Lab 1 - Descriptive Statistics

January 24, 2020

1 Packages and Data Structures

The Python core distribution contains only the functions and methods essential for a general programming language. The power of Python lies in its modules and packages. These are collections of functions that can be imported into and used in your Python workspace. Throughout this module we will use many packages, the most frequent being NumPy, SciPy and Pandas. * **NumPy** allows for the creation of vectors, matrices and higher-dimensional arrays. It allows contains functions for performing operations on these objects, such as matrix multiplication and solving linear systems. * **SciPy** builds on the functionalities of NumPy, with modules for optimisation, statistics, numerical integration, differential equations and linear algebra. There is a lot of overlap between NumPy and SciPy. * **Pandas** is designed for elegant data manipulation and analysis. It allows for data to be structured as tables and contains functions for performing operations on these tables.

Packages need to be imported at the beginning of each script. Typically, we import packages with an abreviation so that we don't need to type out the full package name everytime we wish to use a function from the package.

[1]:
```
import numpy as np
import scipy as sp
import pandas as pd
```

NumPy *arrays* can be created from basic Python objects, such as lists, tuples and dicts, usung the `np.array()` funtion. If we perform a standard operation, such as addition or multiplication, to the array it will be applied to each element of the array

[2]:
```
a = [2,5,7,1,6,3]
vec = np.array(a)
print(vec+1)
print(vec*2)
print(vec[2:4])
```

```
[3 6 8 2 7 4]
[ 4 10 14  2 12  6]
[7 1]
```

As mentioned above Pandas provides data structures for elegant data manipulation and analysis. These data structures are known as *DataFrames*, and are similar to spreadsheets. The standard way to create a DataFrame is from a dict or by loading in a data file as a DataFrame. It is convenient to import the `DataFrame` function from Pandas to save using the `pd.` prefix every time we want to use it.

```
[3]: from pandas import DataFrame
     dict1 = {'Names':['Aoife','Brian','Catherine','Daniel','Eamonn','Fiona'],
              'Age':[23,45,17,64,57,32],'Height':[1.7,1.9,1.55,1.8,1.75,1.65],
              'Weight':[82,88,55,101,75,67]}
     df = DataFrame(dict1)
     print(df)
```

```
       Names  Age  Height  Weight
0      Aoife   23    1.70      82
1      Brian   45    1.90      88
2  Catherine   17    1.55      55
3     Daniel   64    1.80     101
4     Eamonn   57    1.75      75
5      Fiona   32    1.65      67
```

Columns are referenced by their heading and can be accessed using either `df.ColumnName` or `df['ColumnName']`, e.g. `df.Names` or `df['Names']` for the first column. You can also access elements in the DataFrame using their position with `df.iloc[a,b]`, where a and b are the row and column numbers, respectively, that you wish to access. Remember that indexing in Python starts at 0, so `df.iloc[4,2]` would return the entry in the 5th row and 3rd column, i.e. Eamonn's height 1.75.

```
[4]: print(df.iloc[3:5,:2])
```

```
    Names  Age
3  Daniel   64
4  Eamonn   57
```

2 Importing Data

There are manuy ways to import data in Python. We will predominantly use the Pandas `pd.read_csv()` function to load data files into Python as a DataFrame. Download the dataset `marks.csv` from Brightspace and save it into you current working directory. To useful functions for inspecting DataFrames are `.head()` and `.tail()`. They return the first 5 entries and last 5 entries, respectively.

```
[5]: data = pd.read_csv('marks.csv')
     print(data.head())          # printing all 365 entries would be ridiculous
```

```
  StudentID  Coursework  Project   Exam
0     34258        86.4       68  90.67
1    566811        77.4       55  90.67
2   6359256        15.5       66  40.00
3   6361307        83.3       85  92.00
4   6390081        85.2       90  93.33
```

3 Numerical summaries

Pandas DataFrames have lots of associated methods and function. If you type `data.` into your Python shell and hit the tab key you will get a list of possible completions, these are all of the methods that can applied to `data`, as well as the objects associated with it. `data.describe()` will produce a table with summary statistics for each of the columns in `data`.

[6]:
```
print(data.describe())
```

```
       Coursework    Project       Exam
count   51.000000  51.000000  51.000000
mean    76.807843  73.333333  70.535882
std     20.465921  13.387556  19.616650
min     15.500000  35.000000  24.000000
25%     65.550000  65.000000  53.330000
50%     85.200000  75.000000  74.670000
75%     91.300000  85.000000  86.670000
max     98.600000  95.000000  96.000000
```

We can compute these statistics individually with functions such as `mean()`, `std()`, `min()` etc.

[7]:
```
print(data.mean())
std_exam =  data['Exam'].std()
print('The standard deviation of the exam score was', std_exam,'%.')
max_proj = data['Project'].max()
max_proj_ID = data.loc[data['Project'].idxmax(),'StudentID']
print('The best project score was achieve by student', max_proj_ID,'with a␣
↪score of'
      , max_proj,'%.')
```

```
Coursework    76.807843
Project       73.333333
Exam          70.535882
dtype: float64
The standard deviation of the exam score was 19.616650088786376 %.
The best project score was achieve by student 6900976 with a score of 95 %.
```

4 Graphical summaries

The best plotting package in Python is Matplotlib, which contains functions for all of the plots we will consider in this lab session. For more examples and information on Matplotlib see the documentation page.

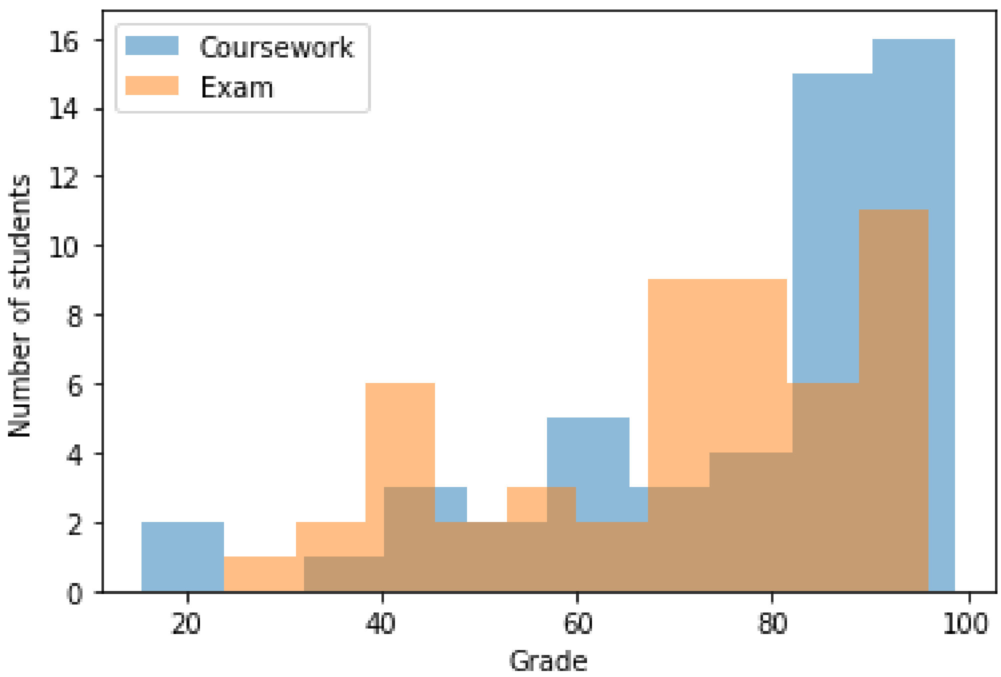
[8]:
```
import matplotlib.pyplot as plt

plt.figure()
plt.scatter(data.Coursework,data.Exam)
```

```
plt.xlabel('Coursework grade')
plt.ylabel('Exam grade')
plt.axis([0,100,0,100])
```

[8]: [0, 100, 0, 100]

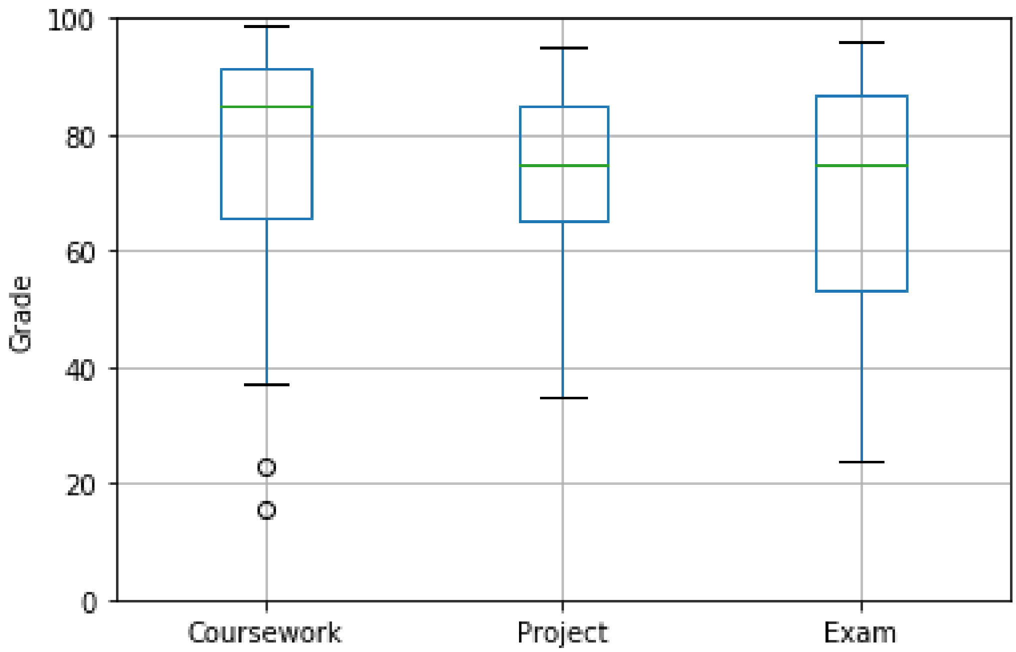
[9]:
```
plt.figure()
plt.hist(data.Coursework,bins=10,alpha=0.5,label='Coursework')
plt.hist(data.Exam,bins=10,alpha=0.5,label='Exam')
plt.xlabel('Grade')
plt.ylabel('Number of students')
plt.legend()
```

[9]: <matplotlib.legend.Legend at 0x1170d2790>



[10]:
```
plt.figure()
data.boxplot()
plt.ylabel('Grade')
plt.ylim(0,100)
```

[10]: (0, 100)



5 Exercises

1. Download the `weather2019.csv` dataset from Brightspace and save it into you current working directory.
2. Start a **new** Jupyter notebook, import the necessary packages and load in the `weather2019.csv` dataset.
3. Go to the *Lab 1* Quiz on Brightspace to find your questions for this week. Note that your questions will be different to other students.
4. You will need to write Python code to answer the questions. This code should be submitted to the *Lab 1* Assignment object on Brightspace in addition to completing the Quiz.

Note: You should not submit your Quiz attempt unless you are completely finished. Starting a new attempt will result in a new set of questions. You can save your current attempt and come back to it later if you need to.

The deadline for submitting your Quiz attempt and Python file is **Monday February 3rd at noon**.

6 Additonal non-assessed exercises

1. Download the `MichelsonMorley.csv` dataset from Brightspace and save it into you current working directory. This is the dataset from Michelson and Morley's experiments on the speed of light.
2. Reproduce the histogram on slide 19 of the leture notes on Descriptive Statistics.

3. Interpret your plot. What do you learn about the data from it?
4. Describe what happens when you change the number of bins.
5. What is the mean measurement for the speed of light?
6. What is the standard deviation?
7. Explore the options associated with the histogram command (`plt.hist?`). Try to change the colour, transparancy, labels, etc of your plot.