CSSE1001/7030

Semester 2, 2015
Assignment 1
10 marks Due 3rd September, 2015, 9:30am

1 Introduction

Data Analytics is the science of examining raw data with the purpose of drawing conclusions about that information.
Some metrics that are commonly used to summarise data sets include: ***Mean***, ***Median***, ***Standard deviation from the mean***, ***Range***.

This assignment requires you to create a program capable of extracting these features from an arbitrary data set. The data to be used in this assignment will be provided in `csv` files that can be found on the course web page. Each data set contains a number of data subsets with each new subset appearing on a new line. The data subsets consist of comma separated values. The first (index 0) value is the name of the subset. All subsequent items are the data points (in floating point format) associated with that subset.

The following is a sample data subset and comes from the `animal_heights.csv` file:

```
Slow Loris, 22.1, 24.2, 27.5, 23.3, ...
```

For this assignment you need to write a text-based interactive program which can accept various commands and display appropriate statistical information in response to those commands..

1.1 The Top-Level Interface

`interact()` is the top-level function that invokes the text-based user interface. When `interact()` is entered a welcome message is printed and the user is prompted to enter a filename. After this filename has been entered the interface prompts for further commands with the prompt, `Command:`. This is illustrated below.

```
>>> interact()

Welcome to the Statistic Summariser

Please enter the data source file: animal_heights.csv

Command:
```

1.2 The required commands

The valid commands which can be entered at the `Command:` prompt are as follows:

- `summary` provides a statistical summary of all the datasets
- `sets d1 d2` provides a statistical summary of the data sets specified in d1, d2,
- `q` quits the program

- `comp d1 d2` displays a comparison between subset d1 and subset d2 (**for CSSE7030 students only**)

Detailed descriptions for these commands are given below.

1.2.1 Summary Command

The command

`summary`

displays a statistical summary of all subsets within the provided data file. Each summary should contain the number of points (count), mean, median, maximum, minimum and standard deviation. An example is given below.

```
Command: summary
Set Summaries

              Slow Loris     Ocelot         Tiger
Count:        50             41             68
Mean:         25.05          49.43          94.2
Median:       25.73          48.21          92.96
Minimum:      19.99          40.1           70.31
Maximum:      29.99          59.02          121.98
Std Dev:      3.02           5.12           13.91
```

1.2.2 Sets Command

The command

`sets` *[subset_index]*

displays the statistical summary for each subset for which an appropriate index is provided. This command can take a varying number of parameters, specifying the number of subset summaries to display. In this example, the *Slow Loris* subset corresponds to index 0, *Ocelot* to index 1 and *Tiger* to index 2. Two examples are given below.

```
Command: sets 0
Set Summaries

              Slow Loris
Count:        50
Mean:         25.05
Median:       25.73
Minimum:      19.99
Maximum:      29.99
Std Dev:      3.02


Command: sets 1 2
Set Summaries
```

```
               Ocelot          Tiger
Count:         41              68
Mean:          49.43           94.2
Median:        48.21           92.96
Minimum:       40.1            70.31
Maximum:       59.02           121.98
Std Dev:       5.12            13.91
```

1.2.3 Quit Command

The command

`q`

causes the program to quit.

1.2.4 Unknown Command

Entering an unknown command should cause an error message to appear, which should begin with `Unknown command:` and then repeat the invalid command back to the user.

This message should appear if the command is not one of the commands discussed above or if the command has the wrong number of arguments. This is the only error handling that is required for this assignment. You may assume that all functions you will need to write will be tested with sensible data.

```
Command: csse
Unknown command: csse

Command: comp 0
Unknown command: comp 0

Command: sets
Unknown command: sets
```

1.2.5 Comparison Command

The command

`comp` (**for CSSE7030 students only**)

displays a comparison between each subset. This comparison is a simplified version of a technique called ANOVA, which will be elaborated on later in the task sheet. Sample output is as below:

```
Command: comp 0 1
Comparison between Slow Loris and Ocelot

               Mean            Std Dev
Shared (S)     36.04           12.8
Slow Loris     S-10.98         3.02
Ocelot         S+13.39         5.12
```

2 Assignment Tasks

For each function that you write you need to provide a suitable comment giving a description, the type and any preconditions. You should use the triple-quote commenting style. CSSE7030 students are required to add extra features as described at the end of this section.

Note: You must not use any functions in the `statistics` library, or any other functions available in standard Python libraries, to calculate statistics values. This must only be done using Python's built-in functions (see: `https://docs.python.org/3/library/functions.html`).

2.1 Download files

The first task is to download `assign1.py`, `assign1_support.py` and the data files. The data files should all come with a `.csv` extension and will be structured as outlined above.

We suggest you create a folder in which to write your solution and put these files in that folder.

The file `assign1.py` is for your assignment. **Do not modify the support file or your assignment file outside the area provided for you to write your code**.

When you have completed your assignment you will submit the file `assign1.py` containing your solution to the assignment (see Section 4 for submission details).

2.2 Write the code

Finally, write your solution to the assignment making sure you have included suitable comments. There are several functions you need to write and these are described below. **Do not use global variables in your code.** We are not expecting you to deal with exceptions for this assignment and so we will not be testing your code with things like invalid date strings.

2.2.1 Commenting - 1 mark

Students must apply docstring comments to each of their functions according to the Commenting section of the course notes.

See `http://csse1001.uqcloud.net/notes/commenting`.

2.2.2 Load Data - 2 marks

`load_data(filename)` takes a string corresponding to the name of the data file and returns a list of tuples containing the subset name (as a string) and a list of floating point data values.

The following is an example of this function operating on the `animal_heights` data set.

```
>>> data = load_data(‘animal_heights.csv‘)
>>> data
[(’Slow Loris’, [21.72, 29.3, ..., 22.94, 29.68]),
(’Ocelot’, [57.51, 47.59, ..., 51.95, 54.54]),
(’Tiger’, [75.0, 82.43, ..., 78.28, 80.18])]
```

2.2.3 Get Ranges - 0.5 marks

`get_ranges(data)` takes a list of floating point numbers and returns a tuple containing two floats: the minimum and maximum values in the data set respectively.

```
>>> data = load_data('animal_heights.csv')
>>> data[0][1]
[21.72, 29.3, ..., 22.94, 29.68]
>>> get_ranges(data[0][1])
(19.99, 29.99)
```

2.2.4 Get Mean - 0.5 marks

`get_mean(data)` takes a list of data points and returns the mean of those points.

For example, using the Slow Loris subset as in 2.2.2:

```
>>> data = load_data('animal_heights.csv')
>>> get_mean(data[0][1])
25.0544
```

2.2.5 Get Median - 1 mark

`get_median(data)` takes a list of data points and returns the median (middle value) of that list. In the event that there are an even number of data points (and hence there is no single 'middle' point), the average of the two middle points should be returned.

```
>>> data = load_data('animal_heights.csv')
>>> get_median(data[0][1])
25.735
```

2.2.6 Get Standard Deviation - 1 mark

`get_std_dev(data)` takes a list of data points and returns the standard deviation of data points in that list about the mean. The standard deviation can be calculated using the following equation:

$$\sigma = \sqrt{\frac{1}{N}(\Sigma(x - \overline{x})^2)}$$

Where σ is the standard deviation, x is each data point and $\overline{x}$ is the mean of the set.

```
>>> data = load_data('animal_heights.csv')
>>> get_std_dev(data[0][1])
3.0183440228045577
```

2.2.7 Data Summary - 1 mark

`data_summary(data)` takes a list of tuples (as should be returned from load_data) and returns a list of tuples containing the summary statistics and name of each subset.

This function should caluculate and include the number of points in each subset in the result. Values should appear in the resulting tuples in the following order:

1. Subset name
2. Count
3. Mean
4. Median
5. Minimum
6. Maximum
7. Standard Deviation

An example of the `data_summary` function is shown below:

```
>>> data = load_data('animal_heights.csv')
>>> data_summary(data)
[('Slow Loris', 50, 25.0544, 25.735, 19.99, 29.99, 3.0183440228045577),
('Ocelot', 41, 49.42780487804878, 48.21, 40.1, 59.02, 5.117594000325868),
('Tiger', 68, 94.19852941176471, 92.96000000000001, 70.31, 121.98,
13.908963177063187)]
```

2.2.8 Display Data Set Summaries - 1 mark

`display_set_summaries(data)` displays the summary information for the supplied data set summaries (the same structure to that returned from the data_summary function). Note that this function returns `None`, but instead prints data to `stdout`.

The following is an example of this function:

```
>>> data = load_data('animal_heights.csv')
>>> result = display_set_summaries(data_summary(data))
Set Summaries

               Slow Loris     Ocelot         Tiger
Count:         50             41             68
Mean:          25.05          49.43          94.2
Median:        25.73          48.21          92.96
Minimum:       19.99          40.1           70.31
Maximum:       29.99          59.02          121.98
Std Dev:       3.02           5.12           13.91

>>> print(result)
None
>>>
```

2.2.9 The Top-Level Interface - 2 marks

interact() is the top-level function that defines the text-base user interface as described in section 1.2.

2.2.10 Hints

For string processing: `strip, split, partition`
For user interaction: `input`

2.3 Extra Task for CSSE7030 Students Only

The CSSE7030 students are required to add extra features to the program. (**Note:** CSSE1001 students will not gain credit for attempting the CSSE7030 task.) The added features consist of an extra command in interact, `comp`, and two extra functions.

2.3.1 Data Set Comparison - 1.5 marks

`data_comparison(data, set1, set2)` generates the data for the simplified one-way ANOVA mentioned earlier in the task sheet. The data should come in a tuple of tuples, in the format specified by the following pseudocode.

```
(
  (set_1_name,set_2_name),
  (shared_mean, set_1_mean-shared_mean, set_2_mean-shared_mean),
  (shared_std_dev,set_1_std_dev,set_2_std_dev)
)
```

The shared mean and shared standard deviation are the mean and standard deviation computed on a dataset made up of both of the two datasets being compared.

2.3.2 Display Comparison - 1.5 marks

`display_comparison(data)` takes the output from `data_comparison` and prints a table in the format below.

```
Comparison between Slow Loris and Ocelot

               Mean           Std Dev
Shared (S)     36.04          12.8
Slow Loris     S-10.98        3.02
Ocelot         S+13.39        5.12
```

It is important to note the `S-10.98` and the `S+13.39` – these are used to signify to the uninformed user that these values are relative to the shared mean.

3 Assessment and Marking Criteria

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in the practical session you have signed up to in week 7.

You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution
- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. A technically correct solution will not elicit a pass mark unless you can demonstrate that you understand its operation.

Criteria	**Mark**
Your code is mostly complete, correct, clear, succinct and well commented. You are able to explain your code.	8 - 10
Your code has some problems OR you have some problems explaining your code.	4 - 7
Your code is clearly incomplete, incorrect, too complex or hard to understand OR you have major problems explaining your code.	1 - 3
Your work has little or no academic merit.	0

The following table gives an indication of how the total marks for the assignment are distributed for each of the required functions and commenting. For example, CSSE1001 students would obtain 2 marks for correctly implementing the `load_data` function from 2.2.2 and for demonstrating proper understanding of their implementation. Partial marks will be given for incomplete implementations and/or insufficient understanding. CSSE7030 students will be marked out of 13 and then scaled down to 10.

Section/Function	CSSE1001 Weight	CSSE7030 Weight
2.2.1 Commenting	1	1
2.2.2 `load_data`	2	2
2.2.3 `get_ranges`	0.5	0.5
2.2.4 `get_mean`	0.5	0.5
2.2.5 `get_median`	1	1
2.2.6 `get_std_dev`	1	1
2.2.7 `data_summary`	1	1
2.2.8 `display_set_summaries`	1	1
2.2.9 `interact`	2	2
2.3.1 `data_comparison`	-	1.5
2.3.2 `display_comparison`	-	1.5
Total	**10**	**13**

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out that code and we will mark that. Please read the section in the course profile about plagiarism.

4 Assignment Submission

You must submit your completed assignment electronically through Blackboard. Please read
`http://www.library.uq.edu.au/ask-it/blackboard-assessment`
for information on submitting through Blackboard.

You should electronically submit your copy of the file `assign1.py` (use this name - all lower case). You may submit your assignment multiple times before the deadline - only the last submission will be marked. Late submission of the assignment will not be accepted. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on-time, you may submit a request for an extension. Requests for extensions should be made as soon as possible, and preferably before the assignment due date.

All requests for extension must be submitted on the UQ Application for Extension of Progressive Assessment form:
`http://www.uq.edu.au/myadvisor/forms/exams/`
`progressive-assessment-extension.pdf`
no later than 48 hours prior to the submission deadline. The application and supporting documentation (e.g. medical certificate) must be submitted to the ITEE Coursework Studies office (78-425) or by email to `enquiries@itee.uq.edu.au`. If submitted electronically, you must retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.