CSCD01 Deliverable 4

Issue 1: Overhaul external process calls in TexManager (https://github.com/matplotlib/matplotlib/issues/7490)

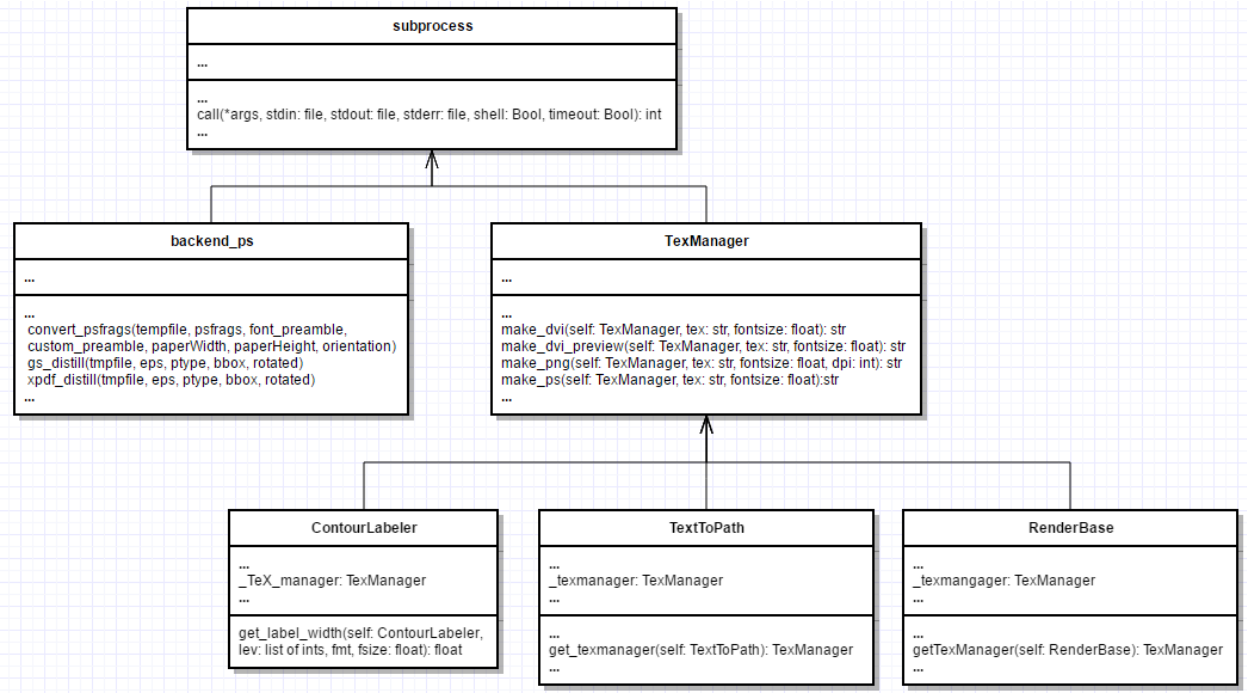
This issue here is that there are failures in the class TexManager due to a race condition around the various system calls. The program right now is only catching the standard output, but what is wanted is for the standard error to also be caught so that it can aid in debugging. The other issue is that an older system call module is being used, the os.system module, so the old system module needs to be switch with the newer subprocess module.

Issue 2: Copy a color map object does not isolate changes to cm (https://github.com/matplotlib/matplotlib/issues/8299)

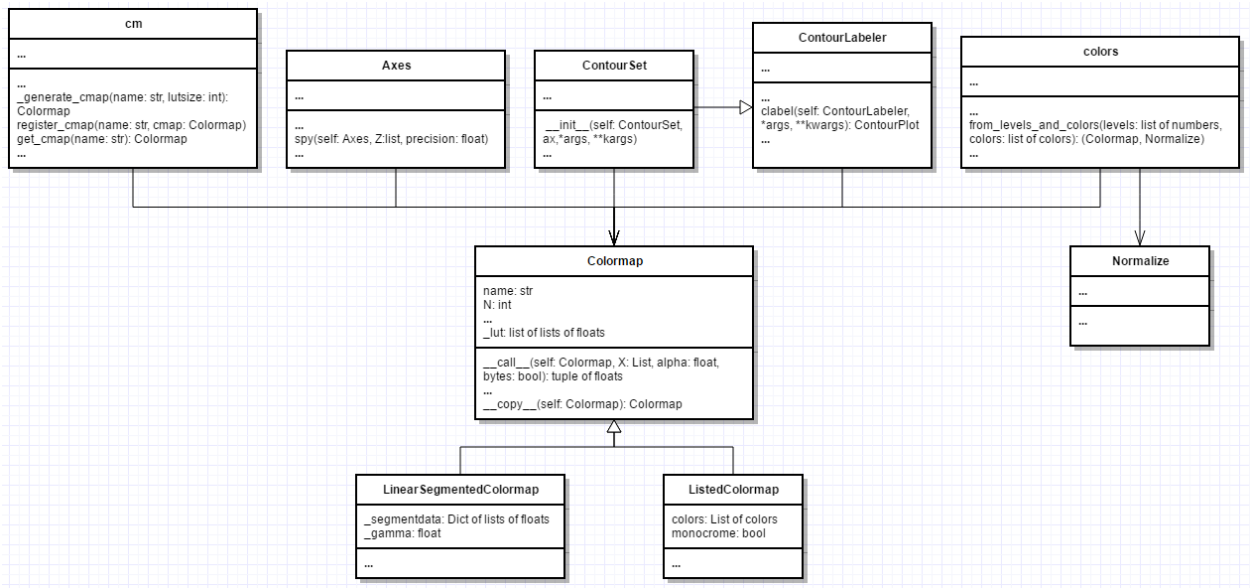
The issue here is that the color map object is mutable, so creating another variable which references to it and then using that variable may result in the value of the original variable also being changed. When you try to copy a new color map, the copy library uses a default copy method which only copies a reference the color map. This means that when you create a copy, any changes you make to the copy are also applied to the original.

Implementation

Issue 1: In TexManager and backend_ps we are going to upgrade all instances of the old system calls of os.system(cmd) to the new system calls of subprocess.call(cmd, shell=True). After each modified system call we would then try to catch all outputs to stderr and return them in exceptions because right now only stdout is caught and returned, but stderr would be good for debugging.



Issue 2: In the Colormap file we are going to implement a __copy__ function that will copy the entire Colormap object and return it. The specific part that need special attention is the _lut attribute which contains a list of lists of floats, this is a major part of the colormap. We will copy this attribute carefully, so that we create a new object that is a list of lists of floats, and not just a copy of the address of the original _lut attribute. This needs to be done carefully because lists are mutable, so if we do not create a brand new list of lists of floats object then when someone modifies the copy's colormap, the original's colormap will also be modified.



Code Trace:

Before bug fix:

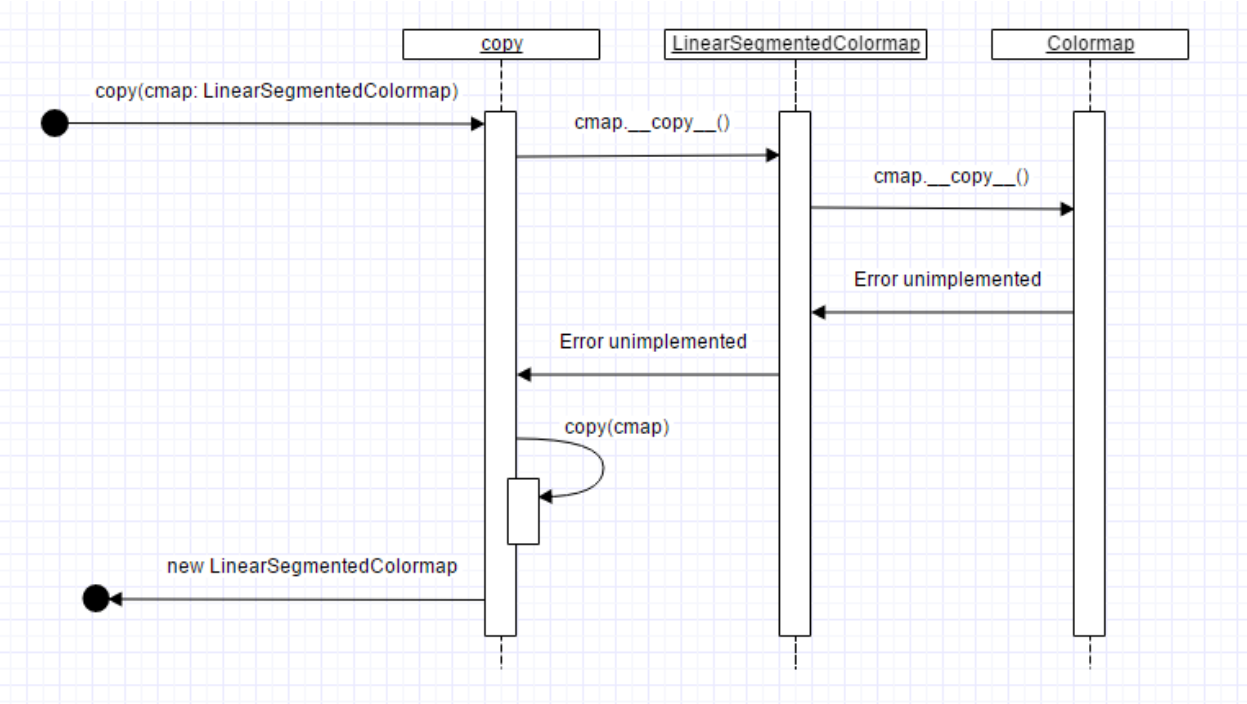


Figure 1: Runs a default copy function

After bug fix:

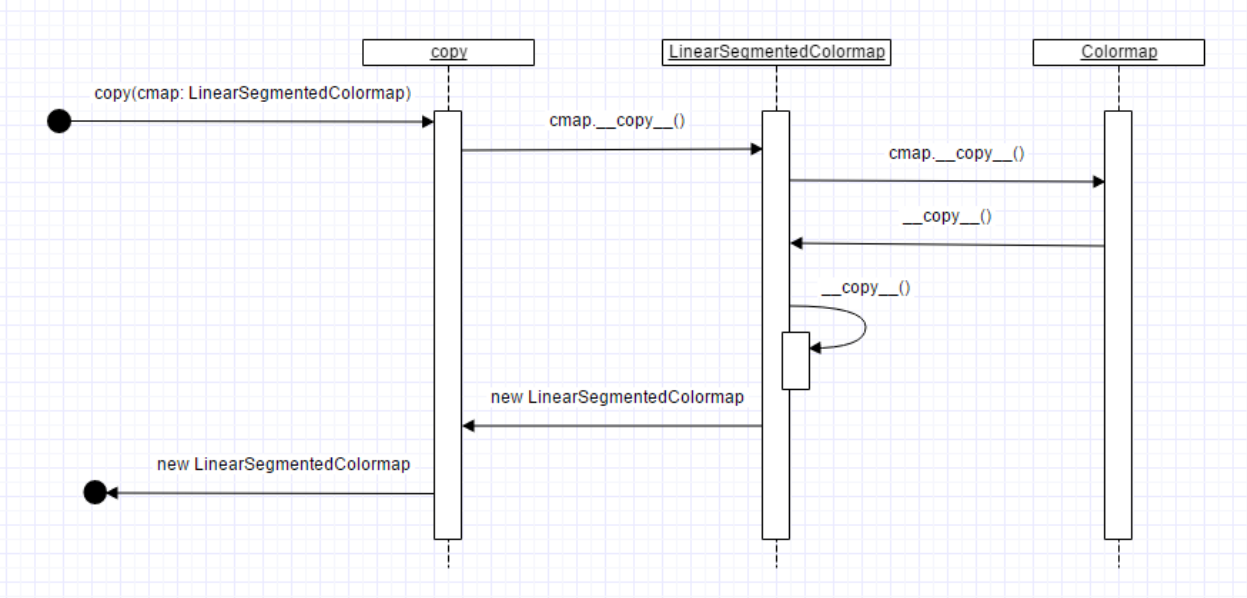


Figure 2:Copy.copy tries to call linearsegmentedcolormap.copy if not a default copy function is run

The issue we chose to implement

We decided to implement issue 2 because it involved a class which was easier to understand, and we had a better idea of how to implement it than issue 1. We found that issue 1 was harder to reproduce, so it seemed better to implement the second issue. Our attempts at tracing the code in Issue 1 showed that determining what was happening in the code was extremely difficult, and the fact that we couldn't even trace through the code made us realize that it would be very difficult to try to a fix a bug for a piece of code that we couldn't trace through or test properly.

Acceptance Tests

Each of the acceptance tests will produce images, the acceptance tests pass if they look like the images which are posted in the acceptance tests section of the repository. The comments at the beginning of each file indicate what each image generated by the test files should look like.

```
1  import matplotlib.pyplot as plt
2  import copy
3  import numpy as np
4
5  #test should not crash and the console should be empty
6
7  #test that copying and modifying only modifies the modified colormap and not the original
8  #test will create 4 graphs
9
10 #test_copy_modify_save_multiple-original.png and test_copy_modify_save_multiple-unmodified.png and
11 #       test_copy_modify_save_multiple-unmodified2.png should look identical to original.png
12
13 #test_copy_modify_save_multiple-modified.png should look identical to modified.png
14
15 data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
16 cm1 = plt.cm.Reds
17 plt.imshow(data,cmap=cm1)
18 plt.savefig("test_copy_modify_save_multiple-original.png")
19 plt.cla()
20
21 cm2 = copy.copy(cm1)
22 cm2.set_bad('b')
23 plt.imshow(data,cmap=cm1)
24 plt.savefig("test_copy_modify_save_multiple-unmodified.png")
25 plt.cla()
26
27 plt.imshow(data,cmap=cm2)
28 plt.savefig("test_copy_modify_save_multiple-modified.png")
29 plt.cla()
30
31 plt.imshow(data,cmap=cm1)
32 plt.savefig("test_copy_modify_save_multiple-unmodified2.png")
33 plt.cla()
```

Figure 3: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_copy_modify_save_multiple.py

```python
import matplotlib.pyplot as plt
import copy
import numpy as np

#test should not crash and the console should be empty

#test that copying and modifying still produces a modified colormap
#test should create 2 graphs
#test_copy_modify_save_new-new_modified_copy.png should look identical to modified.png
#test_copy_modify_save_new-original.png should look identical to original.png

data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
cm1 = plt.cm.Reds
plt.imshow(data,cmap=cm1)
plt.savefig("test_copy_modify_save_new-original.png")
plt.cla()

cm2 = copy.copy(cm1)
cm2.set_bad('b')
plt.imshow(data,cmap=cm2)
plt.savefig("test_copy_modify_save_new-new_modified_copy.png")
plt.cla()
```

Figure 4: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_copy_modify_save_new.py

```python
import matplotlib.pyplot as plt
import copy
import numpy as np

#test should not crash and the console should be empty

#test that copying still creates a colormap
#test should create 2 graphs
#test_copy_save_new-original.png and test_copy_save_new-copy.png should look identical to original.png

data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
cm1 = plt.cm.Reds
plt.imshow(data,cmap=cm1)
plt.savefig("test_copy_save_new-original.png")
plt.cla()

cm2 = copy.copy(cm1)
plt.imshow(data,cmap=cm2)
plt.savefig("test_copy_save_new-copy.png")
plt.cla()
```

Figure 5: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_copy_save_new.py

```python
import matplotlib.pyplot as plt
import copy
import numpy as np

#test should not crash and the console should be empty

#test that copying does not affect original colormap
#test should create 2 graphs
#test_copy_save_old-original.png and test_copy_save_old-notcopy.png should look identical to original.png

data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
cm1 = plt.cm.Reds
plt.imshow(data,cmap=cm1)
plt.savefig("test_copy_save_old-original.png")
plt.cla()

cm2 = copy.copy(cm1)
plt.imshow(data,cmap=cm1)
plt.savefig("test_copy_save_old-notcopy.png")
plt.cla()
```

Figure 6: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_copy_save_old.py

```python
import matplotlib.pyplot as plt
import copy
import numpy as np

#test should not crash and the console should be empty

#test that modifying a colormap still works
#test should create 2 graphs
#test_no_copy_modify_save-modified.png should look identical to modified.png
#test_no_copy_modify_save-unmodified.png should look identical to original.png

data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
cm1 = plt.cm.Reds
plt.imshow(data,cmap=cm1)
plt.savefig("test_no_copy_modify_save-unmodified.png")
plt.cla()

cm1.set_bad('b')
plt.imshow(data,cmap=cm1)
plt.savefig("test_no_copy_modify_save-modified.png")
plt.cla()

```

Figure 7: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_no_copy_modify_save.py

```
1  import matplotlib.pyplot as plt
2  import copy
3  import numpy as np
4
5  #test should not crash and the console should be empty
6
7  #test that the adding a colormap still works
8  #test_no_copy_save.png should look identical to original.png
9
10 data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])
11 cm1 = plt.cm.Reds
12 plt.imshow(data,cmap=cm1)
13 plt.savefig("test_no_copy_save.png")
14 plt.cla()
```

Figure 8: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/acceptence_tests/test_no_copy_save.py

Unit Tests

Unit tests: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/unit_tests/colormap_unittests.py

```
@cleanup
def test_no_copy_save():
        '''
        Tests that adding a colormap still works.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_no_copy_save.png")

        #remove temporary image
        os.remove("test_no_copy_save.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_no_copy_save, Adding a colormap failed")


@cleanup
def test_no_copy_modify_save():
        '''
        Tests that modifying the colormap still works.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_no_copy_modify_save-unmodified.png")

        #remove temporary image
        os.remove("test_no_copy_modify_save-unmodified.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_no_copy_modify_save, Original colormap was wrong")

        #modify the colormap
        cm1.set_bad('b')
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_no_copy_modify_save-modified.png")

        #remove temporary image
        os.remove("test_no_copy_modify_save-modified.png")
        assert_array_equal(cm1._lut, np.load("modified.npy"), "test_no_copy_modify_save, Modifying a colormap failed")
```

```
@cleanup
def test_copy_save_old():
        '''
        Tests that copying does not affect original colormap.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_save_old-original.png")

        #remove temporary image
        os.remove("test_copy_save_old-original.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_save_old, Original colormap was wrong")

        #copy the colormap and save the original
        cm2 = copy.copy(cm1)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_save_old-notcopy.png")

        #remove temporary image
        os.remove("test_copy_save_old-notcopy.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_save_old, Original colormap was changed")


@cleanup
def test_copy_save_new():
        '''
        Tests that copying still creates a colormap.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_save_new-original.png")

        #remove temporary image
        os.remove("test_copy_save_new-original.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_save_new, Original colormap was wrong")

        #copy the colormap and save the copy
        cm2 = copy.copy(cm1)
        plt.imshow(data,cmap=cm2)
        plt.savefig("test_copy_save_new-copy.png")

        #remove temporary image
        os.remove("test_copy_save_new-copy.png")
        assert_array_equal(cm2._lut, np.load("unchanged.npy"), "test_copy_save_new, Copied colormap was changed")
```

```
@cleanup
def test_copy_modify_save_new():
        '''
        Tests that copying and modifying a colormap still produces a modified
        colormap.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_modify_save_new-original.png")

        #remove temporary image
        os.remove("test_copy_modify_save_new-original.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_modify_save_new, Original colormap was wrong")

        #copy the colormap, modify the colormap, and save the copy
        cm2 = copy.copy(cm1)
        cm2.set_bad('b')
        plt.imshow(data,cmap=cm2)
        plt.savefig("test_copy_modify_save_new-new_modified_copy.png")

        #remove temporary image
        os.remove("test_copy_modify_save_new-new_modified_copy.png")
        assert_array_equal(cm2._lut, np.load("modified.npy"), "test_copy_modify_save_new, Copied colormap was not modified")


@cleanup
def test_copy_modify_save_multiple():
        '''
        Tests that copying and modifying only modifies the modified colormap and
        not the original.
        '''
        data = np.ma.masked_array([[1,2,3],[2,3,4]], mask=[[1,0,0],[0,0,0]])

        #use a copy to make it independent from other tests
        cm1 = copy.copy(plt.cm.Reds)
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_modify_save_multiple-original.png")

        #remove temporary image
        os.remove("test_copy_modify_save_multiple-original.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_modify_save_multiple, Original colormap was wrong")

        #copy the colormap, modify the colormap, and save the original
        cm2 = copy.copy(cm1)
        cm2.set_bad('b')
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_modify_save_multiple-unmodified.png")

        #remove temporary image
        os.remove("test_copy_modify_save_multiple-unmodified.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_modify_save_multiple, Original colormap was changed")

        #save the copy
        plt.imshow(data,cmap=cm2)
        plt.savefig("test_copy_modify_save_multiple-modified.png")

        #remove temporary image
        os.remove("test_copy_modify_save_multiple-modified.png")
        assert_array_equal(cm2._lut, np.load("modified.npy"), "test_copy_modify_save_multiple, Copied colormap was not changed")

        #save the original
        plt.imshow(data,cmap=cm1)
        plt.savefig("test_copy_modify_save_multiple-unmodified2.png")

        #remove temporary image
        os.remove("test_copy_modify_save_multiple-unmodified2.png")
        assert_array_equal(cm1._lut, np.load("unchanged.npy"), "test_copy_modify_save_multiple, Original colormap was changed after copied one was saved")
```

Script to run the unit tests: https://github.com/CSCD01-Winter2017/team06-Project/blob/master/Deliverable4/unit_tests/test_all.py