Arrays and files in Visual Basic I

Topics

- Arrays and parallel arrays
- Reading from CSV files into parallel arrays
- More practice with formatting strings

Preliminaries

If you haven't already done so, start by making a copy of today's lab folder (`Lab06_Arrays_files`) and saving it in your `M:\CS130\Labs` folder. Right-click on the folder you just copied and rename it `Lab06_YourLastName_YourFirstName` (but use your actual last and first names). You may now work locally by opening the write-up from within the copied folder.

TL;DR

Implement a program to predict the rating that a user will assign to a movie based on their ratings for other movies and the ratings of other users for the same movies.

Background

These lab exercises use a dataset that is maintained by the GroupLens research group at the University of Minnesota. The dataset is part of the MovieLens project. In their own words, MovieLens is a platform for "*Non-commercial, personalized movie recommendations*"; think Netflix recommendations, but not for profit. The dataset that we will be exploring consists of movie ratings for five popular movies from 22419 unique users. In the file `ML-latest.csv` each row constitutes the ratings for a single user for each of the five movies. For example, the first three rows in the file look like:

```
4.0,4.5,4.0,4.0,3.0
3.0,4.0,5.0,4.0,5.0
5.0,5.0,5.0,5.0,5.0
```

This means that user 0, rated movie 1, 4.0 stars out of 5 possible, movie 2, 4.5 stars, movie 3, 4.0 stars, and so on. Likewise for user 1 and 2. The title for each of the five movies is recorded in file `ML-titles.txt`.

Instructions

The objective for the lab is to make an accurate prediction of the rating that a user will assign to a movie based on their ratings for other movies and the ratings of other users for the same movies. The basic idea is to take a user's ratings for a set of movies and find other users with similar ratings. These similar users can then be used to inform our prediction.

In order to identify similar users, we must have some quantitative way to determine similarity, i.e., a *similarity measure*. A common way to do this is to calculate the absolute difference for each of the movie ratings for two users and add these together to compute a total difference measurement. This distance measure is known as the Manhattan Distance (a.k.a. Taxicab distance).

Take the first two users in the example above. The difference between them is calculated as:

```
  |4.0-3.0| + |4.5-4.0| + |4.0-5.0| + |4.0-4.0| + |3.0-5.0|
= 1.0 + 0.5 + 1.0 + 0.0 + 2.0
= 4.5
```

So we would say that the Manhattan distance between the two users is 4.5. It is important to note that in this case, *distance* is the opposite of *similarity*, which is what we are really after. So, to find the most similar users, we should find the users with that have the least distance between them.

Give it a try. Fill in the following table by computing the distance between the three users in the example above, then determine which two users are the most similar. Note, it is not necessary, or meaningful in this case, to compute the distance between a user and themselves, so each of these entries in the table is marked with `---`. You should also note that the Manhattan distance is *symmetric*, i.e., the distance between user *X* and user *Y* is the same as the distance between user *Y* and user *X*, so you only need to compute this distance once and then can fill in both cells in the table.

User	0	1	2
0	---		
1		---	
2			---

1. Launch the RAPTOR software and use it to **open** an existing program called `RatingPrediction_Part1.rap` inside of your lab folder.

   The program creates an array of user ratings for **ONE** movie, called `movie1` which currently stores three ratings, one for each of the three users above. Then, using the rating input by the user into variable `userRating`, the program is meant to compute the distance between the user input rating, `userRating`, and the ratings of the other three users, stored in the array `movie1`. The distance for between the input rating and the three user ratings in the array should be stored in a second array called `distance`. For instance, the first rating in `movie1[1]` is 4.0. If the user input the value 3.5, then `distance[1]` should have the value 0.5. However, the program is not completed yet!

   Currently, the program loops over all the ratings in `movie1` (until `index > length_Of(movie1)`) but does not compute any distances; you are to complete the program `RatingPrediction_Part1.rap` to make this happen. Note than your program must work correctly even if the ratings in `movie1` changes.

   After you get your program working, fill out `Test Table 1` in the file `RatingPredictions_TestTables.docx`, making sure that your expected output and actual output are consistent.

When done, save and close file `RatingPredictions_Part1.rap`.

Checkpoint 1 (30/100): A successful `RatingPredictions_Part1.rap` program:

- produces correct output for all test cases in Test Table 1

2. Launch VS Express 2013 and open the project `RatingPredictions_Part2` which can be found inside of your lab folder.

   Complete the project `RatingPredictions_Part2` so that it is equivalent to your RAPTOR program. Test your program by checking that it is producing correct results for all of the test cases in Test Table 1.

   Now, update your program so that instead of having ratings for each of the three users for a single movie (`movie1`), the ratings for all five of the movies are stored in separate arrays called: `movie1`, `movie2`, `movie3`, `movie4`, and `movie5`.

   Besides declaring and assigning values to the new arrays, this change will also require you to update your distance calculation to include the additional movie ratings. You do not need a loop to loop over the five movie arrays, you can assume that there will always be five movies from now on.

   After you get your program working, fill out `Test Table 2` in the file `RatingPredictions_TestTables.docx`, making sure that your expected output and actual output are consistent.

Your project should behave identically to the solution which can be run by double-clicking the file `RatingPredictions_Part2.exe` found in the `Executables` folder inside of your lab folder.

When done, save and close project `RatingPredictions_Part2`.

Checkpoint 2 (65/100): A successful `RatingPredictions_Part2` project:

- is based on a complete and correct `RatingPredictions_Part1.rap` program
- produces correct output for all test cases in Test Table 2

3. Use **File Explorer** to navigate to your lab folder. Copy and paste the folder `RatingPredictions_Part2`, and then right-click on the copy to rename it as `RatingPredictions_Part3`. Launch VS Express and use it to open the project `RatingPredictions_Part3`.

   Make the necessary changes to the project `RatingPredictions_Part3` so that instead of the three user ratings being assigned in the program, they are instead read from the comma-separated-value file, `ML-ratings-small.csv`. The rest of your program should not change. This file is provided for you in the project folder `RatingPredictions_Part3`, in a sub-folder called `Bin/Debug`.

   Test your program by checking that it is producing correct results for all of the test cases in Test Table 2.

Your project should behave identically to the solution which can be run by double-clicking the file `RatingPredictions_Part3.exe` found in the `Executables` folder inside of your lab folder.

When done, save and close project `RatingPredictions_Part3`.

Checkpoint 3 (70/100): A successful `RatingPredictions_Part3` project:

- produces correct output for all test cases in Test Table 2, using ratings input from the file `ML-ratings-small.csv`

4. Use **File Explorer** to navigate to your lab folder. Copy and paste the folder RatingPredictions_Part3, and then right-click on the copy to rename it as RatingPredictions_Part4. Launch VS Express and use it to open the project RatingPredictions_Part4.

   Next, open the file ML-ratings.csv and look at its contents. This is the full set of ratings for 22149 users for the five movies described in ML-titles.txt.

   Make the necessary changes to the project RatingPredictions_Part4 to read ratings from this file instead of ML-ratings-small.csv.

   After updating your project to input ratings from the full dataset file, change the output of the project to a table with the following format

```
User ID   Distance
------------------
    XXX        X.X
    XXX        X.X
      .          .
      .          .
      .          .
------------------
```

   Declare a format string for such a table as follows:

```
Dim fmtStr As String = "{0,7}    {1,7:0.0}" & vbNewLine
```

   which could then be used to format two variables like so:

```
String.Format(fmtStr, i, distances(i))
```

   Thus, in fmtStr, we say that there are two *format items*, one for each of the two columns in the table. Each format item is demarked by {...}. Within the curly braces, the first number indicates the *value index*, i.e., which value in the line String.Format(...) is to formatted. The first value after the format string, fmtStr in this case, is considered value 0. So, {0,7} is the format item for the value i, {1,7:0.0} for value distances(i), and so on if there were more format items. After the value index is the *alignment*, i.e., how wide the formatted column should be, as well as whether it should be left or right justified. So, {0,7} indicates that value i should be formatted as seven characters wide and right justified. You can see that in this format string, all format items are seven characters wide and right justified. Lastly, following the alignment is the *format specifier*, which indicates out a value should look when formatted. There are many possibilities for a format specifier, so we will not go into the details. Suffice it to say that the 0.0 format specifier in {1,7:0.0} means that value distances(i) should be formatted as a decimal value with a single digit following the decimal. If you would like more information on any of these topics, see this documentation from Microsoft.

Your project should behave identically to the solution which can be run by double-clicking the file RatingPredictions_Part4.exe found in the Executables folder inside of your lab folder.

When done, save and close project RatingPredictions_Part3.

Checkpoint 4 (75/100): A successful RatingPredictions_Part4 project:

- produces correctly formatted output for the ML-ratings.csv dataset — output should match EXACTLY the formatted specified in the example

Submission Instructions

During our next lab meeting, you will be asked to expand your solutions to Part 4. Prior to the meeting, you are expected to

- finish ALL parts of this lab,
- study (again) any material you struggled with in this lab, and
- study new material needed for the next lab

You will submit your work for this lab and the next one together at the end of our next lab meeting.