Appendix C: Chandra Observations Reduction Pipeline (CORP)

This appendix has been written as a tutorial for the first-time analyzer of *Chandra* data (*e.g.* the "you" role in the text). "When your CIAO-Fu is good, only then will you utilize the stowed backgrounds of the CalDB."

C.1 Copyright

As a formality, I have blanketed CORP with the GNU General Public License. Below is the copyright and license agreement for all scripts in CORP:

C.2 Introduction to CORP

The reduction and analysis of *Chandra* data is given in exquisite detail in the CIAO threads on the CXC's web site[1]. There is very little which is not discussed in the CIAO and HelpDesk threads at the CXC web site. However, to streamline the lengthy reduction and analysis process of extended X-ray sources, such as galaxy clusters and groups, I have written several Perl and IDL scripts which make-up my own *Chandra* Observations Reduction Pipeline (CORP, pronounced "core"). The purpose of this pipeline software is to condense CIAO's tedious prompts and command line intensive steps into an easily executable series of scripts which require minimal interaction and produce science-ready data products. A pipeline also ensures that a large sample of observations are reduced the same way, and a pipeline also eases the pain of analyzing several hundred observations.

There is a critical caveat to the use of CORP and analyzing *Chandra* data in general: **no two *Chandra* observations are the same!** CORP has allowances for many different tool settings and instrument setups, but these options are finite, and no amount of automation can replace human interactivity. It is an absolute ****necessity**** that users of CORP view, scrutinize, and double check the output of every reduction/analysis step. This can be time consuming, but not nearly as time consuming as finding and correcting errors embedded in on-going, or goodness forbid, **published** work.

As of writing this dissertation, all CORP scripts properly interact with CIAO 3.4.1 and CalDB 3.4. The CXC software versions matter because the CXC programmers (whom are great people!) have a tendency to reinvent the wheel every so often. This may result in a change to output filenames, extensions, header keywords, data types, et cetera and can cause a script to err. Errors are not guaranteed however, so it is important to be mindful of how CIAO or the CalDB have changed as

[1] http://cxc.harvard.edu/ciao/threads/index.html

a result of an update (release notes are always provided with an update: read them!). There are some pretty great updates to CIAO included in the final 4.1 version, so I highly recommend CORP users email me to request new scripts when CIAO 4.1 is fully deployed.

Now for a few notes about me, the author and programmer:

1. I use plenty of analysis and X-ray "jargon" in this Appendix. If you come across nomenclature which is unfamiliar, consult the CXC web site, there is absolutely nothing you could want to know about *Chandra* that is not there. CORP is my method for automating most of the logic trees which are in the CXC threads, but this does not mean CORP's operation will always be transparent to a user.

2. I am not a computer programmer. In fact, I only knew pseudo-code when I entered graduate school. Hence, my style of programming is best described as inelegant and brutish. Computer resources are cheap and abundant, so I use lots of inefficient code, but the overhead and time consumption are low, so writing efficient code does nothing to accelerate my work. The scripts of CORP use no command line options besides input, and sometimes output, filenames. Everything the script is being commanded to do is controlled by opening the code and editing the "Options" section near the beginning of the program. Do not worry, it's as simple as editing a Word document.

3. As of now, the scripts are available via a tarball which I will email you. Someday distribution of the code will be handled through a public CVS server. The tarball contains all the scripts, a README file (which is a copy of this appendix), and a folder of example data. Descriptions of what each script does, how it is called, what it takes as input, and what is generated as output are all listed in the header of each program.

4. I am not a debugger. Each script is written to run a very specific set of tasks.

Given the proper input, the scripts return the expected output. However, while I wish the scripts were magic, alas, they are not. If you plan on giving the programs non-standard input and there is some operation you are not sure the script will perform, then find out first by dissecting the code. I may have coded a script to handle your odd data, but I may not. Find out first!

C.3 Initial Reprocessing

C.3.1 Retrieving Data

Getting *Chandra* data from the CDA is not complicated. There are three methods to get data: (1) run the stand-alone *Chaser* program, (2) use the web-based version of *Chaser*, named *WebChaser*[2], or (3) run my script `query_cda.pl`. The first thing to do is determine which ObsIDs need to be downloaded. This is accomplished by searching the CDA via *WebChaser*. The *WebChaser* form is self-explanatory: search via object name, sky coordinates, or using any set of other listed methods and options. After searching the CDA, a list of archived observations will be returned. **VERY IMPORTANT:** Now is the time to make a one column file where each line lists one of the ObsIDs to be downloaded:

```
#Obsid
2419
791
etc.
```

This file is needed to download data and build the reference file used by every script in CORP.

Open the script `query_cda.pl` with an editor (like emacs or xemacs). Within the script are three vital options that need to be set: `$get_nh, $get_z`, and `$get_data`.

[2]http://cda.harvard.edu/chaser/

When set to 'yes', these options tell the script to: (1) find the Galactic absorbing column density ($N_{\rm H}$) using the LAB survey (Kalberla et al., 2005), (2) acquire a redshift from NED[3], and (3) download data from the CDA. The `$get_nh` is trustworthy, so set it to 'yes'. However, the `$get_z` option is not robust. The NED query returns a list of galaxy clusters nearest the aimpoint of the *Chandra* observation, but the proper redshift is not always returned. I typically leave this option set to 'yes' and confirm redshifts by manually querying NED (hey, it's less typing).

The `$get_data` option will cause the script to download data and create a directory for each ObsID and place both into the directory specified by the variable `$dest`. The CDA is large, so a pre-compiled listing of where every ObsID is stored is provided in the file `cdaftp.dat` (this file is part of the CORP tarball). The variable `$ftpdat` needs to point to where this file is stored. The CDA is continually updated, so you will need to update `cdaftp.dat` from time to time. This is accomplished by running:

```
[linux]% perl build_cdaftp.pl <output_filename>
```

Now, run the query script. **WARNING:** If there is an existing `newref.list` in the working directory, it will be overwritten.

```
[linux]% perl query_cda.pl <my_list_of_obsids>
```

The download time is completely dependent on download speed, so if there are many ObsIDs to download, work on something else for a while. The script tells the user what $N_{\rm H}$ and redshift it has found and for what object. The script also informs the user how many of the input ObsIDs were successfully found in the CDA and the total exposure time of observations in the query. You will now have new directories which bear the names of the downloaded ObsIDs (for example, `<hard_drive>/<my_root_datadir>/<obsid>`). There should also be a new file named `newref.list`. Each column within `newref.list` is described below:

[3] http://nedwww.ipac.caltech.edu/

Name: This is the name of the `TARGET` object listed for the observation. This is not necessarily the name you'd like to give the object, so feel free to change it.

ObsID: Obviously this is the ObsID. Most of the file naming convention in CORP involves the ObsID since it is a unique identifier. This may seem clumsy and awkward (especially for the clusters that have multiple ObsIDs) but in the battle of clarity and brevity, clarity wins in my book.

RA: The right ascension of the observation target object. The default output format is decimal degrees, but this can be changed to sexigesimal by changing the `query_cda.pl` variable `$outcrdunit` from `'decimal'` to `'sexigesimal'`.

Dec: The declination of the observation target object in decimal degrees.

Rmax: The maximum observation radius. This is the radius from the cluster center to the nearest detector edge. Rmax needs to be specified by the user for each observation. A script which automatically finds Rmax is forthcoming.

MinCts: The minimum number of counts per temperature annulus. The default is 2500 but this number should be increased for observations with sufficient counts or adjusted depending on scientific goals.

z: The redshift of the target object. Even if the script has automatically queried NED for this value, it is best to double-check. A batch query via NED is simple[4].

Nh20: The galactic absorbing neutral hydrogen column density, $N_{\rm H}$, in $10^{20}{\rm cm}^{-2}$. These values are acquired from the `nh` tool which uses the L.A.B. Survey results (Kalberla et al., 2005).

Tx: The global/virial cluster temperature. There are a number of ways to measure to a cluster temperature, it is best to keep a detailed record of how you made this measurement or where you looked-up the temperature (*e.g.* KEEP CITATIONS you'll want them later). If no temperature can be found in the literature, the script `$find_tx.pl` should be used to determine the cluster temperature. This script iter-

[4]http://nedwww.ipac.caltech.edu/help/batch.html

atively determines temperature in core-excised annuli using a user-specified fraction of the virial radius as the outer radius. When to run the script after removing is specified later.

Fe: The global cluster metallicity. The value listed in the reference file is used only as a starting point for most spectral fits, so the value listed is not all that important. I'd go as far to say this is a deprecated entry. Again, if this value if looked-up in the literature, keep a citation record.

Lbol: The cluster bolometric luminosity. *Chandra* has a small field of view, so if this is a value best taken from the literature, for example from Horner (2001).

Chip: The CCD on which the cluster center is located. The default is to list the array on which the observation is taken (I or S), but specifying which `chip_id` (*i.e.* S3 or I0) is left to the user.

E_obs, Diff, Robs: These are deprecated columns which have been left-in because most scripts were written before they were no longer needed. Forthcoming versions of CORP will not need these columns.

Location: Sometimes data is stored across multiple volumes, hence this column specifies the path to each ObsID. For example if one ObsID is stored on `/mnt/HD1` and another is on `/media/USB1`. This allows a single instance of a script to be run on distributed data.

Great! The data is now out of the archive and into your hands. What are all these different files? The answer to that question is lengthy and of fundamental importance in honing your CIAO-Fu. Read the documentation on data products[5,6].

Included in the CORP tarball is a script named `ds9_viewreg.pl`. The script is very useful for viewing observations quickly and has been invaluable throughout the years. The program performs a multitude of tasks and relies on the functionality of

[5]http://cxc.harvard.edu/ciao/data/basics.html
[6]http://cxc.harvard.edu/ciao/threads/intro_data/

DS9[7]. The script header completely explains all the variables and how to use the program. Before starting the bulk of data reduction I recommend using `ds9_viewreg.pl` to determine the `Chip` and `Rmax` values for each ObsID and then entering them into the `newref.list` file.

C.3.2 Create New Level-2 Events File

Before completing any true analysis, such as finding the cluster center or identifying point sources, the CIAO threads recommend creating a new events file. Removing bad grades, time intervals with flares, bad pixels, et cetera ensures that analysis further downstream is more robust. The initial reprocessing step described here requires running the script `reprocess.pl` with some, but not all, of the internal switches set to "yes". Descriptions for the internal switches of this program are in the code header. The program `reprocess.pl` performs multiple tasks and will be used more than once: In the first pass, `reprocess.pl` will perform the tasks outlined below, later it will be used to exclude point sources. For more detail on any of the steps below, read the CIAO documentation[8,9].

First, edit the script so all the options are "yes" and only `$exclude` is equal to "no". To run the script, simply call it with Perl giving the reference file as input (if you do not want an ObsID analyzed, simply comment it out of the reference file by placing a "#" at the front of the line):

```
[linux]% perl reprocess.pl reference.list
```

There is no specific version of Perl required to run CORP. For each ObsID in the reference file, a new `reprocessed` subdirectory has been created and all new files have been placed in that directory. The output files are listed at the end of the script header. The reduction steps performed by `reprocess.pl` are: remove the ACIS

[7] http://hea-www.harvard.edu/RD/ds9/
[8] http://cxc.harvard.edu/ciao/guides/acis_data.html
[9] http://cxc.harvard.edu/ciao/threads/createL2/

afterglow, create a new bad pixel file, set `ardlib.par`, update time-dependent gain (TGAIN), apply charge transfer inefficiency (CTI) correction (if appropriate), filter on event grade, filter on good time intervals (GTIs), destreak, remove background flares and/or periods of excessively high background, make blank-sky background file, and make off-axis blank-sky background file.

Cleaning for flares is a detailed step and is best understood by reading the CIAO documentation. The main steps in extracting and filtering a light curve are removing bright sources, setting the time bin size specific to front or back illuminated CCDs, setting the energy window for the specific CCD type, analyzing the light curve using the contributed routine `lc_clean.sl`, and filtering out the time periods from the GTIs which contain high background. All of these steps are handled automatically by the script, however `lc_clean.sl` **DOES NOT** always find the proper background count rate mean. This results in the beginning or end of flare not being excluded from the GTIs, while perfectly good intervals are excluded (see Figure C.1 as an example). The solution to this problem is very simple.

After reprocessing you should examine each light curve anyway, so finding these cases will be easy. Run

```
[linux]% perl view_prof.pl reference.list
```

with the internal switches set to view lightcurves. If the lightcurve cleaning failed then there will be no lightcurve to view, in which case run the following script for only those clusters which experienced a failure (logged in `errors.log`) by commenting out all the clusters in `reference.list` (placing a `#` at the beginning of the line in `reference.list`) which did not fail, then running:

```
[linux]% idl
IDL> load_ltcrv, 'reference.list'
```

When you find a case where a flare has been improperly removed, estimate the mean background count rate by finding the peak in the count rate distribution in

the bottom pane of the figure. Now we'll create a new ASCII file which contains information about this OBSID and it's flare. The file should have the format:

```
#Obsid  Mean Rate
2419    1.300
791     0.175
```

Save this file, set the `$flarefile` keyword in `reprocess.pl` to point to this new file, and now re-run `reprocess.pl` with all other options set to "no" except the `$clean_events` option which should be "yes". Examine the new lightcurve and repeat this process if the mean is not exactly where you think it should be.

Making blank-sky backgrounds is easily the most involved step in `reprocess.pl` and to fully understand what is being done, reading Maxim Markevitch's Cookbook[10] in conjunction with the background thread[11] are a must.

After `reprocess.pl` has finished running, it is wise to spend the time examining all of the output files. This entails steps such as viewing the `evt1, evt2, bgevt,` and `clean` files; checking the light curves for missed or improperly excluded flares; verifying the background file have the proper exposure times in the headers. If the data is trustworthy, then move along.

C.3.3 Remove Point Sources and Identify Cluster Center

Determining the cluster center and identifying points sources for exclusion is a crucial step in extended source analysis as these steps significantly affect the final results. Reliably determine the cluster center and finding point sources first requires an exposure map. An exposure map is a replication of the optical system's characteristics (*e.g.* CCD quantum efficiency, CCD non-uniformities, vignetting, bad pixels, etc.) dithered and exposed exactly as the observation. The exposure map is used to re-

[10] http://cxc.harvard.edu/contrib/maxim/acisbg/
[11] http://cxc.harvard.edu/ciao/threads/acisbackground/

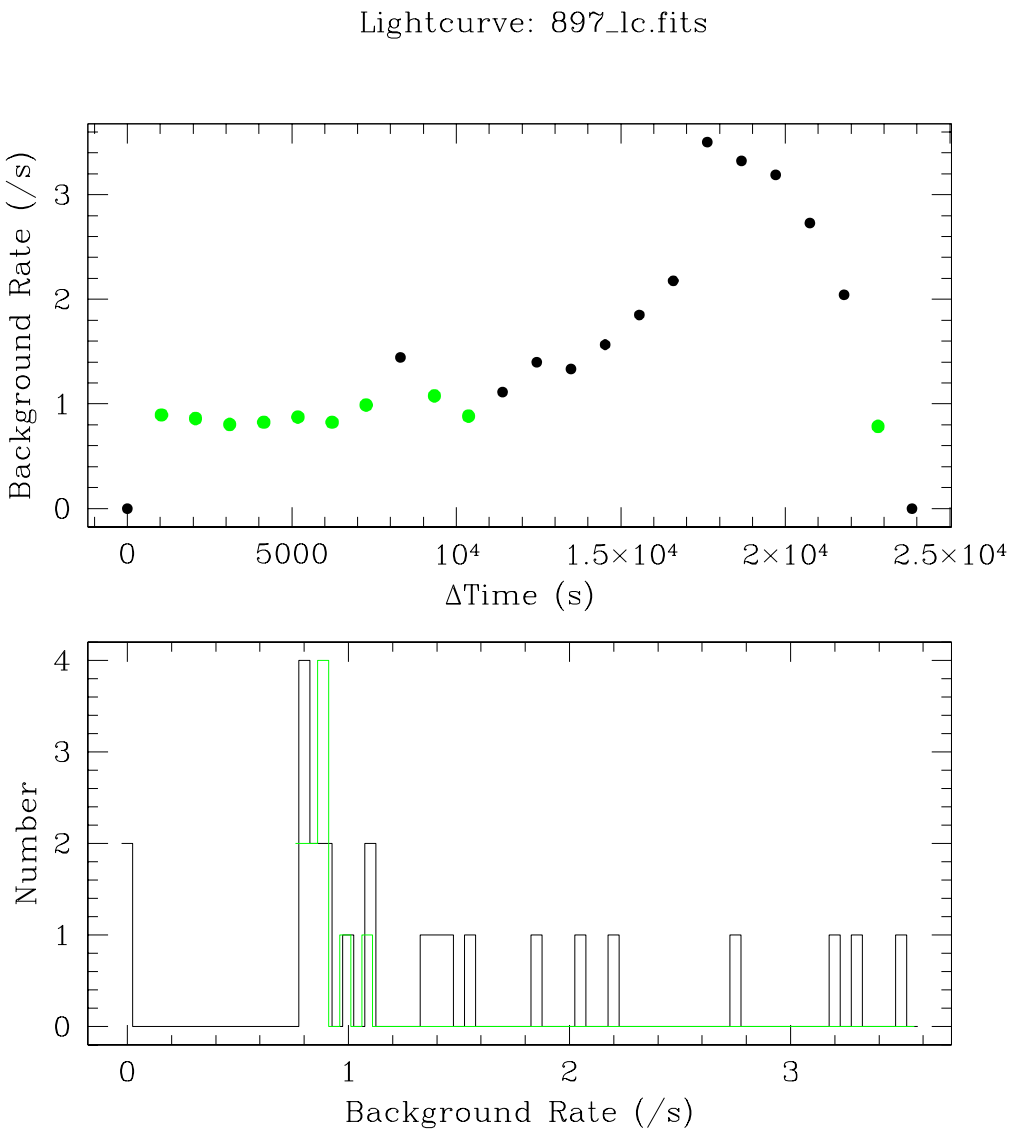

Figure C.1 Shown here is an example light curve output by `lc_clean.sl` which is called when `reprocess.pl` is run. *Top panel:* Plot of background count rate versus time in the observation. Green points mark those time intervals within ±20% of the mean rate. Zero rate bins at the beginning and end of an observation are intentional dead times. The flares can easily be identified in this example. Also note that the automated mean detection did not remove the wings at ∼ 5 ksec and ∼ 9 ksec of the weaker flare. *Bottom panel:* Histogram of the light curve shown in the top panel. The green line is again for the intervals within ±20% of the mean rate.

move instrumental features, like chip gaps, which will be erroneously detected as point sources or skew the calculation of the cluster center.

There are two ways of calculating an exposure map: using a monoenergetic incident spectrum or a more specific spectral model. The script for making exposure maps can do both, and it is up to the user to determine if one or the other is preferred for their analysis. For all of the clusters I have analyzed, the monoenergetic assumption does not give significantly different results from the more elegant spectral model method.

There are many options for making an exposure map and they are explained in the script header. To make an exposure map, run the script `exp_map.pl`:

```
[linux]% perl exp_map.pl reference.list
```

The output will be an instrument map, aspect histogram, exposure map, and a normalized image of the cluster (in flux units). The normalized image is what will be used to find point sources and find the cluster center. By default, the script makes a full resolution (binning=1) exposure map. This has the drawback of being time consuming, but the advantage of more accuracy in spatial analysis. If you are simply experimenting data, then increase the binning factor to four or eight. I do not recommend using such highly binned data for analysis unless the cluster center is especially obvious (*i.e.* in highly peaked clusters) or you plan on remaking point source exclusion regions by hand.

Now that you have an exposure map, edit the options in the script `cent_emi.pl` and run the script.

```
[linux]% perl cent_emi.pl reference.list
```

or

```
[linux]% perl dub_centroid.pl dub_reference.list
```

This program will find the cluster centroid and peak using the CIAO tool `dmstat`. If these two quantities differ by less than 70 kpc, the peak will be returned as the

cluster center. The script also outputs a new reference file with the cluster center in the RA and Dec columns. Now is the moment to be a researcher: view the clusters with the center marked, does this solution look right?

After determining if the cluster center finder worked properly, the next step is to identify and exclude point sources. The script which does this is `find_pt_src.pl` which calls the CIAO tool `wavdetect`. The primary script output is an ASCII file listing the exclusion regions.

```
[linux]% perl find_pt_src.pl reference.list
```

It is very important at this stage to view the observation with the exclusion regions overlaid. The `wavdetect` algorithm is very sophisticated, however it will miss sources (*e.g.* sources very close together), detect spurious sources (*e.g.* chip gaps and edges), detect sources which are bright, diffuse cluster emission (*e.g.* the core of a bright, peaky cluster), and miss sources in regions of high background (*e.g.* point sources in or near bright cluster core).

While viewing the observation and regions with `ds9_viewreg.pl`, it is straight forward to delete, add, and alter regions. After doing so, go to the 'Regions' menu, 'File Format' tab, and click 'Ciao'. Then under the 'Regions' menu click 'Save regions...' and simply overwrite the loaded `<obsid>_exclude.reg` region file. That is all it takes to edit the exclusion regions in a quick, by-eye batch session. Now edit the script `reprocess.pl` so that all options are "no" except for the `$exclude` option which should be "yes". Running `reprocess.pl` will now remove all the regions you just saw/specified in DS9.

```
[linux]% perl reprocess.pl reference.list
```

The output from this final step is the file `<obsid>_exclude.fits` which is the crown jewel of CORP: an up-to-date, flare-clean, point source-clean, events file at level-2. As usual, you should view this file and make sure the point source exclusion functioned as expected. The initial reprocessing steps are now complete, congrats.

C.4 Intermediate Analysis

If at this point you still do not have a temperature for some number of clusters, run the following script to find one:

```
[linux]% perl find_tx.pl reference.list
```

This script can also be used to determine redshifts and metallicities. Read the script header for more detailed instructions on use.

The intermediate analysis steps involve extracting radial profiles and spectra from the observations. These radial profiles form the basis for the final analysis steps of the next section. One script, `make_profile.pl`, extracts both a cumulative counts profile and a surface brightness profile. In the options section of the script you specify the size of the annular bins used to extract the profiles. For the cumulative counts profile I recommend 2 pixel width bins, and either 5 pixel or 10 pixel width bins for the surface brightness profiles. For the surface brightness profiles you also need to specify the energy range for the extraction. There are many other options for this script which are detailed in the program header.

```
[linux]% perl make_profile.pl reference.list
```

or

```
[linux]% perl make_multiprof.pl dub_reference.list
```

Depending on the number of counts in the observation, this step can take a few minutes or a few hours. The script also outputs plots of the two profiles which should be viewed to ensure everything ran correctly.

Now run the script `exp_corr.pl` which extracts a radial profile from the exposure map and will be used for exposure correcting radial profiles later on. As usual, set the options in the header, specifically the bin size of the profile to extract. The bin size needs to match that of the surface brightness profile just extracted. It is possible to extract multiple exposure profiles since the bin size is amended to the output file

name. This step is fast, so extracting profile for bin sizes of 5, 10, or 20 pixels does not take long.

```
[linux]% perl exp_corr.pl reference.list
```

or

```
[linux]% perl multiexp_corr.pl dub_reference.list
```

With the cumulative profile, it is now possible to create annuli for the temperature profile. The script `make_annuli_reg.pro` is used to make the annuli. The cumulative profile is divided up into annular bins containing a minimum number of counts and then these bins are output as region files later used to extract spectra. The entries in the 'Mincts' column of the reference file are used to set the minimum number of counts. I typically run the script with all options set to "no" (meaning only mock regions are produced) and view the output plot to ensure the number and spacing of the bins is appropriate for the cluster in question. What is appropriate? Well, too many closely spaced annuli for a symmetric peaked cluster is redundant, and too few widely spaced annuli for a complex cluster is insufficient, unless of course there are not enough counts to produce more bins. After ensuring the number of annuli produced is agreeable, run the script with the options set to "yes". **WARNING:** by default, the script deletes all annuli and associated spectral files. If you run this script after having extracted spectra, be sure to set `mkbackup` to "yes" AND provide the path to a valid, existing back-up directory, `bkdir`, this will save those existing spectra.

```
[linux]% idl
IDL> make_annuli_reg, 'reference.list'
```

or

```
IDL> make_multiannuli_reg, 'reference.list'
```

A whole ensemble of individual region files with the specified name will be output by this script. With these region files extracting spectra is straightforward, simply

run the script `extract_spectra.pl`. This script has a number of important options which are detailed in the script header.

```
[linux]% perl extract_spectra.pl reference.list
```

If there have been no errors, then you should have radial profiles and spectra for each ObsID in the reference file, congratulations.

C.5 Final Analysis

The final steps in the analysis process are normalizing the background spectra for differences between the blank-sky background and observation background hard-particle count rates, extracting and fitting residual spectra for the local soft background, fitting the cluster spectra, and running the master IDL routine that produces entropy profiles. An explanation of why and how background adjustments are made is presented in Chapter 2.

C.5.1 Spectral Adjustments and Fitting

The hard-particle background is changing as a function of time. Thus, the strength of this background component for the epoch in which the blank-sky backgrounds were taken will be different from when the observations were taken. The first step in accounting for this difference involves running the script `bgd_ratio.pl` which will output the ratio of observation to blank-sky 9.5-12.0 keV count rates.

```
[linux]% perl bgd_ratio.pl reference.list
```

The script outputs a file containing these ratios. Keep this file someplace which is easily accessible as a later script will be querying this list to normalize the spectra.

To account for the spatially varying Galactic soft-component you must extract soft-residuals. Soft-residuals are the leftovers of subtracting a blank-sky spectrum

from an observation spectrum both extracted from the same part of the sky and far from the cluster emission. The first step is to clean-up the off-axis observation chips of all point and diffuse sources. This step can be performed blind because if too many sources are removed it does not matter.

```
[linux]% perl addbg_rm_pt_src.pl reference.list
```

The soft-residual spectra are output from this script.

Prior to fitting any spectra all background spectra need to be normalized. This is done quickly by specifying the names of the spectra to be normalized in the script `adj_backscal.pl` and then running it.

```
[linux]% perl adj_backscal.pl reference.list
```

Normalization is applied by adjusting the header keyword `BACKSCAL` in the spectrum. The `BACKSCAL` keyword is related to the final background subtracted spectrum of an observation by

$$SPEC_{ctr} = \frac{SRC_{cts}}{SRC_{exp}} - \frac{BGD_{cts}}{BGD_{exp}} \cdot \frac{1}{BGD_{scal} \cdot SRC_{scal}} \tag{C.1}$$

where the abbreviations 'SRC' $\rightarrow$ source, 'BGD' $\rightarrow$ background, 'ctr' $\rightarrow$ count rate, 'cts' $\rightarrow$ counts, 'exp' $\rightarrow$ exposure time, and 'scal' $\rightarrow$ `BACKSCAL`. The `BACKSCAL` keyword is defined as the detector area over which the spectrum was extracted, divided by the total detector area. Adjustment of the blank-sky background `BACKSCAL` value follows directly from the above equation by multiplying the existing value of `BACKSCAL` by a correction factor, η, which is related to the ratio of the count rate in the 9.5-12.0 keV range of the observation to the blank-sky background by

$$\eta = (\frac{OBS_{ctr}}{BGD_{ctr}})^{-1}. \tag{C.2}$$

Each background spectrum is copied into a new file before this correction is applied so that reversal at a later date is possible.

Now that the spectra are all adjusted, the fitting can begin. There are a large number of options in spectral fitting and these are detailed in the individual script's header. In addition, interpreting the results of the fitting requires more discussion than is useful here. The order in which fitting is done is important since the master spectral fitting routines need output from the fitting of the soft-residuals.

```
[linux]% perl fit_sofex.pl reference.list <spectral model>
[linux]% perl fit_projected.pl reference.list <spectral model>
or
[linux]% perl fit_simulta.pl dub_reference.list <spectral model>
```

The output of these scripts are data tables with the spectral fits for each annulus associated with each ObsID. With radial profiles in-hand and spectral analysis complete, it is now possible to calculate many more physical properties of a cluster.

C.5.2 Generating Entropy Profiles

So here it is, the end of a long journey. Your CIAO-Fu is good, congratulations. The master program which performs the deprojection, derives electron gas density, pressure, entropy, mass (not to be trusted), and cooling time is `kfitter.pro`. This program has a handler program `run_kfit.pro` which makes batch analysis simpler. These programs have their own `README` files which have not been duplicated here. After running `kfitter.pro` you will have a suite of data tables and plots for each cluster which are publication ready.

C.5.3 Additional Code

I have coded many other tools which are extremely useful in the analysis of galaxy clusters. If you would like to use any of the following tools, simply email me or check

the public CVS repository.

1. Batch query NVSS, SUMSS, or VLA First for radio sources within a region of interest.

2. Calculate the X-ray and Sunyaev-Zel'dovich signatures for a mock cluster based on user input parameters.

3. Generate an image and calculate properties of the Chandra PSF for any location or energy on the ACIS-S or ACIS-I arrays.

4. Generate 2D maps of cluster properties (temperature, density, entropy, abundance, pressure, hardness ratio, *etc.*) and return nice profiles and LaTeX tables for each.

5. Create MySql and/or FITS databases from ASCII tables of cluster data.

6. Calculate gas mass and gravitating mass profiles for a cluster with known temperature and density.

7. Many other tools for simulating data, extracting spectra, making web pages, and on and on.

"Pasta be upon you."

-FSM