Using GitHub for scientific research

LEC Retreat 2015

Jon W. Carr • Sunday, 18th January 2015

Team 1 – Reading in a CSV file

Feel free to ask me questions if you get stuck!

Specification

Write a Python function that reads a two-columned CSV file and separates the two columns into two Python lists. The left column is a list of words in an artificial language, and the right column is a list of the corresponding meanings to which those words refer. The function should be called `ReadFile()`. It should take one argument – a filename – and it should return two lists.

Once you've accessed the repository, take a look at the example CSV file – `KCS2008_chain1_gen9.csv` – to see what the data looks like. *Continue reading below before you start any coding*.

Forking the repository

Search for the **LECretreat15** repository on the GitHub website. Open the page for the repository and click Fork 1. This will create a fork of the repository – i.e. a duplicate of the repo in your own GitHub account. Then, to copy the repository to your local machine, click Clone in Desktop in the right-hand sidebar. This will launch the GitHub app which will ask where to save the files.

Creating a new branch

Depending on the complexity of your project, you may not want to make changes directly to the repository. Instead, new functionality can be developed in a duplicate of the entire repo, which we call a branch. This allows you to have free rein over the entire repo without impacting other ongoing work. It also means that the master version of the repo remains in a known stable condition. Once the code in the separate branch is ready for use, the branched code is merged back into the master.

In the GitHub app, go to the Branches tab at the top of the window. Click the small + button on the master branch. This allows you to create a new branch based on the current state of the master branch. Give your new branch a name – something like **ReadFile**. Once you've created your new branch, it should automatically become the "Current branch" you're working in. Any changes you make to the files local to your machine will now be committed to this branch.

Writing your code

Open `MantelTest.py` in an editor of your choice. Ideally, use an editor that's not going to fiddle with the formatting of the file, e.g. changing tab lengths. Sublime Text or Atom are good choices, but you could also use IDLE or a command line text editor such as Nano or Emacs if you're that way inclined.

In the file, you'll find that I've left some space for you to insert your function. Avoid editing any other parts of the code – doing so might make it tricky for git to resolve conflicts with other teams' changes. Take a quick look over the rest of the code to get an idea of how the module is designed to work (it may

not be obvious until all the teams have contributed their code). Then add your function according to the specification given above and the tips below.

Tips for writing your function

Feel free to use online resources and to write your function however you like. If you're not too sure, here's one way you might approach the task:

- Define the name of the function and the argument it needs to take
- Use `open()` and `read()` to get the file contents into a string
- Initialize two empty lists
- Split the file contents at the line breaks (`\n`)
- Iterate over lines and split each line at the comma
- Put each part of the line into its respective list
- Return the two lists

Test your code

Run `python` or `idle` from the command line, or launch another Python IDE of your choosing. Import the `MantelTest` module. Although you won't be able to use the full module yet, you should be able to test your function in isolation. Does it work as prescribed in the spec?

Commit your code to the branch

Switch over to the GitHub app and click on the Changes tab at the top of the window. It should have automatically detected that you've made changes to `MantelTest.py` and should highlight any new code in green and any deleted stuff in red. Check that you're happy with your new code and add a summary and description. Then click the 

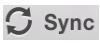
button. Your commit will show up under Unsynced Commits.

Merge your branch back into the master

Switch to the Branches tab and click 

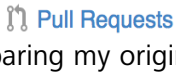
. Drag and drop your new and master branches into the relevant boxes and click Merge Branches. The commit you made to the new branch will now be merged into the master branch. Switch into the master by clicking the  button on the master branch and then Switch To This Branch. Master should now be your "Current branch".

Syncing with GitHub

Click 

in the top right corner of the app. The changes you've made to the repo on your machine will be pushed up to your fork of the repo on GitHub.

Create a pull request

The final step is to ask the owner of the original repo (me) to pull in your changes. You do this by creating a "pull request". Switch over to the GitHub website and reload the page. Click 

in the right-hand sidebar and then 

. GitHub should automatically suggest comparing my original LECretreat15 repo with your fork of the repo and should highlight the differences between them. Click the 

button and write a title and description for your request. Usually, you'll want to explain to the repo owner why they should incorporate your code. Click 

.

Voila!

Using GitHub for scientific research

LEC Retreat 2015

Jon W. Carr • Sunday, 18th January 2015

Team 2 – Calculating pairwise distances

Feel free to ask me questions if you get stuck!

Specification

Write a Python function that takes a list of strings and computes the Levenshtein edit-distance for each possible pairing of strings. If you take 10 input strings, you should output a list of 45 values, i.e. one value for each possible pairing of the 10 strings. The function should be called `PairwiseDistances()`. It should take one argument (a list) and it should return a list.

There is a Levenshtein function already included in the code – `LevenshteinDistance()` – so you won't need to write this part yourself. *Continue reading below before you start any coding*.

Accessing the repository

Search for the **LECretreat15** repository on the GitHub website. Open the page for the repository and click Fork 1. This will create a fork of the repository – i.e. a duplicate of the repo in your own GitHub account. Then, to copy the repository to your local machine, click Clone in Desktop in the right-hand sidebar. This will launch the GitHub app which will ask where to save the files.

Creating a new branch

Depending on the complexity of your project, you may not want to make changes directly to the repository. Instead, new functionality can be developed in a duplicate of the entire repo, which we call a branch. This allows you to have free rein over the entire repo without impacting other ongoing work. It also means that the master version of the repo remains in a known stable condition. Once the code in the separate branch is ready for use, the branched code is merged back into the master.

In the GitHub app, go to the Branches tab at the top of the window. Click the small + button on the master branch. This allows you to create a new branch based on the current state of the master branch. Give your new branch a name – something like **PairwiseDistances**. Once you've created your new branch, it should automatically become the "Current branch" you're working in. Any changes you make to the files local to your machine will now be committed to this branch.

Writing your code

Open `MantelTest.py` in an editor of your choice. Ideally, use an editor that's not going to fiddle with the formatting of the file, e.g. changing tab lengths. Sublime Text or Atom are good choices, but you could also use IDLE or a command line text editor such as Nano or Emacs if you're that way inclined.

In the file, you'll find that I've left some space for you to insert your function. Avoid editing any other parts of the code – doing so might make it tricky for git to resolve conflicts with other teams' changes. Take a quick look over the rest of the code to get an idea of how the module is designed to work (it may

not be obvious until all the teams have contributed their code). Then add your function according to the specification given above and the tips below.

Tips for writing your function

Feel free to use online resources and to write your function however you like. If you're not too sure, here's one way you might approach the task:

- Define the name of the function and the argument it needs to take
- Figure out the length of the input list, which we could call *n*
- Initialize an empty list for the pairwise distances
- Iterate over the numbers from 0 to *n*–1 as *i*
- Iterate over the numbers from *i*+1 to *n* as *j*
- Get the distance between string *i* and string *j* and add it to your list of pairwise distances
- Return the list of pairwise distances

Test your code

Run `python` or `idle` from the command line, or launch another Python IDE of your choosing. Import the `MantelTest` module. Although you won't be able to use the full module yet, you should be able to test your function in isolation. Does it work as prescribed in the spec?

Commit your code to the branch

Switch over to the GitHub app and click on the Changes tab at the top of the window. It should have automatically detected that you've made changes to `MantelTest.py` and should highlight any new code in green and any deleted stuff in red. Check that you're happy with your new code and add a summary and description. Then click the Commit button. Your commit will show up under Unsynced Commits.

Merge your branch back into the master

Switch to the Branches tab and click Merge View. Drag and drop your new and master branches into the relevant boxes and click Merge Branches. The commit you made to the new branch will now be merged into the master branch. Switch into the master by clicking the button on the master branch and then Switch To This Branch. Master should now be your "Current branch".

Syncing with GitHub

Click Sync in the top right corner of the app. The changes you've made to the repo on your machine will be pushed up to your fork of the repo on GitHub.

Create a pull request

The final step is to ask the owner of the original repo (me) to pull in your changes. You do this by creating a "pull request". Switch over to the GitHub website and reload the page. Click Pull Requests in the right-hand sidebar and then New pull request. GitHub should automatically suggest comparing my original LECretreat15 repo with your fork of the repo and should highlight the differences between them. Click the Create pull request button and write a title and description for your request. Usually, you'll want to explain to the repo owner why they should incorporate your code. Click Create pull request.

Voila!

Using GitHub for scientific research

LEC Retreat 2015

Jon W. Carr • Sunday, 18th January 2015

Team 3 – The Monte Carlo loop

Feel free to ask me questions if you get stuck!

Specification

Write a Python function called `MonteCarlo()` that takes three arguments: two lists and a number of randomizations. Run a loop for the number of randomizations. On each iteration of the loop, shuffle one of the input lists (using the included `ShuffleDistances()` function) and then compute the correlation between the two lists. Return the mean and standard deviation of all the correlation coefficients.

The SciPy stats module has already been imported, so you can use `stats.pearsonr()` to calculate the correlation coefficient. *Continue reading below before you start any coding*.

Accessing the repository

Search for the **LECretreat15** repository on the GitHub website. Open the page for the repository and click Fork 1. This will create a fork of the repository – i.e. a duplicate of the repo in your own GitHub account. Then, to copy the repository to your local machine, click Clone in Desktop in the right-hand sidebar. This will launch the GitHub app which will ask where to save the files.

Creating a new branch

Depending on the complexity of your project, you may not want to make changes directly to the repository. Instead, new functionality can be developed in a duplicate of the entire repo, which we call a branch. This allows you to have free rein over the entire repo without impacting other ongoing work. It also means that the master version of the repo remains in a known stable condition. Once the code in the separate branch is ready for use, the branched code is merged back into the master.

In the GitHub app, go to the Branches tab at the top of the window. Click the small + button on the master branch. This allows you to create a new branch based on the current state of the master branch. Give your new branch a name – something like **MonteCarlo**. Once you've created your new branch, it should automatically become the "Current branch" you're working in. Any changes you make to the files local to your machine will now be committed to this branch.

Writing your code

Open `MantelTest.py` in an editor of your choice. Ideally, use an editor that's not going to fiddle with the formatting of the file, e.g. changing tab lengths. Sublime Text or Atom are good choices, but you could also use IDLE or a command line text editor such as Nano or Emacs if you're that way inclined.

In the file, you'll find that I've left some space for you to insert your function. Avoid editing any other parts of the code – doing so might make it tricky for git to resolve conflicts with other teams' changes. Take a quick look over the rest of the code to get an idea of how the module is designed to work (it may

not be obvious until all the teams have contributed their code). Then add your function according to the specification given above and the tips below.

Tips for writing your function

Feel free to use online resources and to write your function however you like. If you're not too sure, here's one way you might approach the task:

- Define the name of the function and the arguments it needs to take
- Initialize an empty list to store the correlation coefficients
- Run a loop for the number of randomizations
- On each iteration, shuffle one of the lists using `ShuffleDistances()` …
- … and correlate the shuffled and unshuffled lists using `stats.pearsonr(x, y)[0]`
- Add this correlation coefficient into your list of correlation values
- Return the mean and standard deviation of the correlation values using `mean()` and `std()`

Test your code

Run `python` or `idle` from the command line, or launch another Python IDE of your choosing. Import the `MantelTest` module. Although you won't be able to use the full module yet, you should be able to test your function in isolation. Does it work as prescribed in the spec?

Commit your code to the branch

Switch over to the GitHub app and click on the Changes tab at the top of the window. It should have automatically detected that you've made changes to `MantelTest.py` and should highlight any new code in green and any deleted stuff in red. Check that you're happy with your new code and add a summary and description. Then click the 

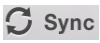
button. Your commit will show up under Unsynced Commits.

Merge your branch back into the master

Switch to the Branches tab and click 

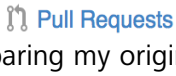
. Drag and drop your new and master branches into the relevant boxes and click Merge Branches. The commit you made to the new branch will now be merged into the master branch. Switch into the master by clicking the button on the master branch and then Switch To This Branch. Master should now be your "Current branch".

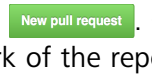
Syncing with GitHub

Click 

in the top right corner of the app. The changes you've made to the repo on your machine will be pushed up to your fork of the repo on GitHub.

Create a pull request

The final step is to ask the owner of the original repo (me) to pull in your changes. You do this by creating a "pull request". Switch over to the GitHub website and reload the page. Click 

in the right-hand sidebar and then 

. GitHub should automatically suggest comparing my original LECretreat15 repo with your fork of the repo and should highlight the differences between them. Click the 

button and write a title and description for your request. Usually, you'll want to explain to the repo owner why they should incorporate your code. Click 

.

Voila!

Using GitHub for scientific research

LEC Retreat 2015

Jon W. Carr • Sunday, 18th January 2015

Cheat sheet

Team 1

```
def ReadFile(filename):
    file_handle = open(filename)
    file_content = file_handle.read()
    file_handle.close()
    lines = file_content.split('\n')
    column1 = []
    column2 = []
    for line in lines:
        part = line.split(',')
        column1.append(part[0])
        column2.append(part[1])
    return column1, column2
```

Team 2

```
def PairwiseDistances(string_list):
    n = len(string_list)
    pairwise_distances = []
    for i in range(0, n-1):
        for j in range(i+1, n):
            dist = LevenshteinDistance(string_list[i], string_list[j])
            pairwise_distances.append(dist)
    return pairwise_distances
```

Team 3

```
def MonteCarlo(list1, list2, randomizations):
    correlations = []
    for i in xrange(0, randomizations):
        r = stats.pearsonr(list1, ShuffleDistances(list2))[0]
        correlations.append(r)
    return mean(correlations), std(correlations)
```