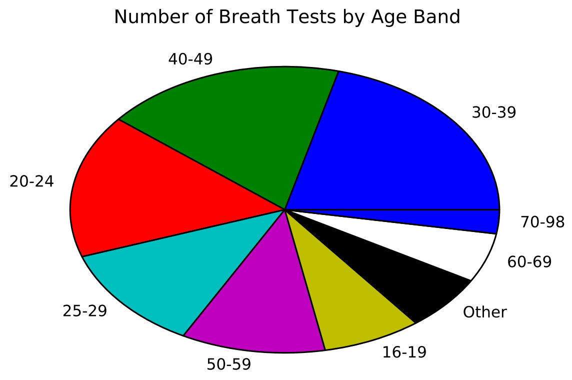
```
#
# Data processing...

ages = breath_data['AgeBand']

value_counts = ages.value_counts()

# Here we manually re-order the age bands so that they're ascending:
value_counts = value_counts.reindex(['16-19', '20-24', '25-29', '30-39', '40-49', '50-59',
    '60-69', '70-98', 'Other'])

counts = value_counts.values
labels = value_counts.keys()

indexes = np.arange(len(counts))

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.plot(indexes, counts)

ax.set_title('Drivers by Age Band')
ax.set_ylabel('# Drivers')
ax.set_xlabel('Age Band')
ax.set_xticks(indexes)
ax.set_xticklabels(labels, rotation='vertical')
plt.show()
```



Pie chart

The following example presents the number of drivers in each age band as a pie chart:

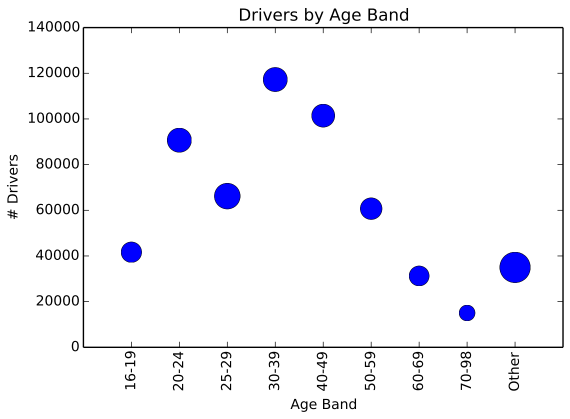
```python
#
# Data processing...

ages = breath_data['AgeBand']
value_counts = ages.value_counts()

counts = value_counts.values
labels = value_counts.keys()

#
# Plotting...

fig = plt.figure()
ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.pie( counts, labels=labels )
ax.set_title('Number of Breath Tests by Age Band')
plt.show()
```



Bubble chart

This example shows how the `scatter` plot function can be used to build a bubble chart. Each bubble will represent a particular age band (x axis), the number of drivers in that age band (y axis), and the average breath-alcohol content in that age band (bubble size).

```python
#
# Data processing...

ages = breath_data['AgeBand']
breath_alcohol = breath_data['BreathAlcoholLevel(microg/100ml)']

# Get the count of drivers in each age band:
value_counts = ages.value_counts()

# Here we manually re-order the age bands so that they're ascending:
value_counts = value_counts.reindex(['16-19', '20-24', '25-29', '30-39', '40-49', '50-59',
    '60-69', '70-98', 'Other'])

# Now get the count of the number of drivers in each age band:
counts = value_counts.values
labels = value_counts.keys()

# Here we decide the bubble sizes. There'll be one bubble per
# age band. The bubble size will be proportional to the average
# alcohol level for each age band
bubble_sizes = []
for label in labels:
    rows = breath_alcohol[breath_data['AgeBand'] == label]
    average_BAC = rows.mean()

    # Exaggerate the size of the bubble by a factor of 30
    # (otherwise they'll be too small)
    bubble_size = average_BAC * 30.0

    bubble_sizes.append(bubble_size)

indexes = np.arange(len(counts))

#
# Plotting...

fig = plt.figure()

ax = fig.add_axes([0.1, 0.1, 0.8, 0.8])

ax.scatter(indexes, counts, s=bubble_sizes, linewidth=0)

ax.set_title('Drivers by Age Band')
ax.set_ylabel('# Drivers')
ax.set_xlabel('Age Band')
ax.set_xticks(indexes)
ax.set_xticklabels(labels, rotation='vertical')
plt.show()
```



Drivers by Age Band
# Drivers
Age Band
0
20000
40000
60000
80000
100000
120000
140000
16-19
20-24
25-29
30-39
40-49
50-59
60-69
70-98
Other

2 Handling figures

The width and height of a figure can be set using the `figsize` parameter. This parameter should be passed in to the `plt.figure()` method. For example, to create a figure that is **5 inches wide** and **10 inches high**:

```
fig = plt.figure(figsize=[5, 10])
```

To save your figure to disk as an image file you can use the `savefig` method. For example, the following code would save the figure to a PNG file named 'myfigure.png':

```
fig.savefig('myfigure.png', bbox_inches='tight')
```

Here, `bbox_inches='tight'` is a useful special parameter that improves the border spacing around the figure. A variety of image file types are supported. PNG and PDF are recommended as they retain the original plot quality.