Cuffdiff2links User Guide

Qiaonan Duan
Ma'ayan Lab, Icahn School of Medicine at Mount Sinai

Introduction

Cuffdiff2links is a python library to process .diff files from RNA-seq data. It finds significant genes in .diff files, separates them into up/down gene lists and posts these lists to Enrichr for enrichment analysis. It also generates a file for principle component analysis (PCA) in Matlab if *read_group_tracking* files exist for the .diff files processed. This PCA input file contains FPKM values for each replicate of all significant genes unioned across different comparison groups. Matlab functions to perform PCA analysis on this file are also provided.

(Although all demos are performed on Windows platform, the library has been tested to work on Mac OS)

Index

I. Installation

1. Download and install python 2.7.
2. Add python to system environment variable. (Appendix I)
3. Install python poster library. (Appendix II)
4. Download and unzip *cuffdiff2links.zip.*
5. In the unzipped folder, run *setup.py* by double-click.

II. Processing .diff files independently

1. Open command window, type `python` and press Enter to enter python environment. You can also enter python environment by open python IDLE.

2. Import `readdiff` function from `cuffidff2links` library. Run following command:

```
>>> from cuffdiff2links import readdiff
```

3. To process a single .diff file, pass the path of the file to `readdiff` function and run:

```
>>>readdiff(r'file path string')
```

For each .diff file, `readdiff` function creates a folder named *EnrichrLinks* in the .diff file's directory and put three output files there. The first file is *updown.txt* file that contains a table of differentially expressed genes grouped by comparison. The second file is *enrichrLinks.txt* file that provides links to Enrichr for each gene list. Copy the links to browser to see the enrichment analysis results. If *read_group_tracking* file of the .diff file processed exists, a PCA input file will be generated as *pcainput.txt* .

To process all .diff files in a folder (including those in subfolders), pass the folder path to `readdiff` function and run:

```
>>>readdiff(r'folder path string')
```

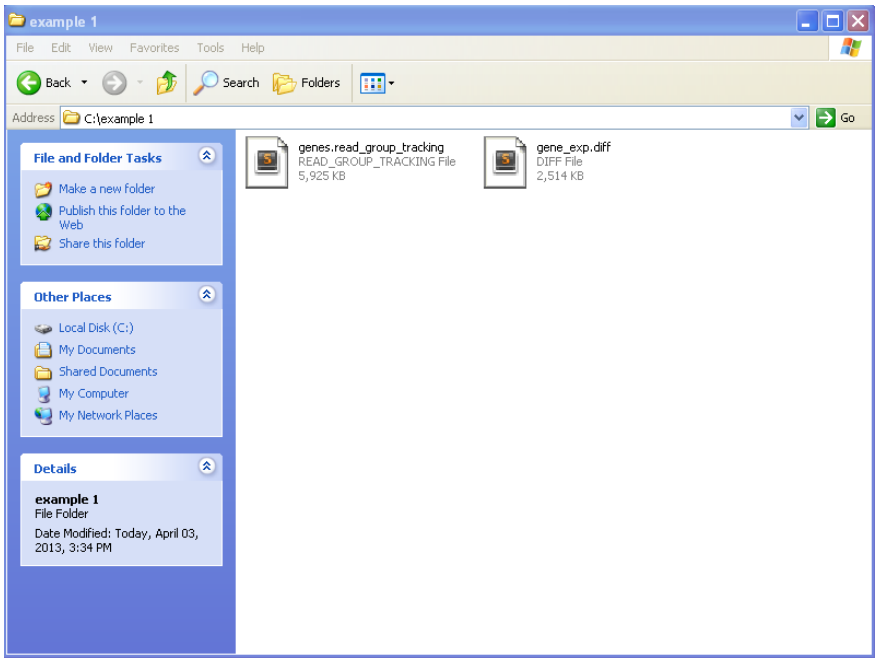
The little `r` before string is necessary by specification in python.

4. `readdiff` function has a second parameter that is a regular expression pattern to select and process a subset of .diff files within a folder. The default pattern is `r'\.+\.diff'` which find all files with a .diff extension. To process only *gene_exp.diff* files in a folder (including subfolders), run:

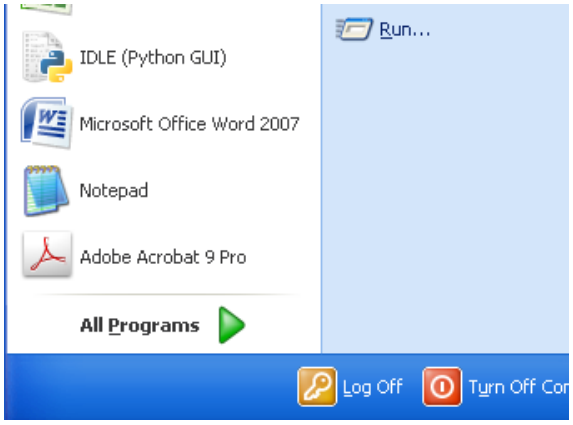
```
>>>readdiff(r'folder path string', 'gene_exp.diff')
```

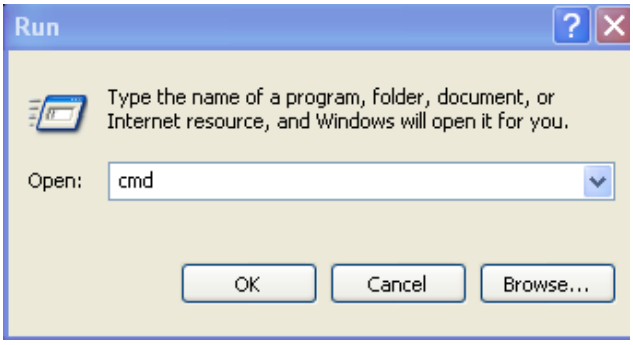
Example 1: process single .diff file

There is a *gene_exp.diff* file and its *read_group_tracking* file in the directory 'C:\example 1' . We will process this .diff file.

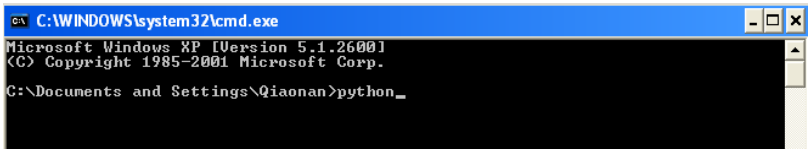


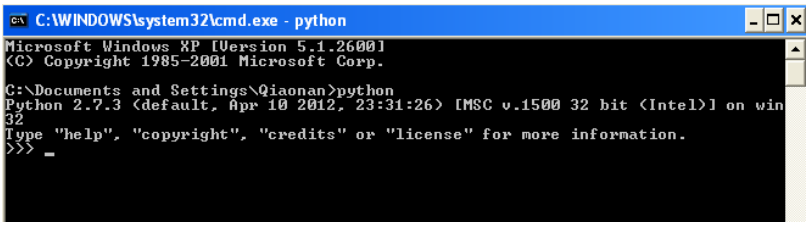
First, open command window: Start → Run → cmd → OK



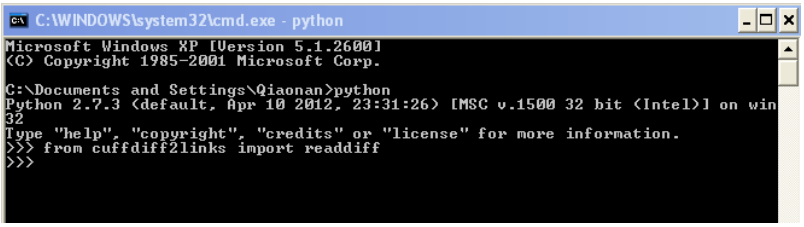


In the command window, type `python` and press Enter to enter python environment.

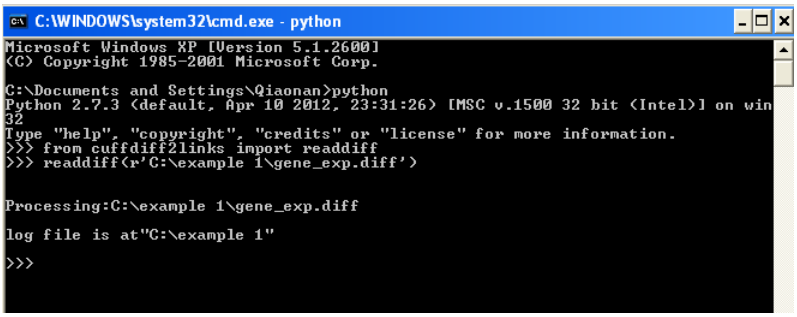




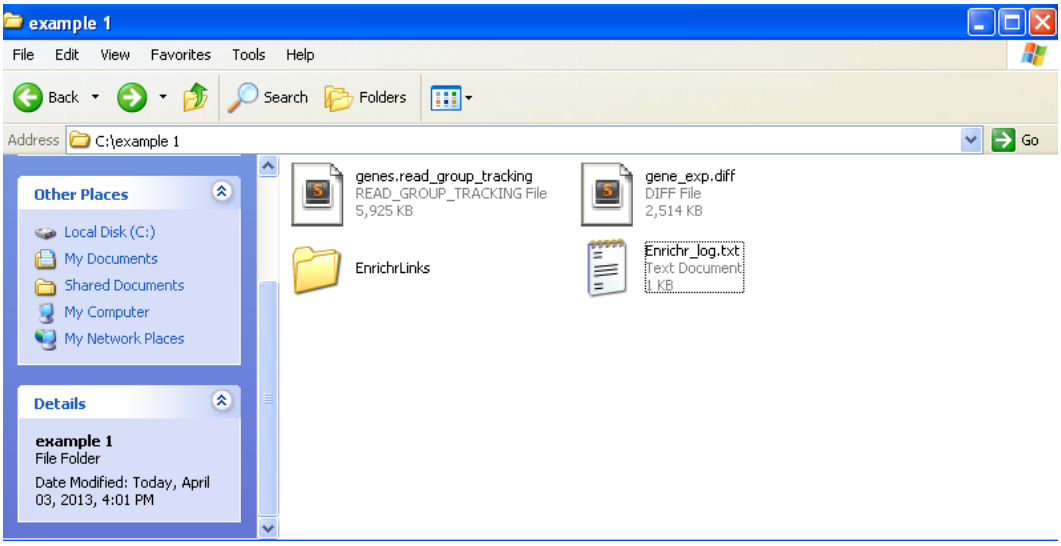
Type `from cuffdiff2links import readdiff` and press Enter to import `readdiff` function.



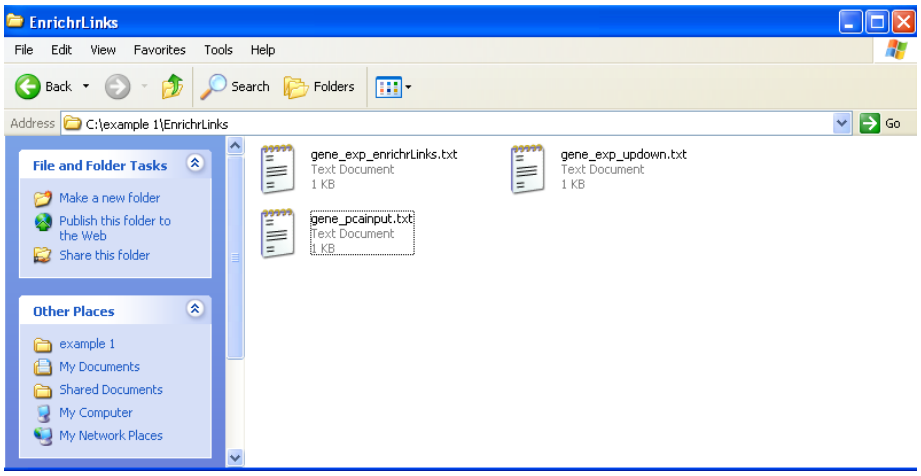
Type `readdiff(r'C:\example 1\gene_exp.diff')` to process the file.



After run above command, there is a new folder *EnrichrLinks* created in directory 'C:\example 1'. There is also one *Enrichr_log.txt* generated that save running-time log information.

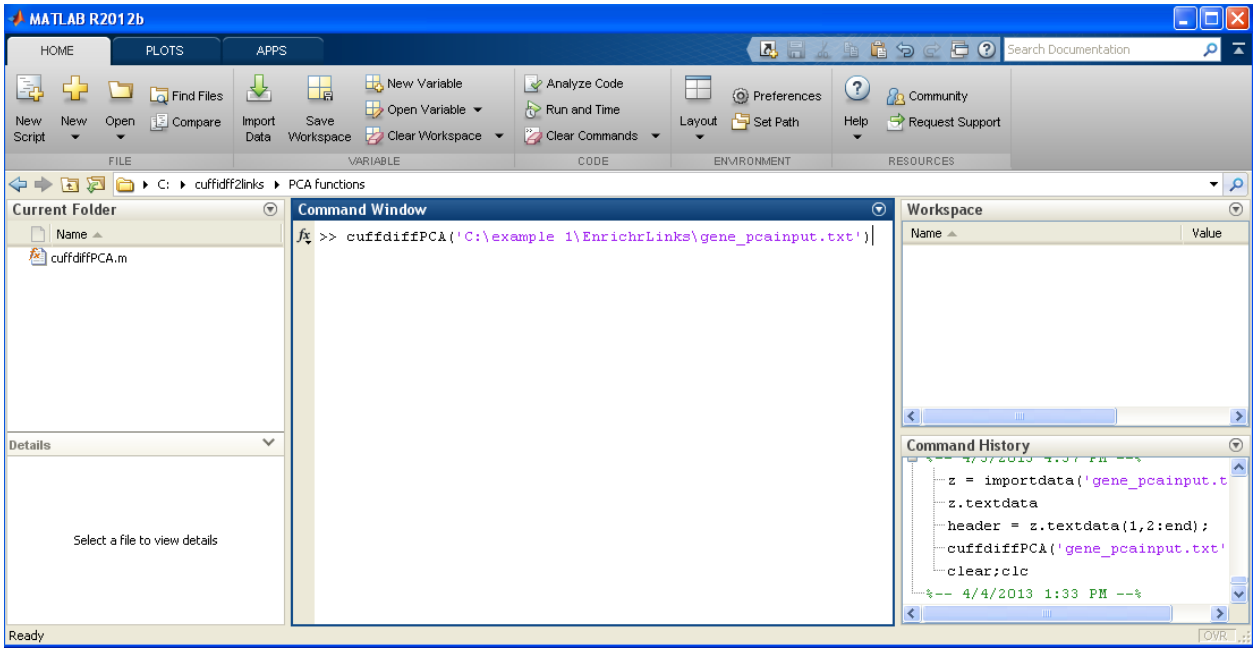


In the *EnrichrLinks* folder there are three output files: *gene_exp_enrichrLinks.txt, gene_exp_updown.txt, gene_pcainput.txt* .

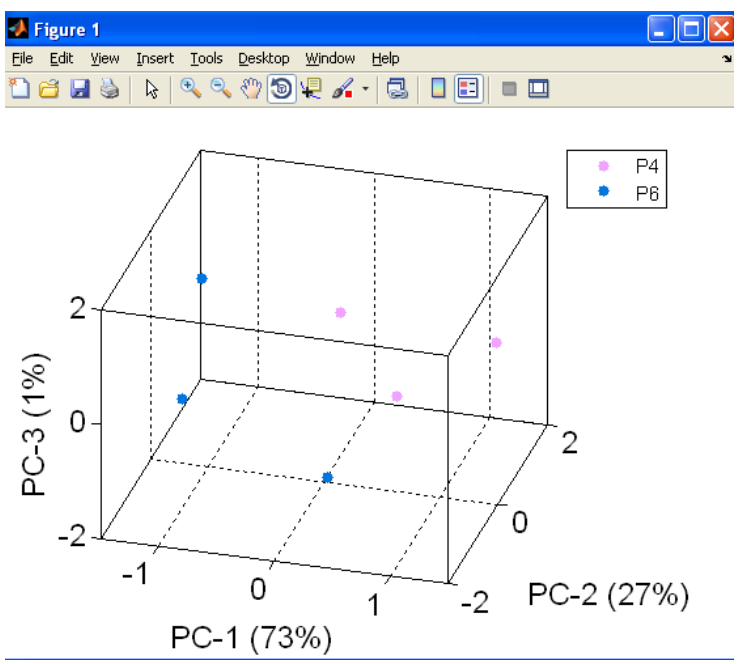


Principle Component Analysis in Matlab

Open Matlab, change current workspace to where *cuffdiffPCA.m* is located and run
`cuffdiffPCA('C:\example 1\EnrichrLinks\gene_pcainput.txt')`

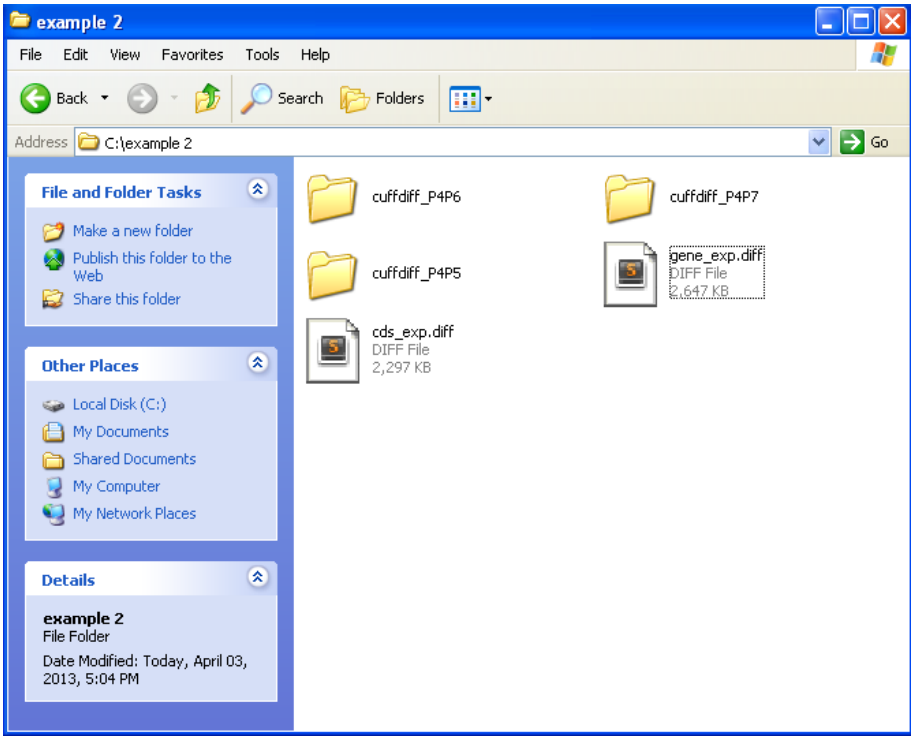


The result is a PCA figure like this. There is one comparison between P4 sample and P6 sample.

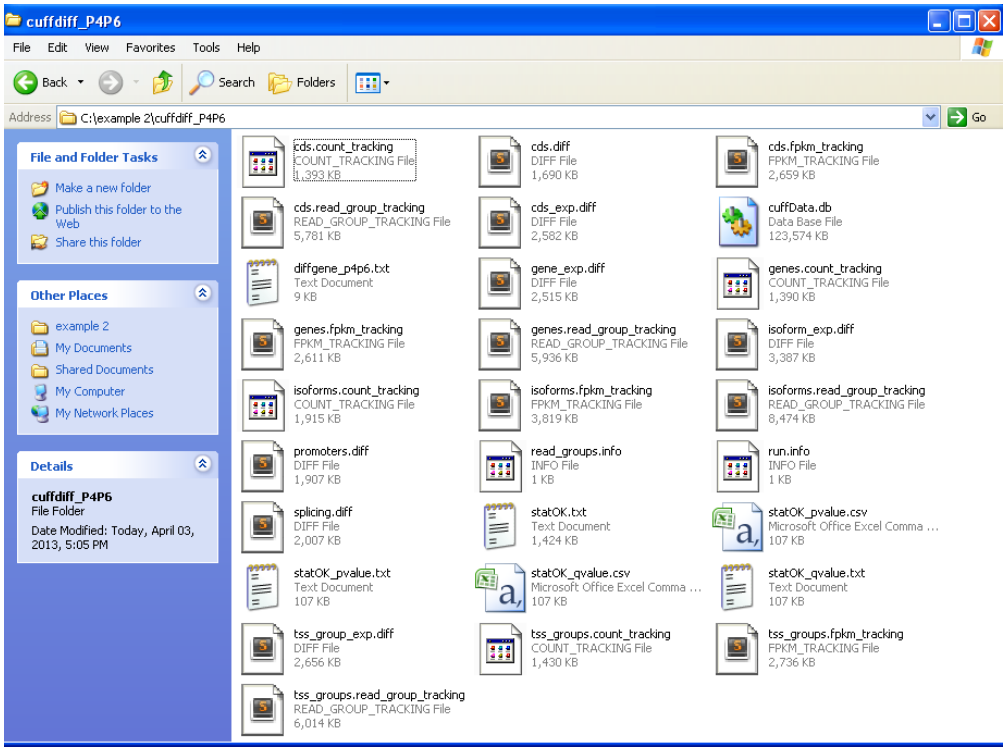


Example 2: process multiple .diff files in a folder independently

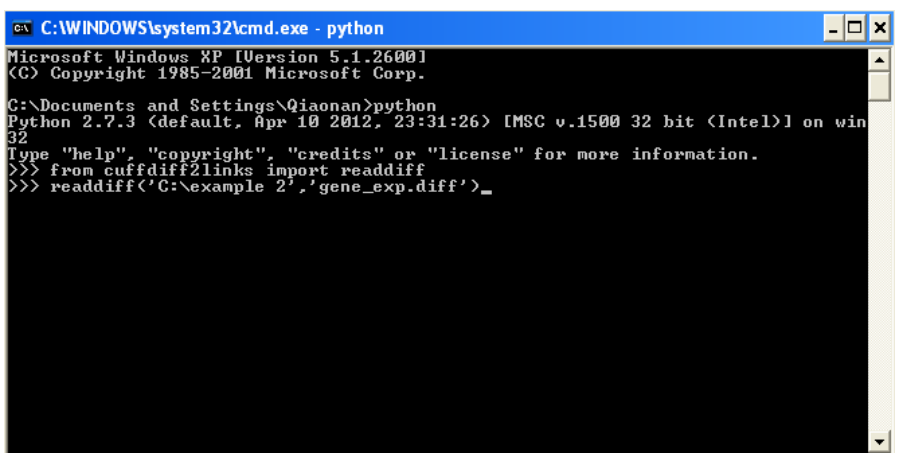
There are 2 .diff files and 3 subfolders in directory 'C:\example 2'.



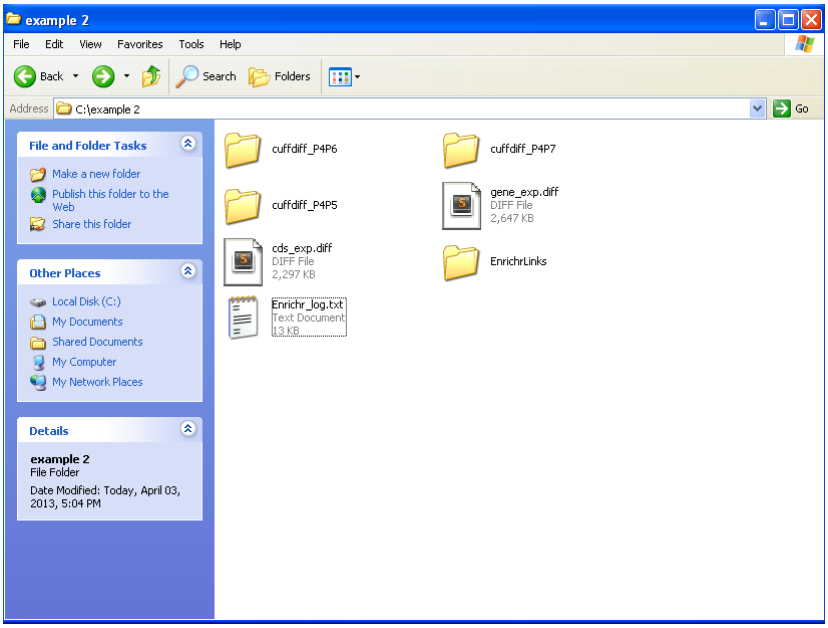
In each of the 3 subfolders, there are many additional .diff files. For example, in subfolder *cuffdiff_P4P6* there are files as following:



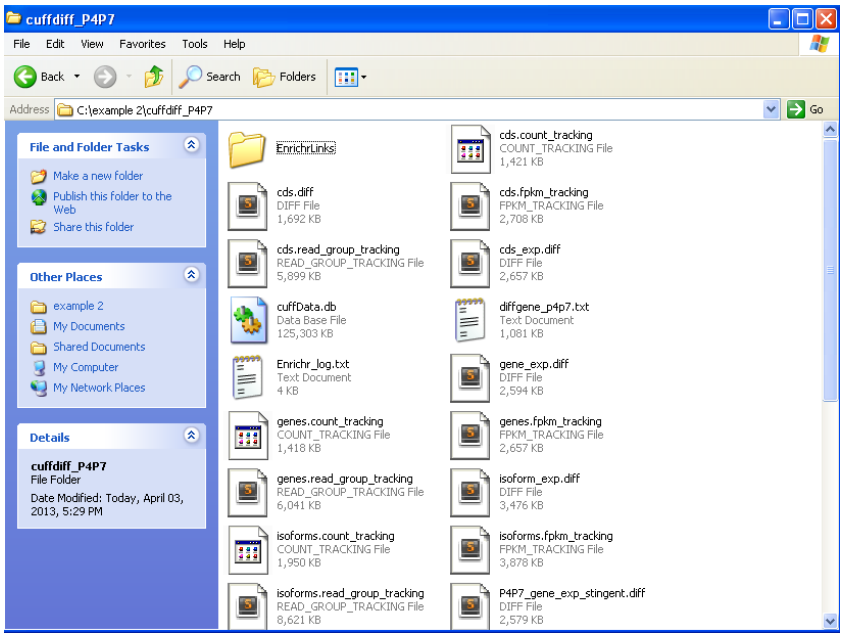
Now we would like to process all the *gene_exp.diff* files in *example 2* folder and all its subfolders. First, open command window and import `readdiff` as in Example 1. Then type `readdiff('C:\example 2','gene_exp.diff')` and press Enter. The second argument is a regular expression to select and process only files that match the pattern. Here it matches files with exact name of *gene_exp.diff*.



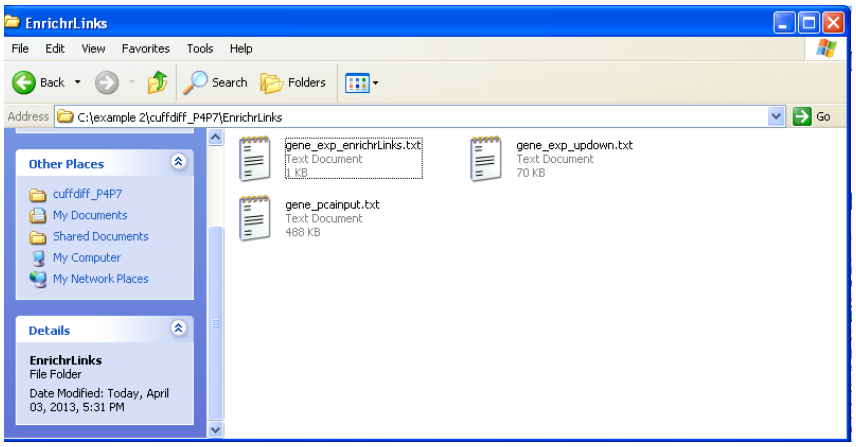
After execute the above command, *EnrichrLinks* folder is generated in *example 2* folder and all its subfolders. Each contains results for the *gene_exp.diff* file in its parent folder. An *Enrichr_log.txt* file is generated recording log information in *example 2* folder.



EnrichrLinks folder is also generated in each subfolder. Here takes the folder in *cuffdiff_P4P7* as an example



Each *EnrichrLinks* folder contain output files for the *gene_exp.diff* file in its parent folder. Below takes *EnrichrLinks* folder in *cuffdiff_P4P7* as an example.

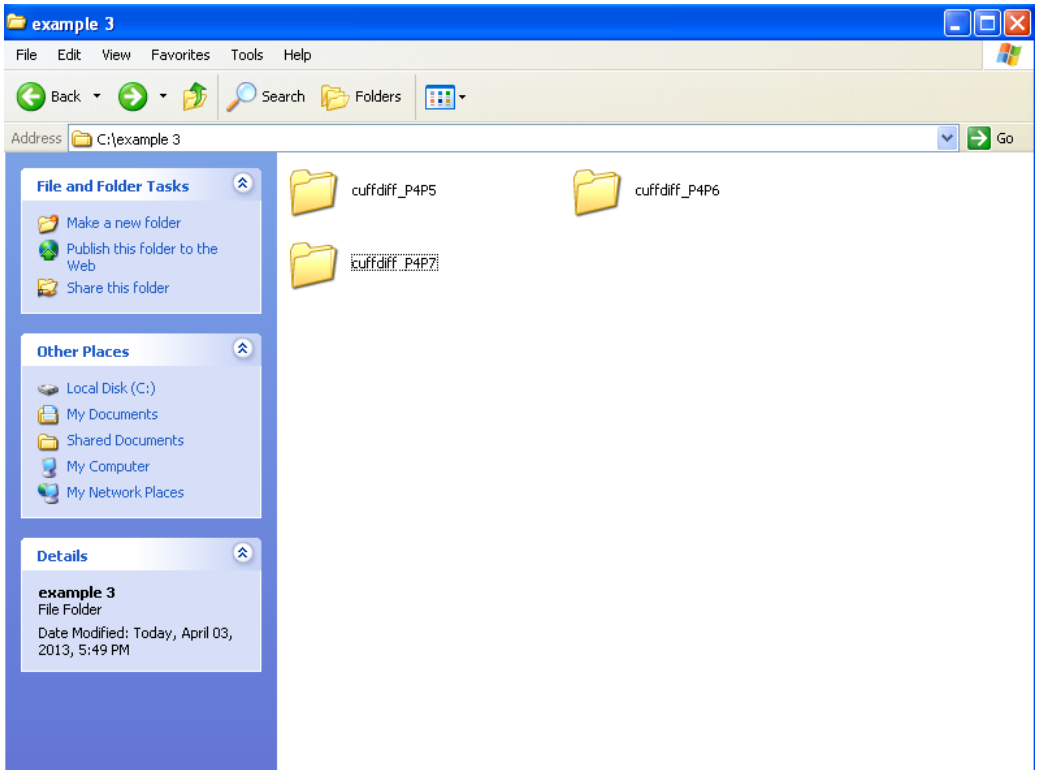


II. Processing several .diff files collectively

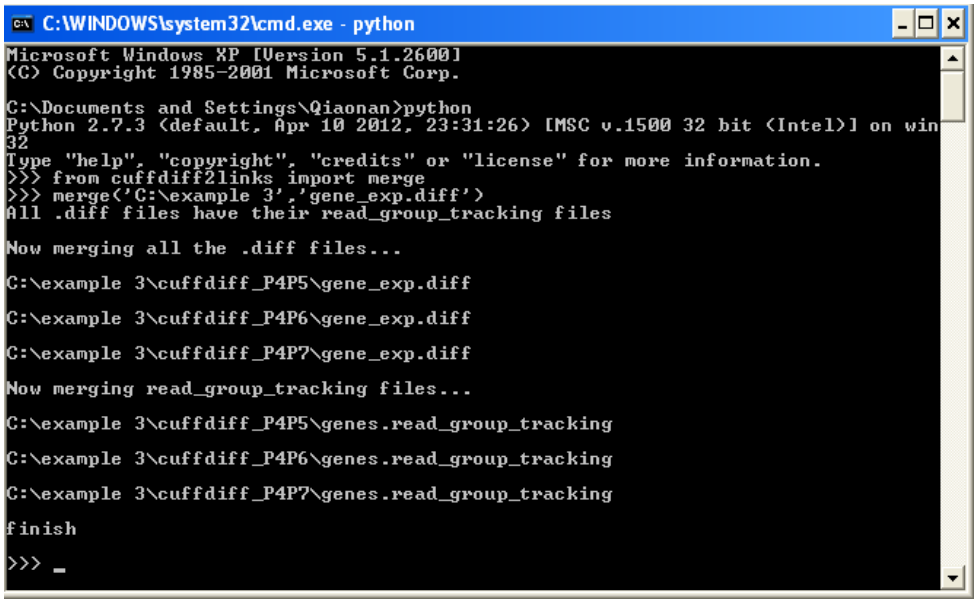
In some cases, comparisons of one RNA-seq experiment are separated in different .diff files in seperate folders. Cuffidff2links provides a `merge` function to merge .diff files and their *read_group_tracking* files scattered in different subfolders into a single file (excluding .diff files in subfolder named as *mergedFiles,* which is the output folder for merge function). Then `readdiff` function can be called upon these merged files to give a collective result. The use of `merge` function can be demonstrated in Example 3

Example 3.1: Processing several .diff files collectively with complete *read_group_tracking*files

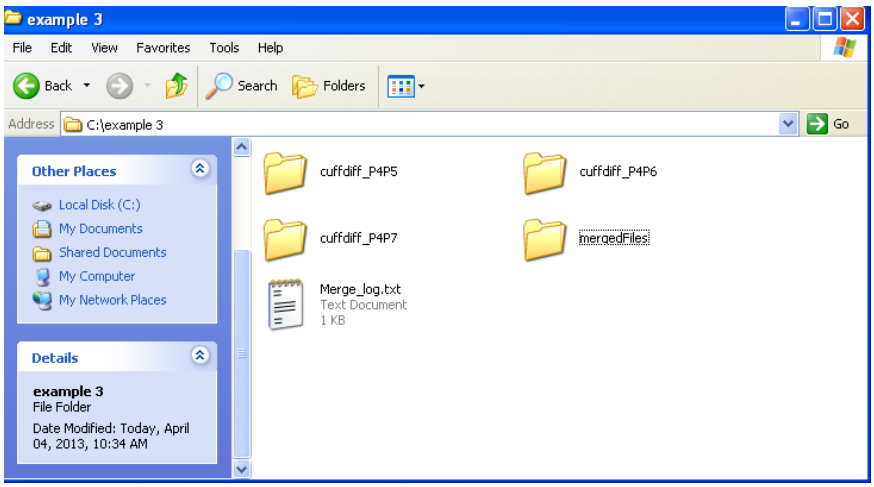
In this example, *gene_exp.diff* files in three folders come from a single RNA-seq experiment. The experiment has four samples P4, P5, P6 and P7. P4 is control and the other three samples are compared with P4. The *gene_exp.diff* files are the result of these three comparisons. We would like have a PCA input file from significant genes in all three comparisons.



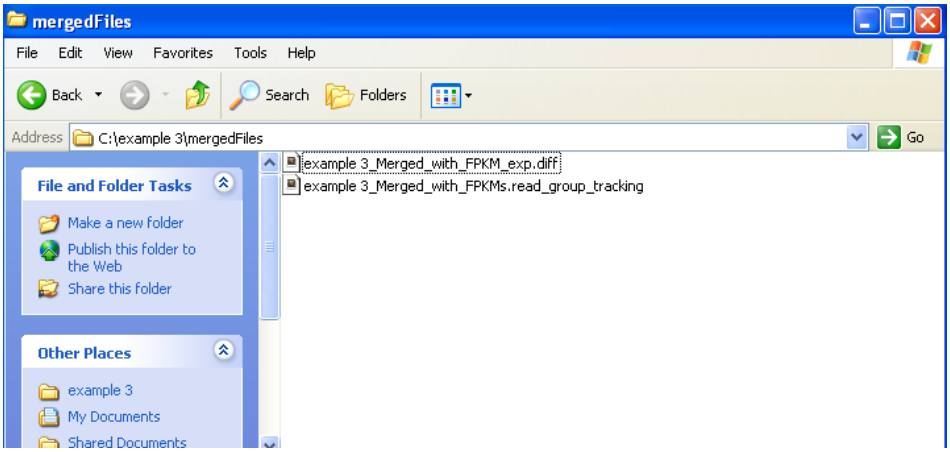
Open command window as in Example 1 and Example 2. Import `merge` function by typing `from cuffdiff2links import merge` and press Enter. Then type `merge('C:\example 3','gene_exp.diff')` and press Enter to merge. The second argument of merge function is also a regular expression pattern to select and merge designated files. Here we merge *gene_exp.diff* files and their *read_group_tracking* files.



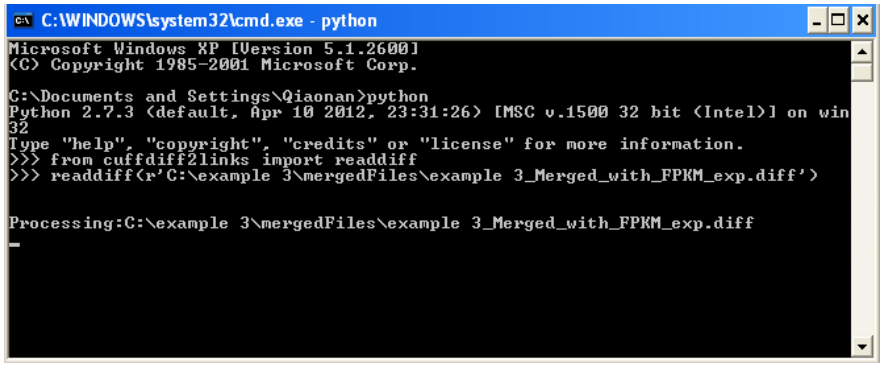
After run the command, there is a folder generated in 'C:\example 3' directory called *mergedFiles*, in which are all the output files. There is also one *Merge_log.txt* file generated that record all the log information displayed in the command window.



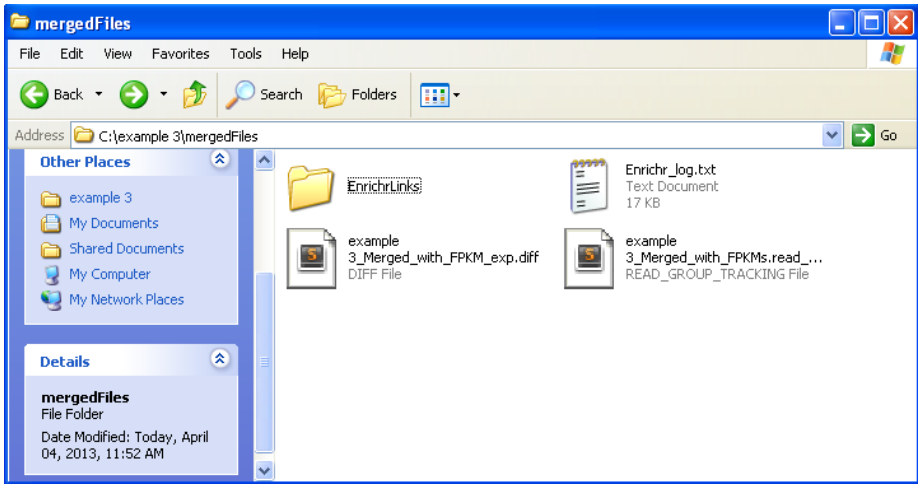
Open the *mergedFiles* folder, there are two files: *example 3_Merged_with_FPKM_exp.diff* and *example 3_Merged_with_FPKMs.read_group_tracking*. The first is the merged .diff file and second file is the merged *read _group_tracking* file.



Then `readdiff` function can be called on this merged .diff file. First import readdiff function and run: `readdiff(r'C:\example 3\mergedFiles\example 3_Merged_with_FPKM_exp.diff')`



After running above command, *EnrichrLinks* folder and *Enrichr_log.txt* are generated as in example 1 and example 2.

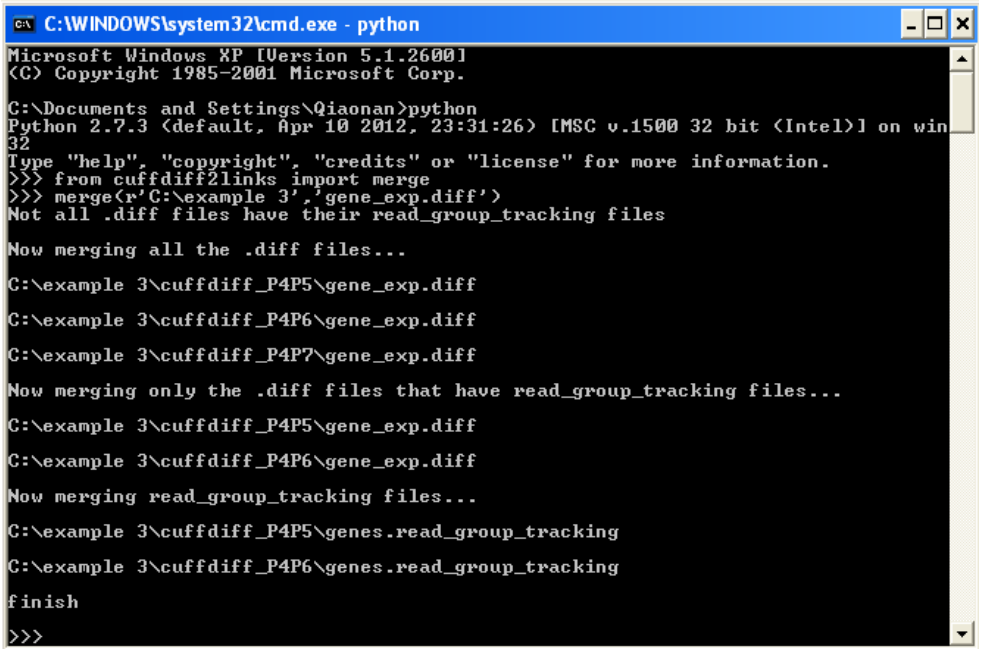


Example 3.2: Processing several .diff files collectively with incomplete *read_group_tracking* files

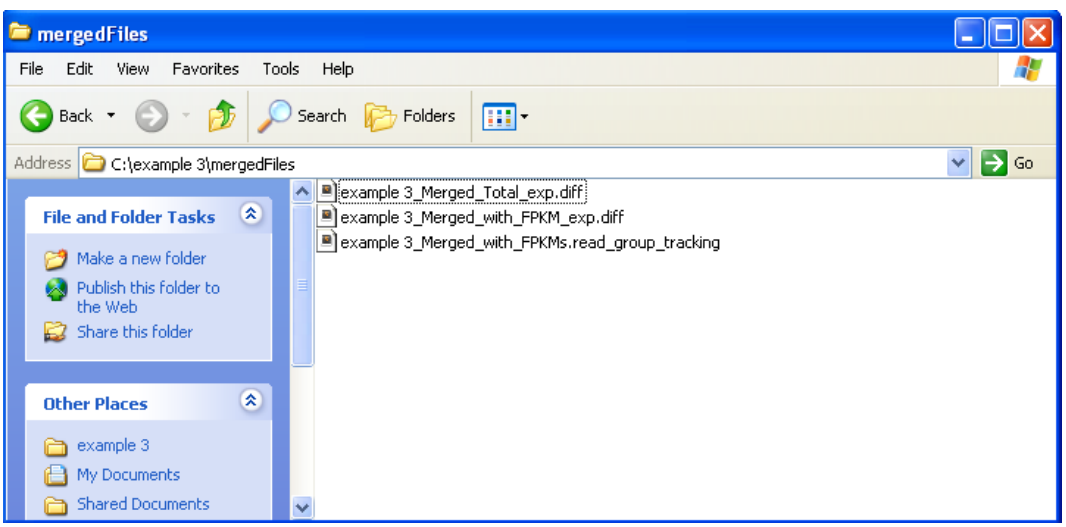
Sometimes not every .diff file has its *read_group_tracking* file. The merge function has been designed to deal with this situation. It will first merge all the .diff files into a single *foldername_Merged_Total_exp.diff* file. Then it merge the .diff files that have *read_group_tracking* files into another .diff file named as *foldername_Merged_with_FPKM_exp.diff* and also the

read_group_tracking files into a single file named as *foldername_Merged_with_FPKMs.read_group_tracking*.

We still use files in example 3.1. But this time we delete the *genes.read_group_tracking* file in *cuffdiff_P4P7* folder. So now the *gene_exp.diff* file does not have its *read_group_tracking* file. We then execute merge function upon directory 'C:\example 3' as in example 3.1:



Compare this with the log information in example 3.1, you can tell the difference. Now open the *mergedFiles* folder generated, there are three files generated rather than two.



Appendix (For Windows):

I. How to add python to system environment variables:
http://www.katsbits.com/tutorials/blender/setting-up-windows-python-path-system-variable.php

II. How to install poster library:

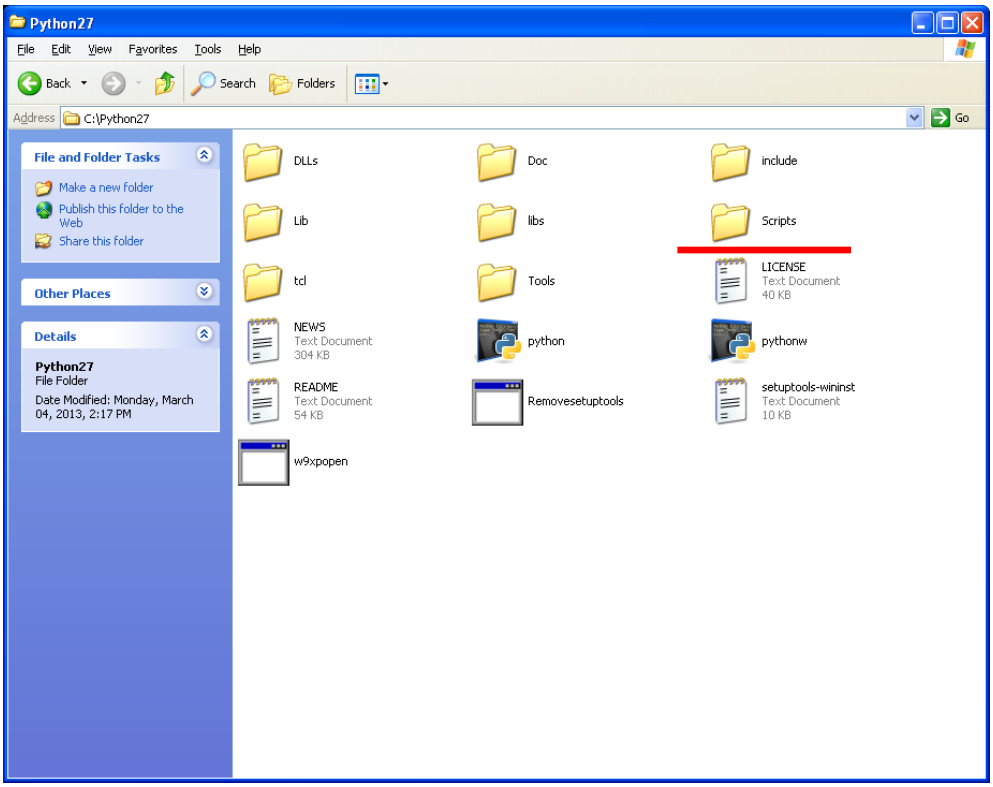
1. Install easy_insall python package.

Go to website: https://pypi.python.org/pypi/setuptools

On the webpage, select proper file and download.

File	Type	Py Version	Uploaded on	Size	# downloads
setuptools-0.6c11-1.src.rpm (md5) built for redhat 4.3	RPM	any	2009-10-20	263KB	45919
setuptools-0.6c11-py2.3.egg (md5)	Python Egg	2.3	2009-10-20	1MB	21582
setuptools-0.6c11-py2.4.egg (md5)	Python Egg	2.4	2009-10-20	329KB	297287
setuptools-0.6c11-py2.5.egg (md5)	Python Egg	2.5	2009-10-20	325KB	565849
setuptools-0.6c11-py2.6.egg (md5)	Python Egg	2.6	2009-10-20	325KB	1746924
setuptools-0.6c11-py2.7.egg (md5)	Python Egg	2.7	2010-07-08	324KB	2007323
setuptools-0.6c11.tar.gz (md5)	Source		2009-10-20	250KB	433050
setuptools-0.6c11.win32-py2.3.exe (md5)	MS Windows installer	2.3	2009-10-20	218KB	17920
setuptools-0.6c11.win32-py2.4.exe (md5)	MS Windows installer	2.4	2009-10-20	222KB	10171
setuptools-0.6c11.win32-py2.5.exe (md5)	MS Windows installer	2.5	2009-10-20	222KB	57996
setuptools-0.6c11.win32-py2.6.exe (md5)	MS Windows installer	2.6	2009-10-20	222KB	174578
setuptools-0.6c11.win32-py2.7.exe (md5)	MS Windows installer	2.7	2010-07-08	222KB	407190

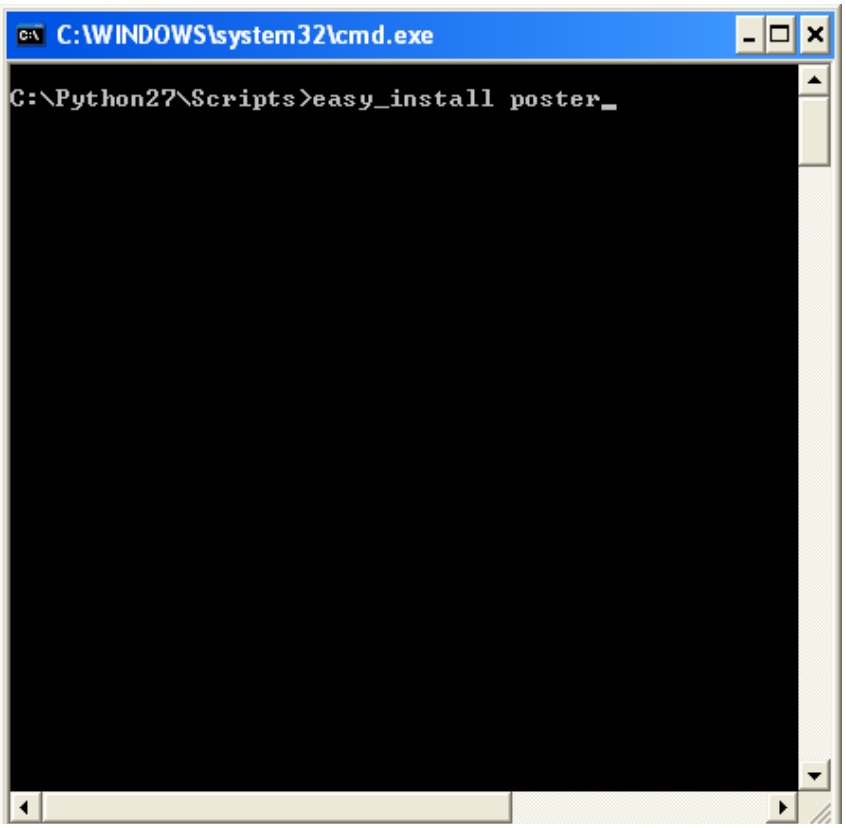
Install the downloaded file. Then there should be a *Scripts* folder in your python directory.



2. Install poster library using easy_install.

Open windows command line and go to the *Scripts* folder mentioned above.

In the command line, type `easy_install poster` and press Enter (Figure 3)



Then poster library should be installed.