

Guide to Analyzing Flares

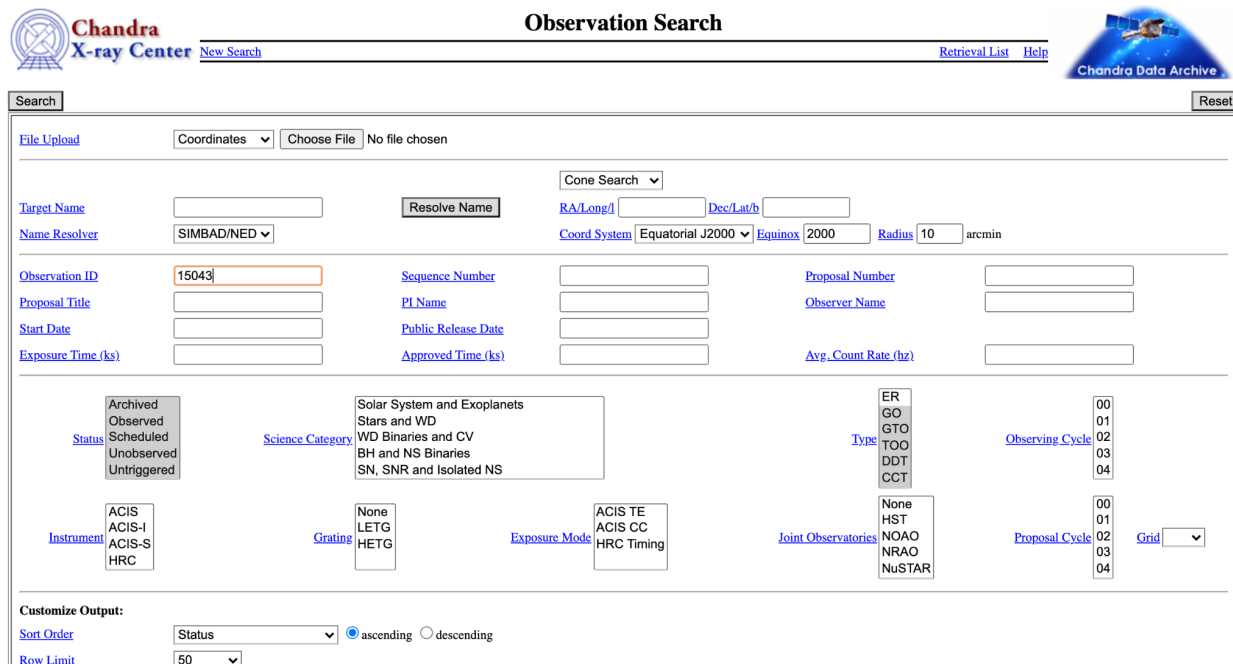
Let's learn how to analyze Sgr A* flares using CIAO! We will go over how to process raw data downloaded from the Chandra Archive (either using ChaSeR or in the command line) and how to run the Bayesian Blocks code adapted for Sgr A* analysis by Chloe Robeyns and Elisa Jacquet. Before starting this guide make sure you have the latest version of [Anaconda](#) and [CIAO](#) installed.

If you have a Windows machine, you will have to install Anaconda for Ubuntu first. You can find it in your Windows Store by searching for Ubuntu and then installing Anaconda:

<https://www.digitalocean.com/community/tutorials/how-to-install-anaconda-on-ubuntu-18-04-quickstart>

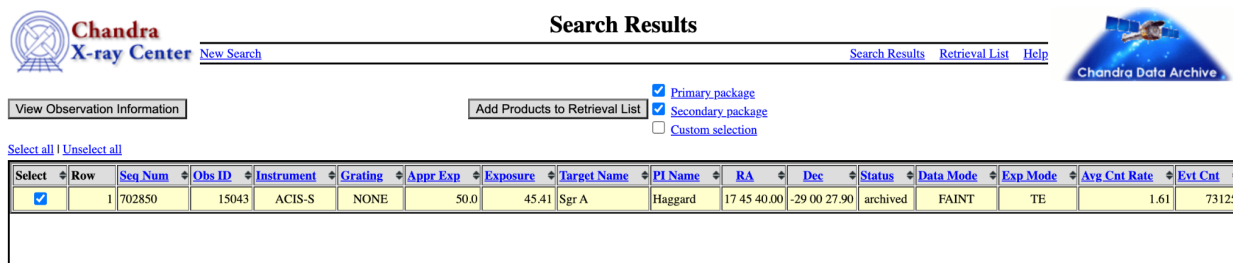
1. Downloading data using ChaSeR

Access ChaSeR here: <https://cda.harvard.edu/chaser/>. Input your ObsID of interest (e.g. 15043) in the Observation ID field and click Search.



The image shows the 'Observation Search' interface of the Chandra X-ray Center. It features a search bar with '15043' entered, and various filters for coordinates, cone search, and observation details. The 'Status' filter is set to 'Archived'. The 'Science Category' is 'Solar System and Exoplanets'. The 'Instrument' is 'ACIS-S'. The 'Grating' is 'None'. The 'Exposure Mode' is 'ACIS TE'. The 'Joint Observatories' are 'None'. The 'Observing Cycle' is '00'. The 'Proposal Cycle' is '00'. The 'Customize Output' section shows 'Sort Order' as 'Status' and 'Row Limit' as '50'.

A Search Results page should pop up that looks something like this:

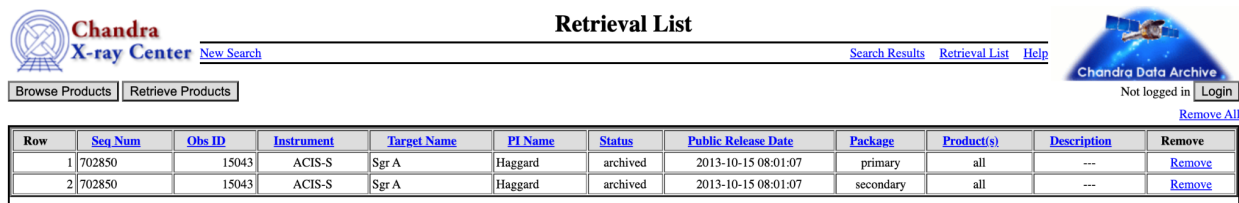


The image shows the 'Search Results' interface of the Chandra X-ray Center. It displays a table of search results for observation ID 15043. The table has columns for 'Select', 'Row', 'Seq Num', 'Obs ID', 'Instrument', 'Grating', 'Appr Exp', 'Exposure', 'Target Name', 'PI Name', 'RA', 'Dec', 'Status', 'Data Mode', 'Exp Mode', 'Avg Cnt Rate', and 'Evt Cnt'. The first row shows a single result for observation 15043, which is archived and has a data mode of 'FAINT'.

Select	Row	Seq Num	Obs ID	Instrument	Grating	Appr Exp	Exposure	Target Name	PI Name	RA	Dec	Status	Data Mode	Exp Mode	Avg Cnt Rate	Evt Cnt
<input checked="" type="checkbox"/>	1	702850	15043	ACIS-S	NONE	50.0	45.41	Sgr A	Haggard	17 45 40.00	-29 00 27.90	archived	FAINT	TE	1.61	73125

Here, we've found one observation of Sgr A* archived under ObsID 15043. If you scroll to the right you should find additional information such as the observation start date (2013-09-14

00:03:46) and the science category (Active Galaxies and Quasars). Make sure the observation you want is selected and that the Primary package and Secondary package are selected. Click “Add Products to Retrieval List”.



Row	Seq Num	Obs ID	Instrument	Target Name	PI Name	Status	Public Release Date	Package	Product(s)	Description	Remove
1	702850	15043	ACIS-S	Sgr A	Haggard	archived	2013-10-15 08:01:07	primary	all	---	Remove
2	702850	15043	ACIS-S	Sgr A	Haggard	archived	2013-10-15 08:01:07	secondary	all	---	Remove

You can go back and search for more observations by going back to “Browse Products” and adding them to the retrieval list so that you download them all at the same time. When you’re ready to download, click “Retrieve Products”. Since it takes a couple minutes to retrieve everything and queue the download, I usually put my email in and wait for a notification. When you get the Chandra Data Notification email, click the link and download the data.

The data will be downloaded in a .tar file so to extract the contents open a terminal window and type `tar xvf file_name` (e.g. `tar xvf 23665`). Now all the files will have a .gz extension so to decompress type `gunzip file_name` (e.g. `gunzip 23665`). To decompress all the files in a directory, navigate to that directory using `cd directory_name` and then type `gunzip *.gz`.

2. Downloading data using the command line

To download data from the command line, start the CIAO software (https://cxc.harvard.edu/ciao/threads/ciao_startup/) and type `download_chandra_obsid` followed by the ObsID you want to download. For example, to download ObsID 23665 I would type `download_chandra_obsid 15043`.

3. Copy data to McGill server

This step is applicable only if you’re using the McGill server. To copy your downloaded data to irulan using the command line type

```
scp -r local_directory server_directory
```

Note the space between the two directories

(e.g. `scp -r ~/chandra_data/15043`

`user@physics.mcgill.ca:/home/zark/user/chandra_data`).

You can use `pwd` to print your current working directory.

4. Reprocess the data

To keep everything organized I like to move all of my downloaded files into a folder/directory (I’ll be using these two terms interchangeably) named “chandra_data”. To begin, open a terminal window and start the CIAO software. Use `cd` to change directories to where your data is (e.g. `cd ~/chandra_data/15043`) . Type `ls` to list the contents of the directory. Your directory

should contain two folders named “primary” and “secondary”. Stay in this directory and type `punlearn ardlb` and then [chandra_repro](#) which reprocesses the data. It will create a new folder named “repro”. Navigate to that directory with `cd repro`.

5. Define the regions

We’ll be working with the `evt2.fits` named something like “`acisf15043_repro_evt2.fits`”. Before we visualize the data, we’ll start by using [wavdetect](#) on our data to find our sources. First, we need to create a cropped image so that `wavdetect` doesn’t have to search the whole field.

```
punlearn dmcopy

dmcopy infile="acisf15043_repro_evt2.fits[EVENTS][bin
x=3992.71:4174.71:1,y=3991.24:4173.24:1][energy=2000:8000]"
outfile="15043_repro_2-8keV_cropped.fits" clobber=yes
```

Here we have used a space between lines to indicate separate lines in input. An example of what this looks like in the terminal is as follows:

```
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ punlearn dmcopy
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ dmcopy infile="acisf15043_repro_evt2.fits[EVENTS][bin x=3992.71:4174.71:1,y=3991.24:4173.24:1][energy=2000:8000]" outfile="15043_repro_2-8keV_cropped.fits" clobber=yes
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$
```

Then, we want to produce a point-spread function (psf) map of our cropped image.

```
punlearn mkpsfmap

mkpsfmap infile="15043_repro_2-8keV_cropped.fits"
outfile="15043_repro_2-8keV_psfmap.fits" energy=3.8 ecf=0.393 clobber=yes
```

```
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ punlearn mkpsfmap
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ mkpsfmap infile="15043_repro_2-8keV_cropped.fits" outfile="15043_repro_2-8keV_psfmap.fits" energy=3.8 ecf=0.393 clobber=yes
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$
```

With this, we’re ready to run `wavdetect` on our data!

```
punlearn wavdetect

pset wavdetect infile="15043_repro_2-8keV_cropped.fits"

pset wavdetect psffile="15043_repro_2-8keV_psfmap.fits"

pset wavdetect outfile="src.fits"

pset wavdetect scellfile="15043_repro_2-8keV_scell.fits"
```

```

pset wavdetect imagefile="15043_repro_2-8keV_img.fits"

pset wavdetect defnbkgfile="15043_repro_2-8keV_nbkg.fits"

pset wavdetect regfile="src.reg"

pset wavdetect scales="1.0 2.0 4.0 8.0 16.0"

pset wavdetect sigthresh=1.e-06

pset wavdetect clobber=yes

wavdetect

Input file name (15043_repro_2-8keV_cropped.fits):
Output source list file name (src.fits):
Output source cell image file name (15043_repro_2-8keV_scell.fits):
Output reconstructed image file name (15043_repro_2-8keV_img.fits):
Output normalized background file name (15043_repro_2-8keV_nbkg.fits):
wavelet scales (pixels) (1.0 2.0 4.0 8.0 16.0):
Image of the size of the PSF (15043_repro_2-8keV_psfmap.fits):

```

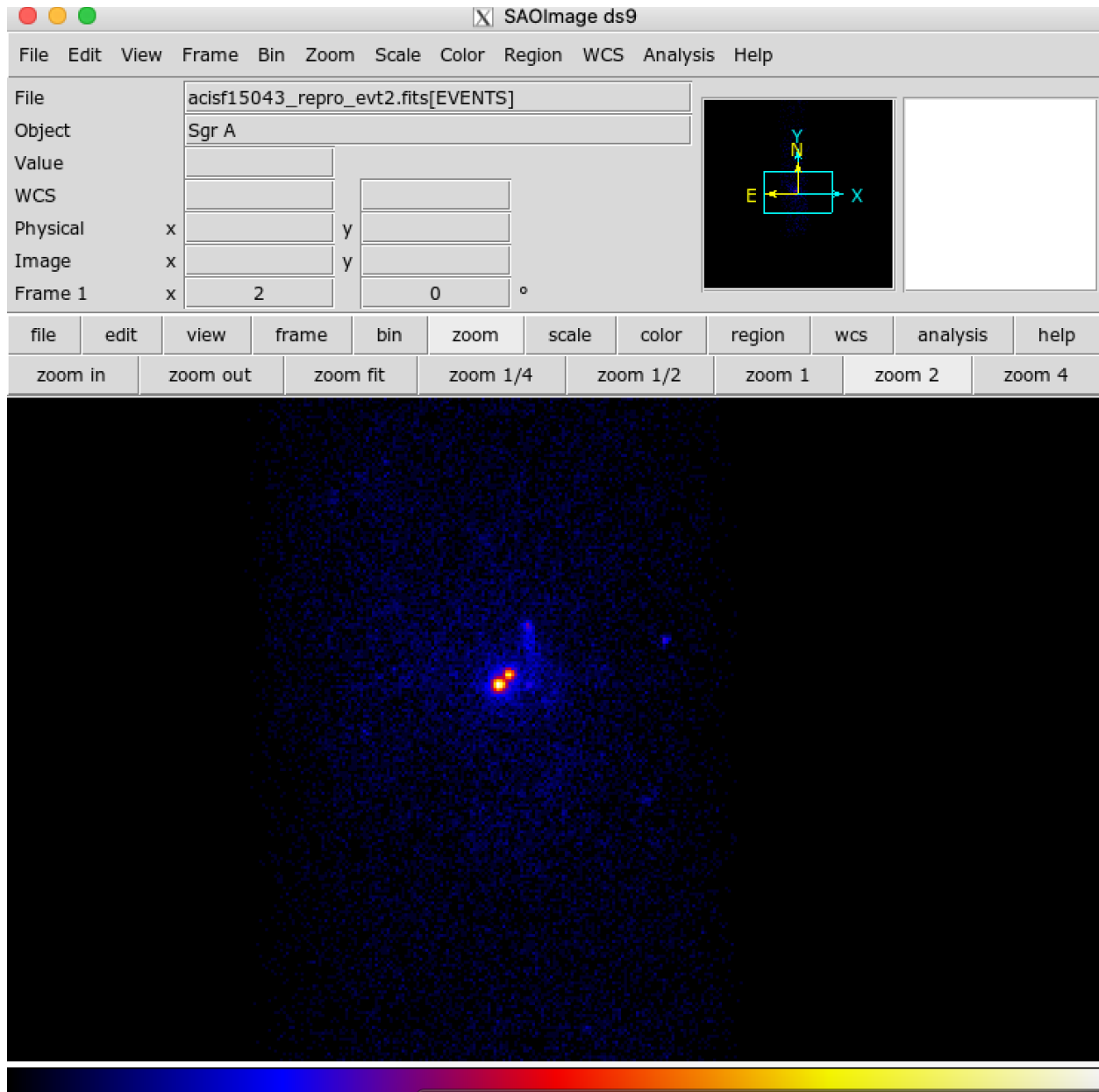
```

(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ punlearn wavdetect
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect infile="15043_repro_2-8keV_cropped.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect psffile="15043_repro_2-8keV_psfmap.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect outfile="src.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect scellfile="15043_repro_2-8keV_scell.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect imagefile="15043_repro_2-8keV_img.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect defnbkgfile="15043_repro_2-8keV_nbkg.fits"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect regfile="src.reg"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect scales="1.0 2.0 4.0 8.0 16.0"
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect sigthresh=1.e-06
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ pset wavdetect clobber=yes
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ wavdetect
Input file name (15043_repro_2-8keV_cropped.fits):
Output source list file name (src.fits):
Output source cell image file name (15043_repro_2-8keV_scell.fits):
Output reconstructed image file name (15043_repro_2-8keV_img.fits):
Output normalized background file name (15043_repro_2-8keV_nbkg.fits):
wavelet scales (pixels) (1.0 2.0 4.0 8.0 16.0):
Image of the size of the PSF (15043_repro_2-8keV_psfmap.fits):
(ciao413) mlariviere@irulan:~/NewObs/15043/repro$ █

```

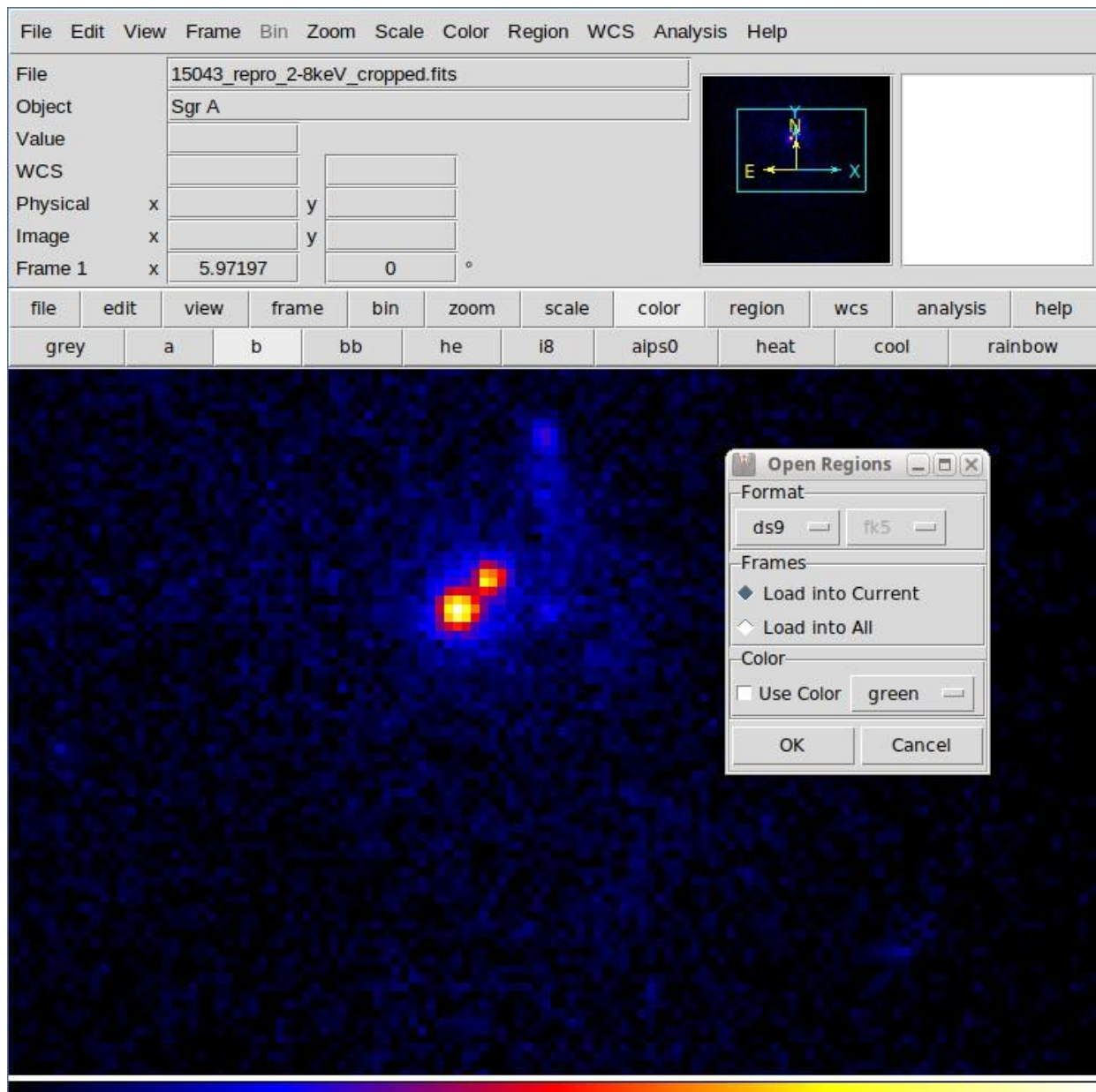
Quick note: when the terminal shows the prompts (shown in blue text) after running `wavdetect`. You just need to check that the file name is ok, and click enter. This should generate a file containing all the sources in the cropped image named "src.reg".

We'll use SAOImage [ds9](#) to visualize the data. In the terminal type `ds9`
`file_name_cropped.fits & (e.g. ds9 acisf15043_repro_2-8keV_cropped.fits`
`&)`. Or if it doesn't work you can just type `ds9` in the terminal then `File → Open →`
`~/chandra_data/15043/repro/acisf15043_repro_2-8keV_cropped.fits` and
click "OK". A window should pop up that looks something like this:

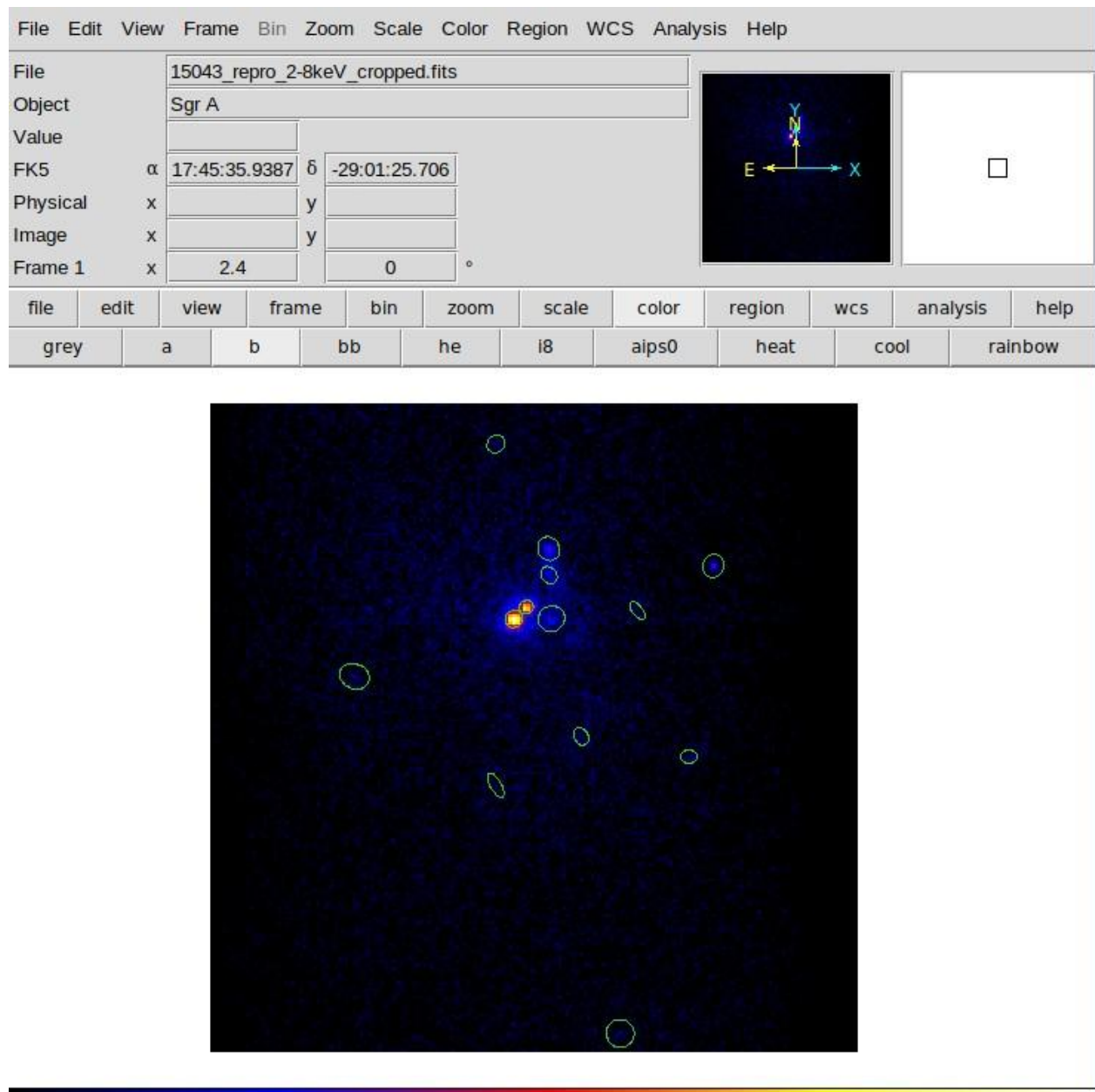


I've changed the colour map with `color → map` and the scale with `scale → log`. Note that you might have to move the field around and zoom a little bit to see the structures.

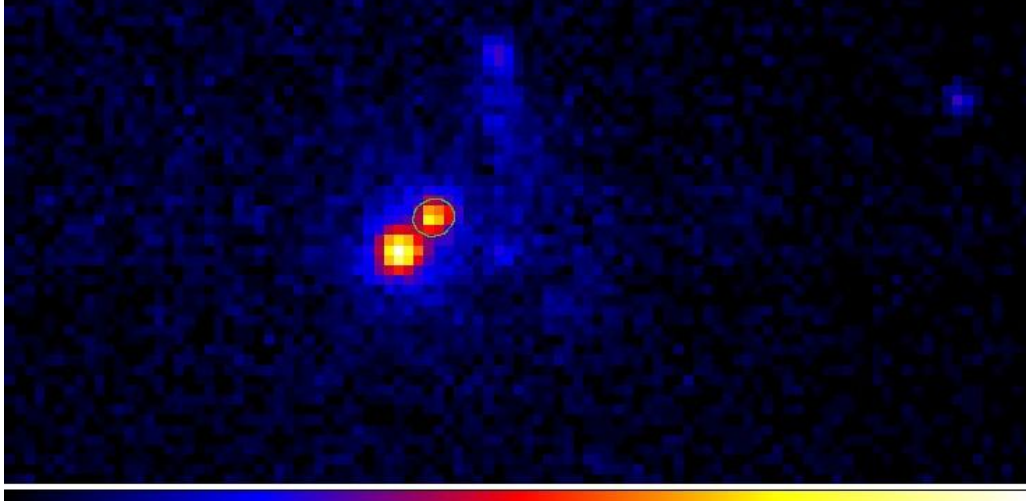
To load the regions file with the sources we found using `wavdetect`, go to `region` → `open regions` → select `src.reg` and click “OK”. A window should open, and all you need to do is click on “OK”.



After loading the in the regions file you should get something that looks like this:



The green ellipses are all the sources that `wavdetect` has found. Make sure that you are zoomed out enough to see the entire field. This is crucial because we need to get rid of all the sources that are not Sgr A*! In order to do so, click on edit → region and then, one by one, click on the middle of the regions you want to delete and press delete. In the end, you should have only ONE region left in the field which is encircling Sgr A*. Double-check it is the correct region by comparing to the known coordinates of Sgr A*: RA=17:45:40.0409, DEC=-29:00:28.118.



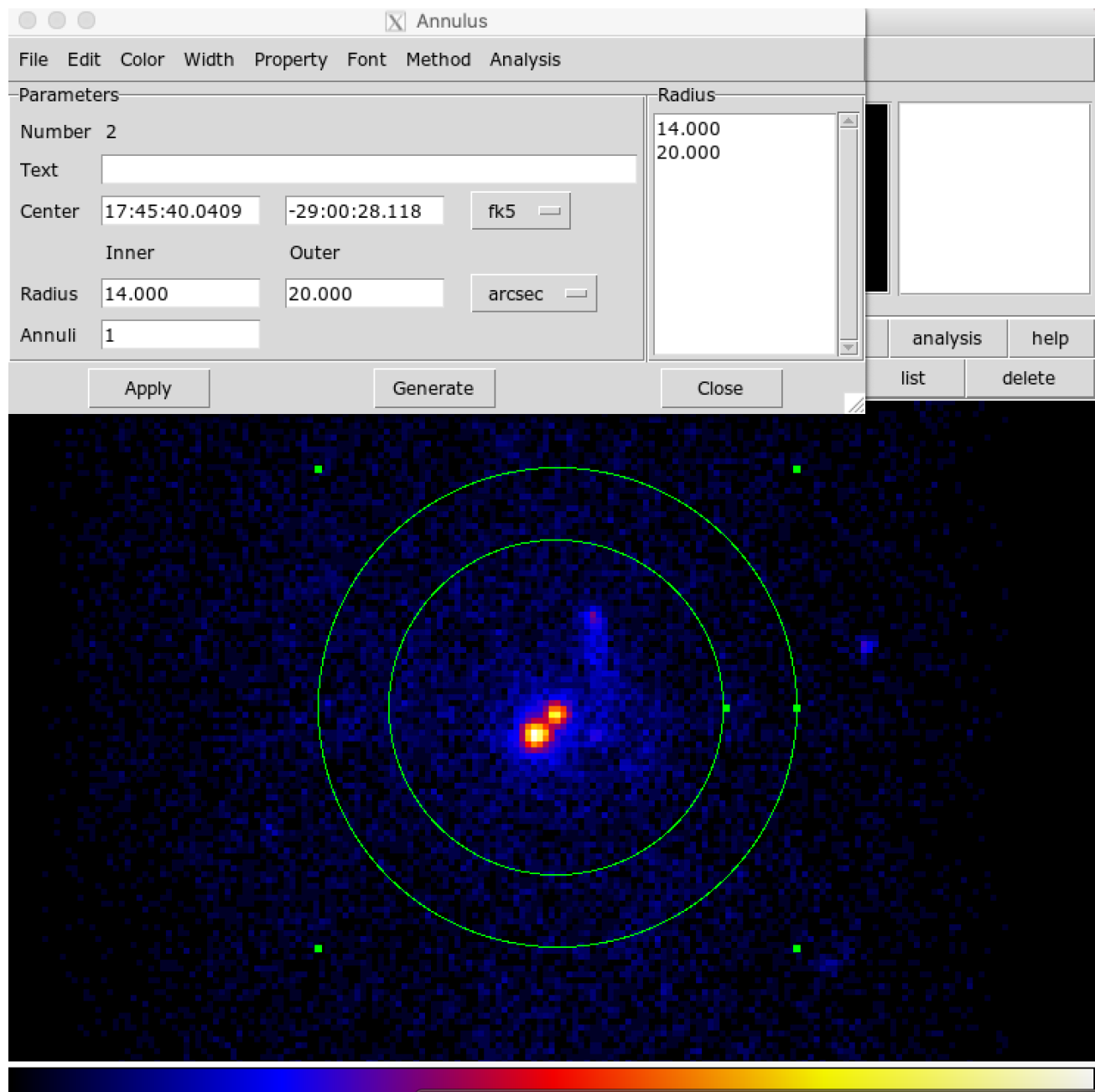
From here, we want to modify the region a little bit, so double click on its middle to see:

In the drop down menu beside the “Radius” field select “arcsec” and give the region a radius of 1.25” for both of the “Radius” boxes. This is because `wavdetect` finds ellipses, but we want a circle. IF AND ONLY IF the circle is *really* offset from Sgr A*, you can move it a little for it to be centered on the brightest pixel. Click “Apply” to save your changes. Make sure to note the final “Center” coordinate somewhere else since we’ll need this for defining our background region. To save your region around Sgr A*, click region → save. Change the file name to “sgra.reg”. After choosing "OK", set the format to "CIAO" and the coordinate system to "Physical".

Next we’ll define a background region. Make sure you delete your Sgr A* region before starting this step. Then in the top menu go to region → shape → annulus.

Left-click and drag on the image in the ds9 display to create an annular region with the same centre as Sgr A*, an inner radius of 14”, and an outer radius of 20”. Click “Generate” then “Apply”.

“Close” the pop-up then region → save as bkg.reg. After choosing "OK", set the format to "CIAO" and the coordinate system to "Physical" like we did for the source region.



6. Determine which chips are being used

From now on and until the end, we return to the UNCROPPED file named something like “file_name_repro_evt2.fits” (e.g. “acisf15043_repro_evt2.fits”). Make sure you are working in the repro directory. We will use [dmstat](#) to determine the correct chip. In the terminal type:

```
punlearn dmstat

dmstat "acisf15043_repro_evt2.fits[sky=region(sgra.reg)][cols ccd_id]"
ccd_id
  min:      7          @:      1
  max:      7          @:      1
  mean:      7
  sigma:     0
  sum:    23807
  good:    3401
  null:      0

dmstat "acisf15043_repro_evt2.fits[sky=region(bkg.reg)][cols ccd_id]"
ccd_id
  min:      7          @:      1
  max:      7          @:      1
  mean:      7
  sigma:     0
  sum:    22176
  good:    3168
  null:      0
```

Make sure to replace the example filename (highlighted in orange) with the name of the file you’re working with. Your output from `dmstat` should look something like the blue text above. Here, the regions I am working with are all located on chip 7. Take note of which regions you’re working with.

7. Create a background-subtracted lightcurve

Make sure you’re in the `repro` directory. Type `ls` to verify that you have two files named “sgra.reg” for the source and “bkg.reg” for the background. We’ll use [dmextract](#) to create a 2-8 keV lightcurve binned in time intervals of 300 s. Check that you’re using the correct filename and `ccd_id` for your ObsID. Change the ObsID in the outfile to the one you’re working with. Be careful to write `pset dmextract` and `infile` on the same line! We have used line breaks to indicate new lines of input in the terminal.

```
punlearn dmextract

pset dmextract
infile="acisf15043_repro_evt2.fits[ccd_id=7,energy=2000:8000,sky=region(sgra.reg)][
```

```

bin time>:::300]"

pset dmextract outfile="15043_sgra_2-8keV_lc300.fits"

pset dmextract bkg="acisf15043_repro_evt2.fits[ccd_id=7,sky=region(bkg.reg)]"

pset dmextract opt="ltc1"

dmextract

```

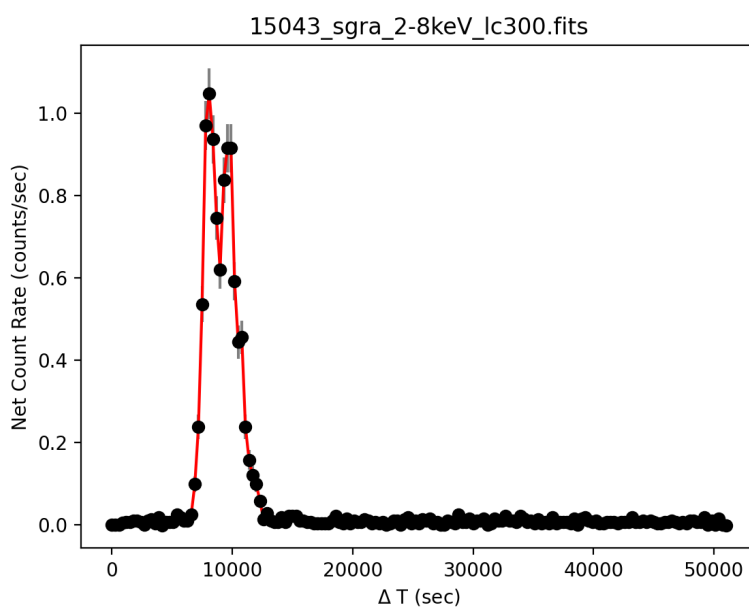
The lightcurve may be plotted with matplotlib. This isn't necessary but it may be useful to check if the lightcurve contains a flare. To exit python from the terminal type `exit()` or `quit()`.

```

python

>>> from pycrates import read_file
>>> import matplotlib.pyplot as plt
>>> tab = read_file("15043_sgra_2-8keV_lc300.fits")
>>> dt = tab.get_column("dt").values
>>> rate = tab.get_column("net_rate").values
>>> erate = tab.get_column("err_rate").values
>>> plt.errorbar(dt, rate, yerr=erate, marker="o", color="red",
mfc="black",mec="black", ecol="grey")
>>> plt.xlabel("$\Delta$ T (sec)")
>>> plt.ylabel("Net Count Rate (counts/sec)")
>>> plt.title("15043_sgra_2-8keV_lc300.fits")
>>> plt.show()

```



8. Create a filtered events file

We'll use [dmcopy](#) to copy a “virtual file” filtered by our source region and our energy band to a physical disk file.

```
punlearn dmcoppy

pset dmcoppy
infile="acisf15043_repro_evt2.fits[EVENTS][sky=region(sgra.reg)][energy=2000:8000]"

pset dmcoppy outfile="15043_sgra_2-8keV_evt.fits"

pset dmcoppy option="all"

dmcoppy
```

9. Save data from McGill server to laptop

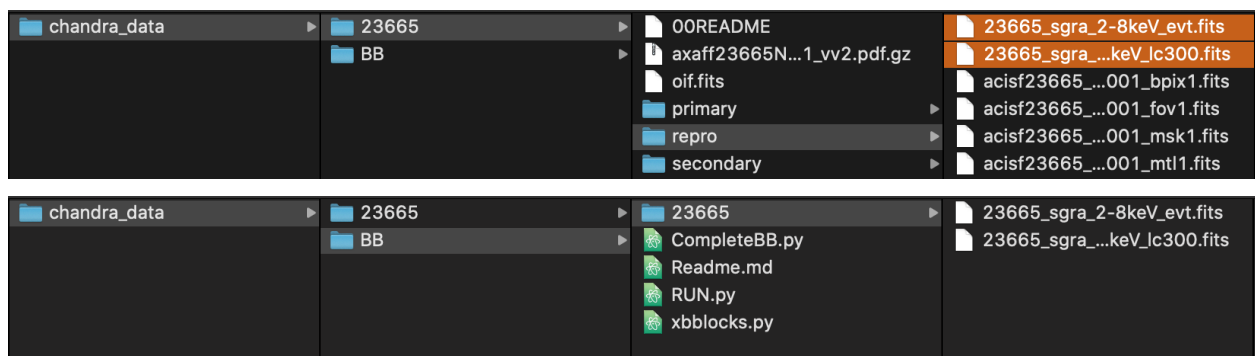
This step is applicable only if you're using the McGill server. Open the terminal but don't log into choco or irulan yet. Type `scp -r server_directory local_directory` (e.g. `scp -r user@physics.mcgill.ca:/home/zark/user/chandra_data ~/chandra_data/15043`).

10. Getting ready to use Bayesian Blocks

Download the CompleteBB.py, xbblocks.py, and RUN.py scripts from this Github in a text format but still written with a py extension:

<https://github.com/mega-sgra/flares/tree/main/BayesianBlockCode>

Move all three scripts into your “chandra_data” folder from Step 4. I like to keep them in a folder named “BB” to keep them separate from the downloaded files. In the “BB” folder create a new folder named by your ObsID. Copy the lightcurve file (e.g. “15043_sgra_2-8keV_lc300.fits”) created in Step 7 and the events file (e.g. “15043_sgra_2-8keV_evt.fits”) created in Step 8 into this folder which can be found in the “repro” folder.



You can do the same thing from the command line:

```
cd ~/chandra_data

mkdir BB

mv ~/downloads/CompleteBB.py ~/downloads/RUN.py ~/downloads/xbblocks.py
~/chandra_data/BB

cd BB

mkdir 15043

cd ..

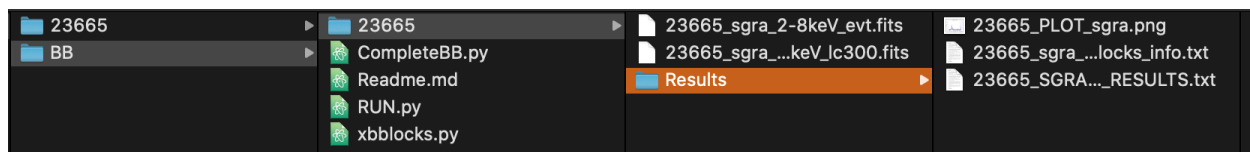
cp 15043/repro/15043_sgra_2-8keV_lc300.fits BB/15043

cp 15043/repro/15043_sgra_2-8keV_evt.fits BB/15043
```

Note that in the third line above I am moving my python scripts from my “downloads” folder to my “BB” folder. Double check that you’re putting in the correct location for where your python files are!

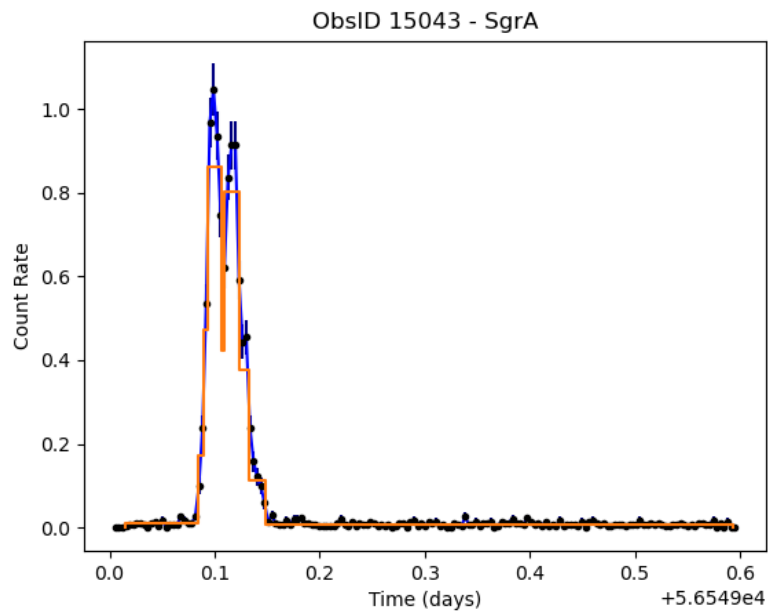
11. Run Bayesian Blocks

In the terminal navigate to the “BB” directory (e.g. `cd ~/chandra_data/BB`) and then type `python RUN.py ObsID False` (e.g. `python RUN.py 15043 False`). If you want to run the Bayesian Blocks code on two sets of data run the command `python RUN.py first_ObsID False second_ObsID False`. This can be done for any number of ObsIDs given that they each have a folder containing the proper lightcurve and events files. The False is for a non magnetar region. For a more in-depth explanation, take a look at the Readme.md file in the previous Github folder <https://github.com/mega-sgra/flares/tree/main/BayesianBlockCode>



After the Bayesian Blocks code runs it will create a "Results" folder in the directory as the lightcurve and events files. In this "Results" directory there will be a plot of the lightcurve plotted with Bayesian blocks, a .txt file with information about each block, and a .txt file containing information for the flare table.

For ObsID 15043 the PLOT.png file should look something like this:



The Results folder should also contain an info.txt file

```

15043_sgra_bayesianBlocks_info.txt
# p0 = 0.05
# timesys = TT
# tstarts = 56549.01465
# tstops = 56549.59415
# n = 3104
56549.01465 56549.08481 61 0.070165 869.379 113.064
56549.08481 56549.08985 75 0.00504331 14871.2 3755.61
56549.08985 56549.09380 161 0.00394328 40829 8420.48
56549.09380 56549.10687 973 0.0130779 74400.3 6253.68
56549.10687 56549.10959 99 0.00271818 36421.4 15391.2
56549.10959 56549.12302 932 0.0134276 69409.5 3517.01
56549.12302 56549.13328 334 0.0102576 32561.2 2653.51
56549.13328 56549.14878 151 0.0155 9741.93 1825.2
56549.14878 56549.59415 318 0.445373 714.008 47.2956

```

and a RESULTS.txt file.

```

15043_SGRA_TABLE_RESULTS.txt
-----
BASIC INFORMATION
-----

Obs_ID: 15043

Obs Date: 2013-09-14T00:04:52

Telescope: CHANDRA

```

```

Instrument: ACIS
Exposure (ks): 50.069314654111864
Quiescent Count Rate (10-3 ct/s): 8.264 +/- 0.463

-----

FLARE NUMBER 1

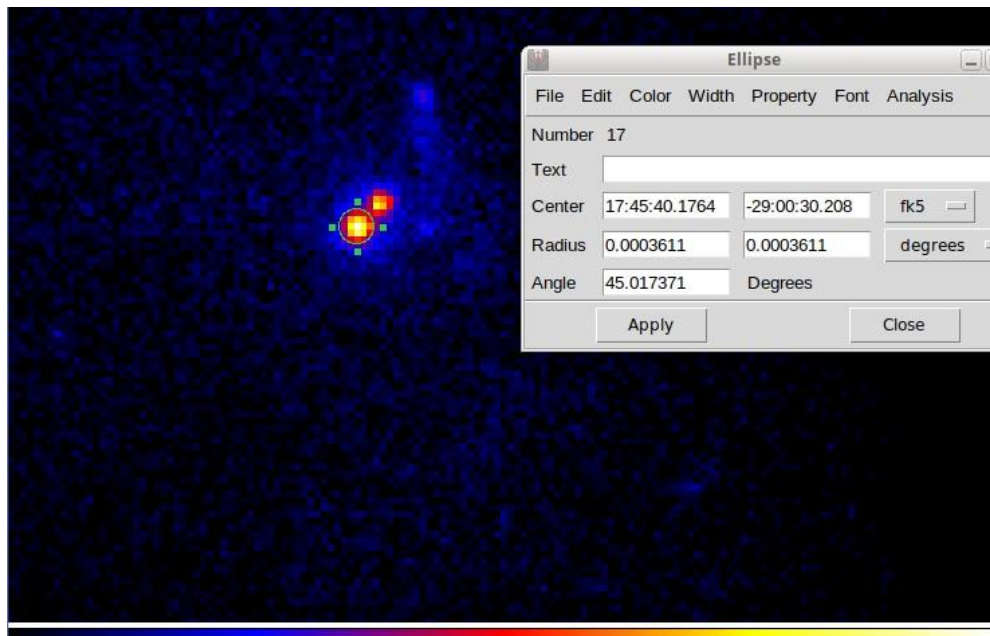
-----

Start Time: 56549.08481 (MJD)
End Time: 56549.14878 (MJD)
Duration: 5526.823968 (s)
Count Rate (mean): 0.49 +/- 0.0094 (ct/s)
Count Rate (max): 1.05 +/- 0.06 (ct/s)
Energy: 206.1 +/- 4.02 1037 ergs
Luminosity: 37.29 +/- 0.73 1034 erg/s
Flux: 49.44 +/- 0.97 10-12 erg/s/cm2
Fluence: 2725.0 ct

```

12. Run the Analysis on the Magnetar

Even though this step is not mandatory in the above flare analysis routine, it can be useful to check whether or not a flare detected from Sgr A* is affected by activity of the nearby magnetar SGR J1745-2900.



To ensure that the flare is not an artifact of the magnetar, you can re-run steps 5 to 11. Keep only the magnetar's region and adjust it so that it is a circle of radius 1.30" (arcsec). Double-check it is the correct region by comparing to the known coordinates of SGR J1745-2900: RA=17:45:40.169, DEC=-29:00:29.84. When you save this magnetar region, use the name "magnetar.reg" and as you follow the steps, just replace "sgra.reg" by "magnetar.reg" and "sgra" by "magnetar". To run the Bayesian Blocks analysis on the magnetar, type `python RUN.py ObsId True` (e.g. `python RUN.py 15043 True`).