Tutorial on Using CSTACK: X-Ray Stacks of Distant Quiescent Galaxies

Sam Cutler

May 6, 2017

Abstract

This tutorial will focus on the process of finding X-ray fluxes of distant massive quiescent galaxies using CSTACK. CSTACK is a software that performs a stacking analysis of X-ray count rates for multiple cosmological objects, which we will select to be these distant quiescent galaxies. Selecting these specific galaxies, from a catalog that will be explained later in the introduction, will be the first part of the tutorial. Next we will go over the basics of CSTACK, including what each entry field means and what the specific format files uploaded to CSTACK should have. Lastly we will analyze the CSTACK outputs and examine what the important values to take away are. This should provide you with a basic understanding on how to make X-ray stacks with CSTACK in general.

Introduction

Before we begin the process of programing and using CSTACK, both of which I will describe in detail later in the tutorial, we need to know why we're going through all this trouble. The goal here is to detect an X-ray flux, and CSTACK is a valuable tool in doing this. Significant detection of X-rays in distant galaxies would indicate a supermassive black hole (also referred to as an Active Galactic Nuclei or AGN) in the observed galaxy. High energy X-rays are also indicative of other objects like compact objects or hot plasma, but we are generally not sensitive to these phenomena at such large cosmological distances. The galaxies we will be observing are called "quiescent". Quiescent galaxies are those that have ceased star formation due to either too much heat for gas to condense or too little gas, as cold gas is the fuel for star formation. Often in astronomy, emission lines are detected using spectroscopy, and are characteristic of many stellar object. These emission lines are essentially specific energy photons emitted from transitions in atoms (e.g. oxygen and hydrogen). In Whitaker et al. (2013), centrally-concentrated [OIII] and H-beta emission lines were noticed in distant, massive, quiescent galaxies, which could indicate either an AGN or residual star formation at their centers. Ultimately, analysis of X-ray data from these galaxies would serve to rule out or confirm black holes as the cause for these emission lines.

In this tutorial, we will compile a sample of galaxies from the NEWFIRM Medium-band Survey (NMBS) catalog, which covers a wide area (i.e. the size of the moon in the sky) but only at a shallow depth. From NMBS, the specific field we use is COSMOS (or the Cosmic Evolution Survey). The NMBS COSMOS data can be downloaded from this tar and must be unpacked before they can be used. The COSMOS field probes the evolution of galaxies based on redshift (a measurement of cosmic time). Cosmological redshift, usually denoted by z, is a measurement of how much the wavelength of light coming from a source has been stretched by the expansion of space (like the Doppler Effect). Light that has traveled further will result in a higher redshift, which is why it is used as a measurement of both the age of the universe and distance to galaxies. Obviously these catalogs are not entirely quiescent galaxies, as they include star forming galaxies and foreground stars in our own Milky Way Galaxy, but we can sort for quiescent galaxies based on their rest-frame colors. Rest-frame colors are the relative ratio of fluxes in certain wavelength bands and correlate with the age of a quiescent galaxy. Red galaxies have more flux at a longer wavelengths (think the red-infrared end of the electromagnetic spectrum), and vice versa for blue. Redder galaxies

are older and bluer ones are younger. Besides the age, quiescent galaxies exhibit characteristic rest-frame colors that can be used to sort for them, which we will discuss later on.

What's Inside

When you unpack the COSMOS data linked in the introduction, the resulting folder, "cosmos-1.deblend.v5.1", will have a series of subfolders and files. The README file details what each file contains and what the headers (column names) denote. Another important file is "cosmos-1.deblend.v5.1.cat", the photometric catalog, which contains the *use, id, ra*, and *dec* headers. *Use* refers to the use flag, which indicates good/usable data with a value of 1, and poor data or foreground stars with 0. *Id* is the COSMOS assigned id number for each object. *Ra* and *dec* are the right ascension and declination, essentially a coordinate system for the location of observed objects on the sky, in units of degrees. The remaining 3 subfolders and their important files are:

1. cosmos-1.deblend.sps: Stellar population parameters (SPS)
   (a) cosmos-1.bc03.del.deblend.v5.1.fout
2. cosmos-1.deblend.rfcolors: Rest frame colors
   (a) cosmos-1.deblend.v5.1.153-155.rf
   (b) cosmos-1.deblend.v5.1.155-161.rf
3. cosmos-1.deblend.redshifts: Photometric redshift catalog
   (a) cosmos-1.deblend.v5.1.zout

From 1(a), "cosmos-1.bc03.del.deblend.v5.1.fout", the *lmass* header denotes the logarithm of the stellar mass of the object, where stellar mass is how many times more massive an object or galaxy is than the sun. For example, an object one billion times more massive than our sun has a stellar mass of 1 billion and a log mass of 9 (since $\log_{10}(10^9)= 9$). 2(a) and (b) give the bands used to find the U-V and V-J ratios. These are ratios of specific wavelength bands that help determine the age and star formation rates of galaxies in units of magnitudes, and are defined by

$$\text{U-V}= -2.5\log_{10}(\text{F}_U/\text{F}_V), \text{ and} \tag{1}$$

$$\text{V-J} = -2.5 \log_{10}(\text{F}_V/\text{F}_J), \tag{2}$$

where F is the flux in the given band. In the two rest frame color catalogs, the U, V, and J fluxes are labeled with the headers *L153, L155*, and *L161*, respectively. Lastly, in 3(a), the header *z_peak* gives us our desired redshift values. All these headers will prove very important in a later section, when we work on sorting this large catalog into a selection of distant, massive, quiescent galaxies.

Reading in Catalogs

In order to make all these files available in your Python code, you first must read them in using the *ascii* package from astropy. This program reads in files based on their catalog paths using the command *ascii.read*. In order to use this you have to import the programs using the following lines of code at the beginning of your program (the importance of numpy will be discussed later on):[1]

```
import numpy as np
from astropy.io import ascii
```

Once this has been done, you must define the path of each folder. It is imperative that this is done right, or astropy will not be able to locate your file. If you are unsure of the path of a folder enter the command *pwd* into the terminal while in that folder. In this tutorial, the example path of "cosmos-1.deblend.v5.1" will be "/home/user/tutorial/cosmos-1.deblend.v5.1/". Since there are multiple folders that our files are located in, we must establish multiple paths. To do so, simply assign an arbitrary variable to the desired path, like the one above. Example code to define paths is shown below.

```
pathphoto="/home/user/tutorial/cosmos-1.deblend.v5.1/"
pathcolor="/home/user/tutorial/cosmos-1.deblend.v5.1/cosmos-1.deblend.rfcolors/"
pathz="/home/user/tutorial/cosmos-1.deblend.v5.1/cosmos-1.deblend.redshifts/"
pathsps="/home/user/tutorial/cosmos-1.deblend.v5.1/cosmos-1.deblend.sps/"
```

[1] Code put in LaTeX using \lstlisting command (see brief tutorial at end).

These lines of code define the paths to the photometric catalog, the two color band catalogs, the redshift catalog, and the SPS catalog, respectively.

The final step of this section is to read in the files described in the previous section. The function *ascii.read* takes four parameters: the file location, the data start, the header start, and the delimiter. The file location is simply *path+'filename'* using the paths defined above and the required files in them. The delimiter is what separates each column of the catalogs. In every file here the delimiter is a blank space, which can be given by *delimiter=' '*. The tricky parameters are the header and data stop. Before the data begins there are multiple lines beginning with a pound sign (#) which are comment lines. The data start is the first non-comment line where data is located. Be careful, as a *data_start=0* means the first non-comment line is the data start, whereas *data_start=1* refers to the second such line. Fortunately, all of our data starts are 0. The header start is the first comment line where the headers (i.e. column titles) are labeled. This is denoted with *header_start=line number*. This can be confusing, so I've included two examples.

```
1 # id z_spec z_a z_m1 chi_a z_p chi_p z_m2 odds l68 u68  l95 u95  l99 u99  nfilt q_z z_peak
2 # EAZY v1.1.9 (Jun. 7, 2010)
3 1. -1.000000 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99
4 2. 0.251800 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99
5 3. 0.519200 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99 -99
```

Figure 1: First few lines of cosmos-1.deblend.v5.1.zout file.
Headers appear on first comment line, so *header_start=0*.
Data begins on the first non-commented line, so *data_start=0*.

```
 1 # FAST version: 0.8d
 2 # Photometric catalog file: cosmos-1.deblend.v5.1.cat
 3 # Photometric redshift file: cosmos-1.deblend.v5.1.zout
 4 # Template error function: TEMPLATE_ERROR.fast.v0.2
 5 # AB_ZP:         25.00
 6 # Library:       Bruzual & Charlot (2003)
 7 # SFH:           Delayed exponential SFH: SFR ~ t exp(-t/tau)
 8 # Stellar IMF:   Chabrier
 9 # metallicity:   0.020
10 # log(tau/yr):   7.0    - 10.0, in steps of 0.20
11 # log(age/yr):   7.6    - 10.1, in steps of 0.10
12 # A_V:           0.0    -  4.0, in steps of 0.10
13 # z:             0.0100 -  4.0000, in steps of 0.0100
14 # Filters:       21  20  19  18 222 134 221 132 131 220 130 129 128  83  92  82  91  81
15 # ltau: log[tau/yr], lage: log[age/yr], lmass: log[mass/Msol], lsfr: log[sfr/(Msol/yr)],
16 # For sfr=0. lsfr is set to -99
17 #    id         z      ltau     metal      lage        Av     lmass      lsfr     lssfr
18       1        -1        -1        -1        -1        -1        -1        -1        -1
19       2        -1        -1        -1        -1        -1        -1        -1        -1
20       3        -1        -1        -1        -1        -1        -1        -1        -1
```

Figure 2: First few lines of cosmos-1.bc03.del.deblend.v5.1.fout file.
Headers appear on 17$^{\text{th}}$ comment line, so *header_start=16*.
Data begins on the first non-commented line, so *data_start=0*.

Once you have all four parameters you simply set an arbitrary variable equal to *ascii.read(file, data_start, header_start, delimiter)*. Below is the code for our example.

```
gen=ascii.read(pathphoto+"cosmos-1.deblend.v5.1.cat",
    data_start=0, header_start=0, delimiter=' ')
uvband=ascii.read(pathcolor+"cosmos-1.deblend.v5.1.153-155.rf",
    data_start=0, header_start=0, delimiter=' ')
vjband=ascii.read(pathcolor+"cosmos-1.deblend.v5.1.155-161.rf",
    data_start=0, header_start=0, delimiter=' ')
redshift=ascii.read(pathz+"cosmos-1.deblend.v5.1.zout",
    data_start=0, header_start=0, delimiter=' ')
params=ascii.read(pathsps+"cosmos-1.bc03.del.deblend.v5.1.fout",
    data_start=0, header_start=16, delimiter=' ')
```

With all files read in we can now proceed with sorting for these quiescent galaxies.

Sorting and Selecting

Now that you are familiar with the contents of the COSMOS data and the various headers and the correct files are read in, we can begin the process

of sorting the data. All the data we need is in the correct form, except for the UV and VJ colors. In order to get these ratios we write code to implement equations (1) and (2) into our program. This is where the *numpy* package we imported earlier comes in handy. Using the command *np.log10()* takes the logarithm of an argument, which is needed in the equations. In order to get just the columns of flux in the U, V, and J bands from the read in files, use brackets. For example, *file['column name']* selects the column starting with the header *column name* from the file read into the variable *file*. This is the same in the general case and is vital in making selections. Putting these pieces of code together gives us an array of each of the UV and VJ color ratios the same length as a column of the parent file. For our example, recall the U, V, and J headers are *L153, L155*, and *L161*. The resulting code is below.

```
UV=-2.5*np.log10(uvband['L153']/uvband['L155'])
VJ=-2.5*np.log10(vjband['L155']/vjband['L161'])
```

From here we are ready to make selections of both older and younger distant massive quiescent galaxies.

The first part of this is to obtain the required U-V and V-J selections which represent quiescent galaxies. Whitaker et al. (2012)defines a quiescent galaxy as one that falls within the region contained by

$$\text{U-V} > 0.8 \times (\text{V-J}) + 0.7, \tag{3}$$

$$\text{U-V} > 1.3, \text{ and} \tag{4}$$

$$\text{V-J} < 1.5. \tag{5}$$

Figure 3 demonstrates this selection region. This plot, from Whitaker et al. (2013), also shows a dotted line perpendicular to the line formed by equation (3) which splits the quiescent region in to a portion of red and blue data points. The red data points represent older quiescent galaxies, while the blue represent more recently quenched ones. The equation for the dotted line as given in Whitaker et al. (2013) is as follows:

$$\text{U-V} = -1.25 \times (\text{V-J}) + 2.85. \tag{6}$$

If the U-V colors are below (less than) this line, the galaxies are younger and if the colors are above (greater than), the galaxies are older.

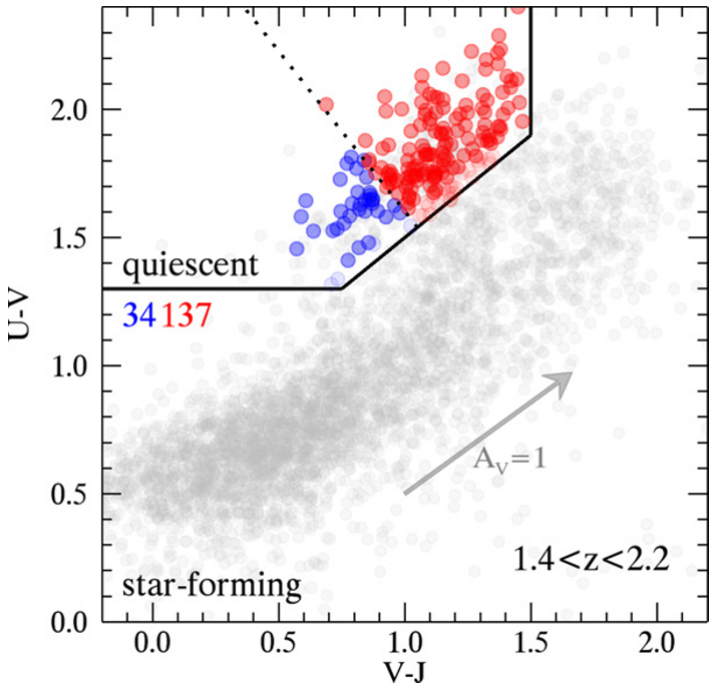


Figure 3: U-V vs V-J colors plot with quiescent region highlighted. Red and blue data points represent older and younger quiescent galaxies, respectively. The distinction between older and younger is given by the dotted line separating the two colors.

Whitaker et al. (2012) also details the redshift at which these galaxies are present. The selection here is $1.4 < z < 2.2$. Since we want only massive quiescent galaxies, we restrict the log mass (lmass) to 10.5 <lmass< 12. Our final selection is to ensure we only take data that has a use flag equal to one, which if you recall, means the data is good and is a galaxy. The code for making a selection is to set a variable equal to your restriction in parentheses. For example, *select = (gen['use '] == 1)* makes a selection for all values of the "use" column of the catalog assigned to "gen" that are equal to 1. To make multiple selections, close all of your selections in square brackets and separate each individual one with ampersands (&). For example, *select = [(gen['use '] == 1) & (redshift['z_peak'] > 1.4)]* selects all data with a use flag of 1 and a redshift greater than 1.4. Applying this to all the selections

we want to make, we can select younger and older massive quiescent galaxies using the following code.

```
young=[(gen['use'] == 1) & (redshift['z_peak']>1.4) &
    (redshift['z_peak']<2.2) & (params['lmass']>10.5) &
    (params['lmass']<12) & (UV > (0.8 * VJ + 0.7)) & (UV > 1.3) &
    (VJ < 1.5) & (UV < (-1.25 * VJ + 2.85))]
old=[(gen['use'] == 1) & (redshift['z_peak']>1.4) &
    (redshift['z_peak']<2.2) & (params['lmass']>10.5) &
    (params['lmass']<12) & (UV > (0.8 * VJ + 0.7)) & (UV > 1.3) &
    (VJ < 1.5) & (UV > (-1.25 * VJ + 2.85))]
```

Once our selections have been made, we can write the selected data to an output file, which we will later edit to fit the correct CSTACK format. To do this, we can use the *f.write()* command, which writes a line into an external file that you create using *f.open('cosmosyoung.txt', 'w+')*. The first part of *f=open()* names the output file, while the second part tells the program what you want to do to this output file, i.e. *w+* is write, *a* is append, etc. In our output file we want columns of *id, ra, dec, z, U-V, V-J,* and *lmass*. In order to create headers for these columns, we put the header names in single quotes plus a space in quotes as the delimiter in between each header in the argument of *f.write()*. In order to specify that this line of headers is a comment line and not data, start the line with a pound sign. Lastly, put $\backslash n$ after the last item to tell the program to move to a new line. The resulting code will look like this:

```
f=open('cosmosyoung.txt', 'w+')
f.write('# id'+' '+'ra'+' '+'dec'+' '+'z'+' '+'U-V'+' '+'V-J'+'
    '+'lmass \n')
```

To write our data to the same file, we must loop over every the entire length of our sorted catalog. To do this we use a for loop. Writing a for loop is as simple as writing *for i in range(len(gen['young'])):*. This loop occurs over the ith row in the range of the length of the "gen" catalog with the "young" selection applied. This ensures that the program will loop through all rows of data in our selection and print all of the data we selected. It does not matter which file we loop over, as they should all be the same length with the selection applied. Indented on the line after the for loop declaration include the *f.write()* command to write all of the selected data to your opened file.

The code to write the data from each file is *str(file['header'][selection][i])*. You separate each of these terms in the same way you did the headers above, including the $\backslash n$ at the end. Doing this for both older and younger galaxies in our example yields the following code.

```
f=open('cosmosold.txt', 'w+')
f.write('# id'+' '+'ra'+' '+'dec'+' '+'z'+' '+'U-V'+' '+'V-J'+'
    '+'lmass \n')
for i in range(len(gen[young])):
   f.write(str(gen['id'][young][i])+' '+str(gen['ra'][young][i])+'
       '+str(gen['dec'][young][i])+'
       '+str(redshift['z_peak'][young][i])+' '+str(UV[young][i])+'
       '+str(VJ[young][i])+' '+str(params['lmass'][young][i])+"\n")
f.close()

f=open('cosmosyoung.txt', 'w+')
f.write('# id'+' '+'ra'+' '+'dec'+' '+'z'+' '+'U-V'+' '+'V-J'+'
    '+'lmass \n')
for i in range(len(gen[old])):
   f.write(str(gen['id'][old][i])+' '+str(gen['ra'][old][i])+'
       '+str(gen['dec'][old][i])+'
       '+str(redshift['z_peak'][old][i])+' '+str(UV[old][i])+'
       '+str(VJ[old][i])+' '+str(params['lmass'][old][i])+"\n")
f.close()
```

CSTACK Basics

CSTACK is an online software designed to obtain the total X-ray counts of a group of individual objects using stacking analyses. According to Gawiser et al. (2010), stacking analysis is used to detect faint sources by estimating a summed signal from many individual objects. Since the signals we are receiving have background noise, many objects that only emit a faint signal, such as some distant AGNs, can get lost in the noise. However, because we know the location of these objects (or objects at the same location) accurately based on their signals at other wavelengths, we can average the noisy data at each known position, resulting in an aggregate count rate. This aggregate count rate is our stacked count rate. CSTACK does this for X-ray signals using data from the Chandra X-ray Observatory. Chandra is a telescope which detects X-rays from various sources, including exploded stars and black holes. As such, Chandra's X-ray data is very useful in identifying

AGNs in our selected galaxies.

Another useful aspect of CSTACK is that it can make these X-ray stacks for many different fields and catalogs, but most importantly, in our case, COSMOS. Using the link above to access the CSTACK website will give you a page with four main links. The specific software version we will be using is "CSTACK V4.32@UC San Diego". There will be a pop-up asking for a username and password, but for the public Chandra data they are both "guest". On this page the first important inputs are the following (Fig. 4).

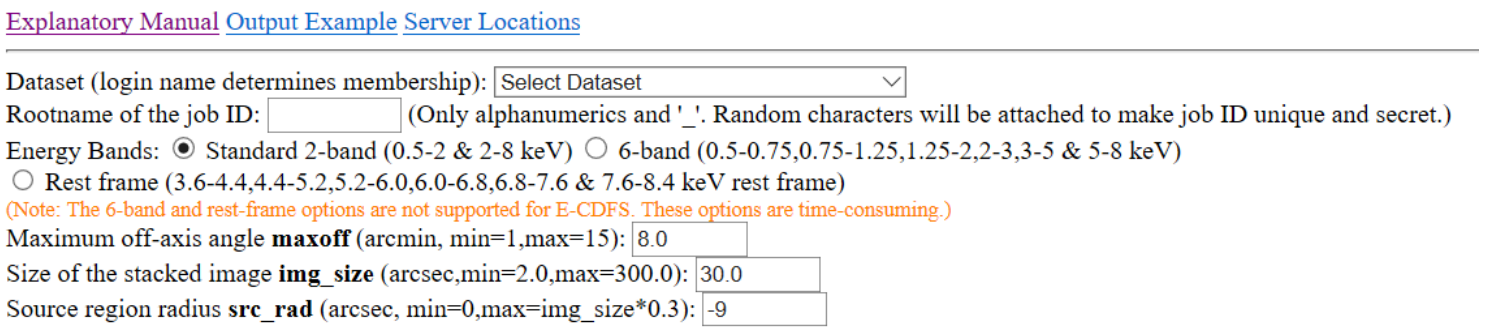


Figure 4: First CSTACK inputs

The important fields here are the Dataset and job ID. The dataset is the field that the inputted data was observed in (i.e. COSMOS). There are two COSMOS options, but the only public one is C-COSMOS, so select that. The job ID is just what you want to call the output files, so for our example we will use *"example_young"* and *"example_old"* for the recent and older quiescent galaxies. The next important entries are the following (Fig. 5).

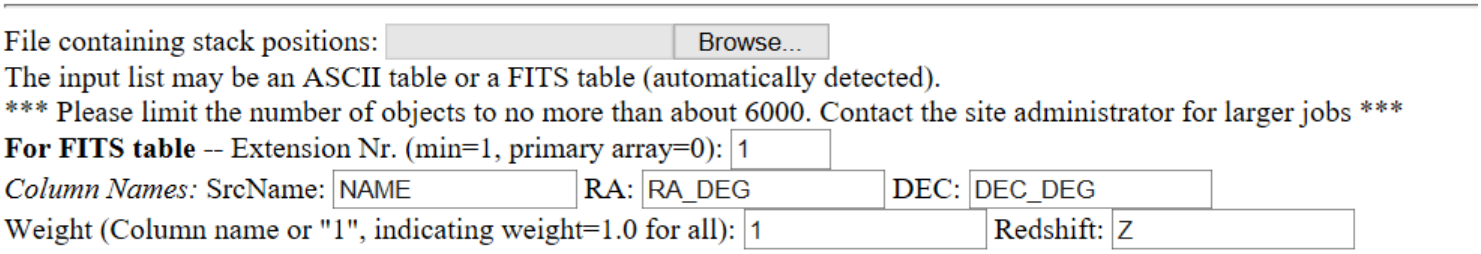


Figure 5: Second CSTACK inputs

This is where you select the file to upload to CSTACK and tell it which header points to the id, right ascension, declination, weight, and redshift data[2]. This is where the input files get tricky.

[2]CSTACK automatically detects the files as ASCII tables.

CSTACK Formatting

CSTACK is very specific about the ordering of columns and headers in the files uploaded to it. The columns have to be in the order of id, right ascension, declination, weight, and redshift exactly. To get the data in this order you have to write another Python program that reorders the columns and adds a weight column. This is relatively easy, and you can just follow the above tutorial sections, just without making a selection. Basically all you have to do is read in the *cosmosold.txt* and *cosmosyoung.txt* files you made before and reprint them to new files (call them *cstackyoung.txt* and *cstackold.txt*) with only the columns above in the right order. We won't weight any of the data, so just set a variable *weight=1* and include in in your output string as *+str(weight)+*. The program will look like this.

```
import numpy as np
from astropy.io import ascii

path="/home/user/tutorial/"

old = ascii.read(path+'cosmosold.txt',
    data_start=0,header_start=0,delimiter=' ')

young = ascii.read(path+'cosmosyoung.txt',
    data_start=0,header_start=0,delimiter=' ')

weight = 1.0
f=open('cstackold.txt', 'w+')
f.write("# id"+' '+"ra"+' '"dec"+' '+"weight"+' '+"z\n")
for i in range(len(old['id'])):
   f.write(str(old['id'][i])+' '+str(old['ra'][i])+'
       '+str(old['dec'][i])+' '+str(weight)+'
       '+str(old['z'][i])+"\n")
f.close()

f=open('cstackyoung.txt', 'w+')
f.write("# id"+' '+"ra"+' '"dec"+' '+"weight"+' '+"z\n")
for i in range(len(young['id'])):
   f.write(str(young['id'][i])+' '+str(young['ra'][i])+'
       '+str(young['dec'][i])+' '+str(weight)+'
       '+str(young['z'][i])+"\n")
f.close()
```

Thus for the file upload you select *cstackold.txt* or *cstackyoung.txt*, depending on which galaxies you want to stack and input the header names used in the code into those boxes.

CSTACK Outputs

CSTACK can take a while to run so when you do obtain an output look for a few things. On the "results.html" page you should see the following output screen.

Stacked Images, Count Rates & Bootstrap Percentiles

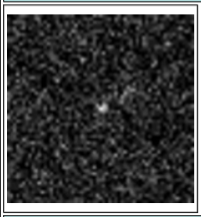
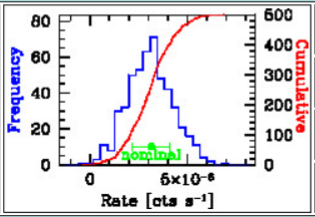
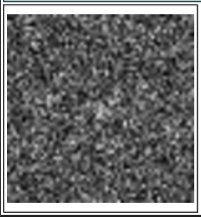
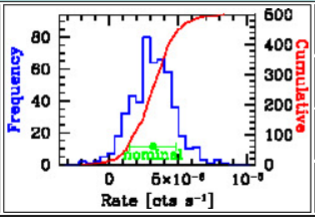
Stacking Analysis Summary Band = 500_2000			
		**Mean CR +- ct stat err**	**Detailed Results**
		3.14e-06 +- 9.60e-07	stat_cts_500_2000.out
		Bootstrap 5|16|50|84|95% percentiles	boot_histo_500_2000.out
		1.30e-06|1.92e-06|3.08e-06|4.23e-06|5.09e-06	boot_sorteddump_500_2000.out
Stacking Analysis Summary Band = 2000_8000			
		**Mean CR +- ct stat err**	**Detailed Results**
		3.17e-06 +- 1.68e-06	stat_cts_2000_8000.out
		Bootstrap 5|16|50|84|95% percentiles	boot_histo_2000_8000.out
		6.32e-07|1.54e-06|3.11e-06|4.58e-06|5.70e-06	boot_sorteddump_2000_8000.out

Figure 6: CSTACK results screen for recently quenched distant massive galaxies.

The top box represents data from low energy (500-2000 electron volts) X-ray photons and the bottom is the high energy (2000-8000 eV) photons. The two images on the far left are FITS images ('Flexible Image Transport System'), which is the standard image format for astronomy, and give a visual representation of how clear or noisy the signal is. A clear dot in the middle represents a good, not noisy X-ray count rate (see the image in the upper left in Fig. 6). A fuzzy/static image means the signal is very noisy and there is not a very precise count rate. The two plots are histograms of the frequency of detected count rates from each individual source object from which the mean count rate is measured. The important values to take away from this page are the mean count rates and their error. These count rates are the

stacked X-ray count rates for all of the objects and the error is essentially the amount of noise for these signals, or the error from performing a stacking analysis. A good test to see if your X-ray signal is clear is to divide the count rate by its error, which is called a signal to noise ratio. If it is greater than one then your signal is stronger than the surrounding background noise, meaning you have a good X-ray count rate. The six files in this table under "Detailed Results" provide more in depth values and statistics used in the stacking process.

The most important output result is the "stat_cts_[BAND].out" file, where the [BAND] is replaced with either "500_2000" or "2000_8000" depending on which energy band the data is for. This file contains data for each individual object, but the most important parts are the total values at the end of the file (see Fig. 7).

```
!          ============================================================
!Accepted: 263  Near_src: 18  Out: 0   Total: 281
!Final Stacked Rate:  3.14e-06 +/-  9.60e-07 (Counting Statistics Error)
!Total Exposure:  2.36e+07 [sec]
!Please refer to bootstrap results for  better error analysis.
! Notes from Dataset (if any):
!
! This is based on the Chandra COSMOS Data 1.8 Ms.
!
```

Figure 7: Final results of stat_cts_500_2000.out file for recently quenched distant massive galaxies.

The "Final Stacked Rate" is the same as the value reported on the results screen (Fig. 6). The first row shows the number of galaxies stacking analysis was done on. The "Total" (281) and "Accepted" (263) are the number of galaxies in the input file (i.e. the number of galaxies that you selected to be recently quenched) and the number of galaxies that CSTACK actually performed the stacking analysis on. The "Near_src" (18) refers to the number of galaxies in your file that are near a known X-ray source, and as such are not counted in the stacking. The "Out" (0) refers the number of galaxies outside of the area of the sky which Chandra took data from, and these are also not counted. Exposure time[3] refers to how long a telescope (in this case Chandra) spent taking data from a cosmological object. Longer exposure times are better, as they provide more X-ray photons for a more accurate

[3]Exposure time is usually reported in megaseconds, though here it is in seconds (1 Ms= 10^6 s).

count rate. The "Total Exposure" in the file is the sum of all the individual exposure times of each accepted object. As such, to get the average exposure time of the data you need to divide the total exposure time by the number of accepted galaxies.

The values in this file are essentially the goal of this entire tutorial. To convert the count rate (and error/noise) to a flux you need an energy conversion factor (ECF), which are in units of erg/cm^2[4]. Multiplying the count rates you obtain through stacking by an ECF will give you the X-ray flux of the group of galaxies the stacking was done on. ECFs vary by dataset and energy band, and for COSMOS the ECFs are 7.50×10^{-11} for 500-2000 eV photons and 3.06×10^{-11} for 2000-8000 eV photons. Now we have completed our goal of finding X-ray fluxes for recent and older distant massive quiescent galaxies. Below are the "results.html" page and "stat_cts_[BAND].out" file for the older distant massive quiescent galaxies (and the "stat_cts_2000_8000.out" file for the younger galaxies), so that you can compare your final values (Figs. 8-11).

```
!          ================================================================
!Accepted: 262  Near_src: 19  Out: 0   Total: 281
!Final Stacked Rate:  3.17e-06 +/-  1.68e-06 (Counting Statistics Error)
!Total Exposure:  2.47e+07 [sec]
!Please refer to bootstrap results for  better error analysis.
! Notes from Dataset (if any):
!
! This is based on the Chandra COSMOS Data 1.8 Ms.
```

Figure 8: Final results of stat_cts_2000_8000.out file for recently quenched distant massive galaxies.

[4]An erg is a unit of energy equal to 10^{-7} Joules.

Stacked Images, Count Rates & Bootstrap Percentiles

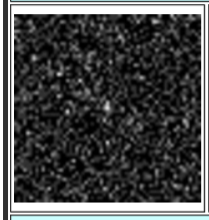
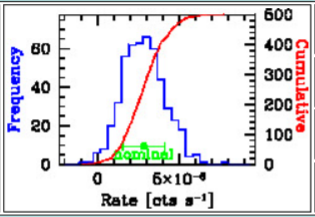
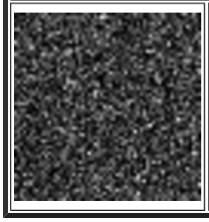
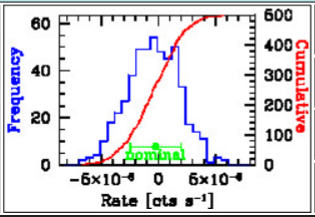
Stacking Analysis Summary Band = 500_2000		
 	**Mean CR +- ct stat err**	**Detailed Results**
	2.83e-06 +- 1.29e-06	stat_cts_500_2000.out
	Bootstrap 5|16|50|84|95% percentiles	boot_histo_500_2000.out
	9.12e-07|1.59e-06|2.79e-06|4.11e-06|5.03e-06	boot_sorteddump_500_2000.out
Stacking Analysis Summary Band = 2000_8000		
 	**Mean CR +- ct stat err**	**Detailed Results**
	-2.46e-07 +- 2.24e-06	stat_cts_2000_8000.out
	Bootstrap 5|16|50|84|95% percentiles	boot_histo_2000_8000.out
	-4.19e-06|-2.68e-06|-4.55e-07|1.79e-06|3.20e-06	boot_sorteddump_2000_8000.out

Figure 9: CSTACK results screen for older distant massive quiescent galaxies.

```
!          ================================================================
!Accepted: 151  Near_src: 12  Out: 0   Total: 163
!Final Stacked Rate:  2.83e-06 +/-  1.29e-06 (Counting Statistics Error)
!Total Exposure:  1.27e+07 [sec]
!Please refer to bootstrap results for  better error analysis.
! Notes from Dataset (if any):
!
! This is based on the Chandra COSMOS Data 1.8 Ms.
```

Figure 10: Final results of stat_cts_500_2000.out file for older distant massive quiescent galaxies.

```
!          ================================================================
!Accepted: 148  Near_src: 15  Out: 0   Total: 163
!Final Stacked Rate: -2.46e-07 +/-  2.24e-06 (Counting Statistics Error)
!Total Exposure:  1.32e+07 [sec]
!Please refer to bootstrap results for  better error analysis.
! Notes from Dataset (if any):
!
! This is based on the Chandra COSMOS Data 1.8 Ms.
```

Figure 11: Final results of stat_cts_2000_8000.out file for older distant massive quiescent galaxies.

Addendum: Syntax Highlighting in LaTeX

As you may be able to tell, this tutorial was written in LaTeX, which is a word processing/typesetting software that makes scientific publication standard documents. If you are not familiar with the basics of LaTeX there is extensive documentation online; this tutorial will not cover the basics. However, if you are versed in LaTeX, there is a command that will allow you to write code with syntax highlighting like I have done above. To do this write the following in the preamble.

```
\usepackage{listings}
\lstset{frame=tb,
   language=Python,
   aboveskip=3mm,
   belowskip=3mm,
   showstringspaces=false,
   columns=flexible,
   morekeywords={as, write, close},
   deletekeywords={del},
   basicstyle={\small\ttfamily},
   numbers=none,
   identifierstyle=\color{black},
   numberstyle=\tiny\color{gray},
   keywordstyle=\color{blue},
   commentstyle=\color{cyan},
   stringstyle=\color{pink},
   breaklines=true,
   breakatwhitespace=true,
   tabsize=3
}
```

The above are the set parameters I used for my document, though you can easily change these based on what language or syntax highlighting you want. The important parameters are the language, more/deletekeywords, and all of the styles. The language is obviously the coding language that you are typing in (e.g. Python in this tutorial). This affects what keywords and standard formatting are applied to the syntax highlighting. The more/deletekeywords parameters either add extra keywords to the standard set of the language or deletes keywords from the set. The style parameters affect the size, color, and font of certain elements of the code, like keywords or strings. To actually input the code into your LaTeX file do the following.

```
\begin{lstlisting}
   Type code here
\end{lstlisting}
```

The code you type will automatically have its syntax highlighted.