The XMM-Newton ABC Guide:

An Introduction to XMM-Newton Data Analysis

NASA/GSFC XMM-Newton Guest Observer Facility

Steve Snowden, Lynne Valencic, Brendan Perry, Michael Arida, K.D. Kuntz

With contributions by: Ilana Harrus, Stefan Immler, Rick Shafer, Randall Smith, Martin Still

Version 4.7

for XMM-SAS v16.0

February 2017

Copies of this guide are available in `html`, `postscript` and `pdf` formats.

Contents

List of Tables

Table 1: List of Acronyms

ARF	Ancillary Region File
CAL	Calibration Access Layer
CCD	Charge Coupled Device
CCF	Current Calibration File
CIF	Calibration Index File
EPIC	European Photon Imaging Camera
FITS	Flexible Image Transport System
GO	Guest Observer
GOF	NASA/GSFC Guest Observer Facility
GSFC	Goddard Space Flight Center
GUI	Graphical User Interface
HEASARC	High Energy Astrophysics Science Archive Research Center
HTML	Hyper Text Markup Language
OAL	ODF Access Layer
ODF	Observation Data File
OM	Optical Monitor
PDF	Portable Data Format
PP	Pipeline Processing System
PPS	Pipeline Processing
PV	Performance Validation
RGS	Reflection Grating Spectrometer
RMF	Redistribution Matrix File
SAS	Science Analysis System
SOC	Science Operations Center
SSC	Survey Science Centre
SV	Science Validation
XMM	X-ray Multi-Mirror Mission

Chapter 1

Introduction

The purpose of this *ABC Guide* to *XMM-Newton* data analysis is to provide a simple walk-through of basic data extraction and analysis tasks. Also included is a guide to references and help available to aid in the analysis of the data. We have tried to balance providing enough information to give the user a useful introduction to a variety of analysis tasks with not providing too much information, which would make a guide like this too ponderous to use. As such, there is no intention to replace the SAS Handbook, which should be considered the highest authority for the use of SAS. Therefore this document will not display the full versatility of the SAS tasks, and of SAS itself, but it will hopefully show a path through the forest.

Chapter 2 provides lists of web-based references for the *XMM-Newton* project, help desks, analysis guides, and science and calibration data. Chapter 3 provides a description of the data files provided for observation data sets. Chapter 4 discusses the installation and use of SAS. The rest of the chapters discuss the analysis of EPIC, RGS, and OM data respectively, taking the detector mode and SAS interface (command line or GUI) into account. We also include brief introductions to analysis of the reduced data with XSPEC and Xronos.

This document will continue to evolve. Updated versions will be made available on our web site at:
`http://heasarc.gsfc.nasa.gov/docs/xmm/abc/`

1.1 ACKNOWLEDGMENTS

This guide would not have been possible without the help and comments from all people involved in the *XMM-Newton* project. In particular, we would like to thank Giuseppe Vacanti and Julian Osborne whose comments made this a more complete and accurate document.

IMH wishes to thank all the OM calibration team and in particular Antonio Talavera, Matteo Guainazzi and Bing Chen for their help in the preparation of this and other documents related to the OM.

SLS wishes to thank Dave Lumb, Richard Saxton, and Steve Sembay for their helpful insights into EPIC data analysis.

Chapter 2

Useful Information and References

2.1 The Main Websites

- *XMM-Newton* SOC, fount of all *XMM-Newton* project information:

  `https://www.cosmos.esa.int/web/xmm-newton`

- NASA/GSFC GOF, source of US specific information and a mirror site for software and public data access:

  `http://heasarc.gsfc.nasa.gov/docs/xmm/xmmgof.html`

- Survey Science Centre

  `http://xmmssc-www.star.le.ac.uk/`

2.2 *XMM-Newton* Help Desks

- The main project helpdesk is located at Vilspa and can be accessed through the SOC's website. The helpdesk also provides an archive of previously asked questions.

  `https://www.cosmos.esa.int/web/xmm-newton/xmm-newton-helpdesk`

- The NASA/GSFC GOF offers an e-mail helpdesk for both general support and for US-specific issues:

  `xmmhelp@lists.nasa.gov`

  Some questions addressed to the NASA/GSFC GOF may be redirected to the Vilspa helpdesk.

2.3 Mission Planning and Spacecraft Status

- Observation Log:

  `http://xmm2.esac.esa.int/cgi-bin/obs_search/selectobs_cosmos`

  The scheduling information from this data base has been extracted and incorporated into a Browse data base at GSFC:

  `http://heasarc.gsfc.nasa.gov/db-perl/W3Browse/w3browse.pl`

- Short-Term Timeline:

  `https://www.cosmos.esa.int/web/xmm-newton/short-term-schedule`

2.4 Public Data Archives

- SOC Public Data Archive via the XSA:

  `https://www.cosmos.esa.int/web/xmm-newton/xsa`

- GSFC Archive Mirror Site via Browse:

  `http://heasarc.gsfc.nasa.gov/db-perl/W3Browse/w3browse.pl`

2.5 Calibration Data

- *XMM-Newton* Calibration Page. On this page can be found links to the Current Calibration File (CCF) archive, release notes for CCF updates, response and background files for all instruments, and the current calibration status of all instruments.

  `https://www.cosmos.esa.int/web/xmm-newton/calibration`

- Caldb, NASA/GSFC GOF mirror site for canned response files:

  `ftp://legacy.gsfc.nasa.gov/caldb/data/xmm/`

2.6 Software

- *XMM-Newton* Standard Analysis System (SAS):

  `https://www.cosmos.esa.int/web/xmm-newton/sas`

- *XMM-Newton* Extended Source Analysis Software (ESAS):

  `http://http://heasarc.gsfc.nasa.gov/docs/xmm/esas/cookbook/xmm-esas.html`

- HEASARC HEASoft Package:

  `http://heasarc.gsfc.nasa.gov/docs/corp/software.html`

- CXC CIAO Package:

  `http://asc.harvard.edu/ciao/`

2.7 Analysis, Documentation, and Helpful Hints

- On-Line Users Guide to SAS:

  `https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/sas_usg/USG/`

- There is a "watchout" page for current SAS bugs at:

  `https://www.cosmos.esa.int/web/xmm-newton/sas-watchout`

- SAS and ESAS Threads:

  `https://www.cosmos.esa.int/web/xmm-newton/sas-threads`

- Very detailed descriptions of all SAS tasks, including parameter lists, algorithms, warnings, and errors:

  `https://xmm-tools.cosmos.esa.int/external/sas/current/doc/packagelist.html`

- *XMM-Newton* Users Handbook:

  `https://xmm-tools.cosmos.esa.int/external/xmm_user_support/documentation/uhb/index.html`

- This Guide:

  `http://heasarc.gsfc.nasa.gov/docs/xmm/abc/`

Chapter 3

Data

3.1 Useful Documentation

There are a number of documents which the users of *XMM-Newton* data should be aware of. These documents include the *Data Files Handbook* and *Pipeline Products Description*. The most recent versions of these documents can be found in the SOC Document section under
`https://www.cosmos.esa.int/web/xmm-newton/documentation`.

For observation data sets going to US PIs, the GOF makes the data available online after PGP encryption and after converting the file names to upper case. When the proprietary period for the observation expires the data are decrypted leaving the file names unchanged. A simple decryption script, minus the relevant keys of course, can be found at:
`ftp://legacy.gsfc.nasa.gov/xmm/software/decrypt.pl`.
NOTE: Laura Brenneman wrote a script and accompanying help file that gives explicit directions on how to most quickly pull over all the files in a data set from the archive, as well as decrypting, and uncompressing the files in preparation for data analysis. This package can be found at:
`ftp://legacy.gsfc.nasa.gov/xmm/software/prepare_xmm_data.tar.gz`. and contains the following files: `README`, `decrpyt.pl`, and `prepare_xmm_data.pl`.

3.2 The Data

One of the first steps that should be taken when examining your data is to check to see what you actually have. *XMM-Newton* observations can be broken into several exposures which are each assigned separate observation numbers. These separate exposures can be radically different in length and can also have the different instruments in different modes. For example, in one case the full observation was 60 ks with EPIC and RGS active but there was one delivered exposure which was $\sim$ 3 ks and had only RGS active. (This can happen because the RGS can operate farther into regions of higher radiation than the EPIC detector. The additional observation time can be considered an additional exposure with only the RGS active.) The instrument summary pages, with nomenclature `P*SUMMAR0000.HTM`, are a good place to start. Second, to quickly access images from the various instruments, examine the PPSGRA Pipeline Products page, viewable with a web browser. These files have the nomenclature P*PPSGRA000_0.HTM. (see § 3.3.3).

3.3 PI Data

Proprietary *XMM-Newton* data is available for download via your XSA account. Email instructions from the XMM-SOC at Vilspa are sent to the address on record with detailed directions on how to retrieve your data via the XSA.

The data files can be considered to come in two groups in separate subdirectories when retrieved, the Observation Data Files (ODF) files and Pipeline Processing (PPS) files. The ODF data contain all of the observation-specific data necessary for reprocessing the observation. The PPS data contain, among other things, calibrated photon event files and source lists.

For observation data sets going to US PIs, the GSFC GOF makes the data available in two directories containing the following groups of files.

- ODF – The ODF (raw) data files
- PPS – The pipeline processed data products

3.3.1 ODF Data

ODF data come with file names in the following format:

- `mmmm_iiiiiijjkk_aabeeeccfff.zzz`

  `mmmm` – revolution orbit number

  `iiiiii` – proposal number

  `jj` – target ID number in proposal

  `kk` – exposure number for target

  (**NOTE:** The ten-digit combination of `iiiiiijjkk` is the **observation number** and is used repetitively throughout the file nomenclature.)

  `aa` – detector (`M1` – MOS1, `M2` – MOS2, `PN` – PN, `OM` – OM, `R1` – RGS1, `R2` – RGS2, `SC` – spacecraft)

  `b` – flag for scheduled (`S`) or unscheduled (`U`) observations, or general purpose `X` files

  `eee` – exposure number

  `cc` – CCD number or OM window number

  `fff` – data identifier for the three detectors or spacecraft itself; see Table 3.1.

  `zzz` – format (`FIT` - FITS, `ASC` - ASCII)

NOTE: For SAS processing, the file names should contain all upper case characters. However, at least with early CDs, the file names used lower case characters. The GSFC *XMM-Newton* GOF provides a script to rename the files.

3.3.2 Pipeline Product Data – Summary Files and Groupings

The pipeline processing produces quite a number of useful products which allow a first look at the data, but can overwhelm the user by their sheer numbers. The first place to look is the summary files, which have names with the following nomenclature:

- `PPiiiiiijjkkAAX000SUMMAR0000.HTM`

  `iiiiiijjkk` – observation number

  `AA` – detector ID (`EP` - EPIC, `OM` - Optical Monitor, `RG` - RGS, `OB` - Observation)

PP data contain some immediately useful data products such as calibrated photon event lists, source lists, and images. While there are a large number of products which come in a single directory, they can be associated in up to 15 groupings; see Table 3.2. (The number of groups can vary depending on the number of operational instruments, e.g., if the OM is turned off there are no OM products.) Further information on each of these groupings and associated files, such as file contents, file types, and how they may be viewed, can be found in Table 3.3. Each group has an associated HTML file which organizes access to the files and provides a limited description of them. The names of the HTML files are of the following form:

- `PPiiiiiijjkkAAAAAA000_0.HTM`

  `iiiiiijjkk` – observation number

  `AAAAAA` – group identifier (see Table 3.2)

3.3.3 Pipeline Product Data – Data Files

The data file names are of the form (see Table 41 in the *XMM Data Files Handbook*):

- `Piiiiiijjkkaabll lCCCCCCnmmm.zzz`

  `iiiiiijjkk` – observation number

  `aa` – detector, `M1` - MOS1, `M2` - MOS2, `PN` - PN, `CA` - for files from the CRSCOR group, `R1` - RGS1, `R2` - RGS2, `OM` - OM.

  `b` – flag for scheduled (`S`) or unscheduled (`U`) observations, or `X` for files from the CRSCOR group (and any product that is not due to a single exposure)

  `lll` – exposure number

  `CCCCCC` – file identification for data from each detector; see Table 3.3

  `n` – For EPIC data, this is the exposure map band number; for RGS data, this is the spectral order number; for the OM, this is the OM window within the exposure.

  `mmm` – source number in hexidecimal

  `zzz` – file type (e.g., PDF, PNG, FTZ, HTM)

  - `ASC` - ASCII file, use a web browser, or the "more" command
  - `ASZ` - gzipped ASCII file
  - `FTZ` - gzipped FITS format, use *ds9, Ximage, Xselect, fv*
  - `HTM` - HTML file, use Firefox or other web browser
  - `PDF` - Portable Data Format, use *Acrobat Reader*
  - `PNG` - Portable Networks Graphics file, use a web browser
  - `TAR` - TAR file

Table 3.1: ODF data file identifiers.

Data ID	Contents
	EPIC files
IME	Event list for individual CCDs, imaging mode
RIE	Event list for individual CCDs, reduced imaging mode
CTE	Event list for individual CCDs, compressed timing mode
TIE	Event list for individual CCDs, timing mode
BUE	Event list for individual CCDs, burst mode
AUX	Auxiliary file
CCX	Counting cycle report (auxiliary file)
HBH	HBR buffer size, non-periodic housekeeping
HCH	HBR configuration, non-periodic housekeeping
HTH	HBR threshold values, non-periodic housekeeping
PEH	Periodic housekeeping
PTH	Bright pixel table, non-periodic housekeeping
DLI	Discarded lines data
PAH	Additional periodic housekeeping
PMH	Main periodic housekeeping
	RGS files
AUX	on-board processing statistics
SPE	raw event list for one CCD
DII	diagnostic images
D1H	CCD readout settings
D2H	CCD readout settings
PFH	housekeeping data
ODX	pixel offset data
	XMM files
ATS	spacecraft attitude history
	OM files
IMI	imaging file
THX	tracking history file
WDX	window data auxiliary file
NPH	non-periodic housekeeping file
PEH	periodic housekeeping file
PAX	field acquisition data
RFX	priority reference frame data
PFX	priority fast mode data
FAE	event list (if fast mode was used)

Table 3.2: Pipeline Processing groupings.

Group ID	Contents
	PP files
PPSDAT	Contains the Calibration Index File (CIF) used in the pipeline processing (*CALIND*), PPS information, and the attitude history time series (*ATTTSR*) in gzipped FITS or ASCII format.
PPSGRA	Contains the time series plots, images, spectra, for EPIC, OM, RGS observations, and PPS run and observation summaries.
PPSMSG	ASCII file containing pipeline processing report
	EPIC files
CRSCOR	Contains PDF files of POSS II finding charts, HTML files of cross correlations with the SIMBAD data base, FITS tables for the detected sources
EANCIL	Contains the exposure maps in a variety of energy bands and the source-detection sensitivity maps for the EPIC instruments. The sensitivities are in units of counts s^{-1} corrected for vignetting and corresponding to a likelihood specified in the FITS header. The files are gzipped with a .FTZ extension.
EEVLIS	Contains calibrated photon event files for the EPIC detectors. If the files are sufficiently large they may be separated into two tar files. The files are gzipped fits files with a .FTZ extension.
ESKYIM	This group contains the event images in a variety of energy bands. The fits files are gzipped with a .FTZ extension.
ESRLIS	Contains EPIC observation source lists. There is an HTML page of the merged source list and gzipped fits tables of source lists from the different instruments and source detection tasks.
	OM files
OIMAGE which are EPIC images.)	Contains OM sky images in gzipped FITS format (not to be confused with the *OIMAGE* PNG
OMSLIS	Contains OM observation source lists in gzipped FITS format.
	RGS files
REVLIS	Contains the RGS source and event lists in gzipped FITS format
REXPIM	Contains the RGS exposure maps in gzipped FITS format
RIMAGE	Contains the RGS images in gzipped FITS format
RSPECT	Contains the RGS source and background spectra and affiliated files in gzipped FITS format

Table 3.3: Pipeline Processing data files

Group ID	File ID	Contents	File Type	View With
		EPIC files		
CRSCOR	FCHART	Finding chart	PDF	*Acrobat Reader*
	ROSIMG	ROSAT image of region	PDF	*Acrobat Reader*
	SNNNNN[1]	Source cross-correlation results	Zipped FITS	*fv*
	DNNNNN[1]	Catalog descriptions	HTML	web browser
	FNNNNN[1]	FOV cross-correlation result	Zipped FITS	*fv*
ESKYIM	IMAGE_8	Sky image 0.2 - 12.0 keV	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_1	Sky image 0.2 - 0.5 keV	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_2	Sky image 0.5 - 2.0 keV	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_3	Sky image 2.0 - 4.5 keV	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_4	Sky image 4.5 - 7.5 keV	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_5	Sky image 7.5 - 12.0 keV	Zipped FITS	*ds9, Ximage, fv*
EANCIL	EXPMAP8	Exposure map 0.2 - 12.0 keV	Zipped FITS	*ds9, Ximage, fv*
	EXPMAP1	Exposure map 0.2 - 0.5 keV	Zipped FITS	*ds9, Ximage, fv*
	EXPMAP2	Exposure map 0.5 - 2.0 keV	Zipped FITS	*ds9, Ximage, fv*
	EXPMAP3	Exposure map 2.0 - 4.5 keV	Zipped FITS	*ds9, Ximage, fv*
	EXPMAP4	Exposure map 4.5 - 7.5 keV	Zipped FITS	*ds9, Ximage, fv*
	EXPMAP5	Exposure map 7.5 - 12.0 keV	Zipped FITS	*ds9, Ximage, fv*
EEVLIS[2]	MIEVLI	MOS imaging mode event list	Zipped FITS	*ds9, fv, Xselect*
	PIEVLI	PN imaging mode event list	Zipped FITS	*ds9, fv, Xselect*
	TIEVLI	PN, MOS timing mode event list	Zipped FITS	*ds9, fv, Xselect*
ESRLIS	OBLSLI	Box-local detect source list	Zipped FITS	*fv*
	OBMSLI	Box-map detect source list	Zipped FITS	*fv*
	OMSRLI	Max-like detect source list	Zipped FITS	*fv*
	OBSMLI	Summary source list	Zipped FITS, HTML	*fv*, web browser
		RGS files		
REVLIS	SRCLI_	RGS Source Lists	Zipped FITS	*fv*
	EVENLI	RGS Event lists	Zipped FITS	*ds9, fv*
REXPIM	EXPMAP	RGS Exposure Maps	Zipped FITS	*ds9, Ximage, fv*
RSPECT	SRSPEC1	1st Order Source Spectrum	Zipped FITS	*ds9, Xspec, fv*
	SRSPEC2	2nd Order Source Spectrum	Zipped FITS	*ds9, Xspec, fv*
	BGSPEC1	1st Order Background Spectrum	Zipped FITS	*ds9, Xspec, fv*
	BGSPEC2	2nd Order Background Spectrum	Zipped FITS	*ds9, Xspec, fv*
	SBSPEC1	1st Order Source+Background Spectrum	Zipped FITS	*ds9, Xspec, fv*
	SBSPEC2	2nd Order Source+Background Spectrum	Zipped FITS	*ds9, Xspec, fv*
RIMAGE	ORDIMG	Images, disp. vs. X-disp	Zipped FITS	*ds9, Ximage, fv*
	IMAGE_	Images, disp. vs. PI	Zipped FITS	*Ximage, fv*
		OM files		
OIMAGE	SIMAGE	OM Sky Image	Zipped FITS	*ds9, Ximage, fv*
OMSLIS	SWSRLI	OM Source Lists	Zipped FITS	*fv*

[1] NNNNN - Alphanumeric ID
[2] Files for only those modes which were active will be included

Chapter 4

Setting Up and Running SAS

The Science Analysis Software (SAS, `http://xmm.esac.esa.int/sas/`), developed by the Survey Science Centre (SSC) and Science Operations Centre (SOC), is a suite of about 125 programs and scripts that perform data reduction, extraction, and some analysis of *XMM-Newton* data. The Pipeline Processing System (PPS), comprised of a superset of the SAS suite and Perl scripts, is run at Leicester University (`http://xmmssc-www.star.le.ac.uk/`) to create the basic data products provided to the Guest Observer from the satellite ancillary and science data. SAS is not designed for higher level scientific analysis such as spectral fitting and temporal analysis, but does provide for the creation of detector response files and barycentric corrected event timing information. SAS includes extensive EPIC and OM source-detection software. The SAS product files conform to OGIP FITS standards so any high-level analysis package used in high-energy astrophysics should theoretically be capable of processing *XMM-Newton* data. For example, the HEASoft package, `http://heasarc.gsfc.nasa.gov/docs/corp/software.html`, of the High Energy Astrophysics Science Archive Research Center (HEASARC, `http://heasarc.gsfc.nasa.gov/`) at NASA/GSFC and the CIAO package (`http://asc.harvard.edu/ciao/`) of the Chandra X-ray Observatory Center (`http://chandra.harvard.edu/`) can both be used with *XMM-Newton* data files.

4.1 Installation

The primary guide for the installation of SAS can be found through the SOC at `http://xmm.esac.esa.int/sas/current/installation/`. Because of the complexity of the SAS installation, it is strongly recommended that users download and install the binary executables rather than compiling SAS from source code (which also necessitates the purchase of commercial software). It should also be noted that "optional" components, while not needed for running SAS tasks from the command-line, are critical to running SAS from the GUI. These optional components are listed at the SOC page `http://xmm.esac.esa.int/sas/current/requirements/`.

4.2 Calibration Data

XMM-Newton data reduction and analysis requires extensive calibration data which must be available under a Current Calibration File (CCF) directory. Information on the CCF and instructions for downloading/mirroring the files can be found under the SOC *XMM-Newton* Calibration page (`http://xmm2.esac.esa.int/external/xmm_sw_cal/cal`) links to the CCF release notes. In addition, background event files and canned spectral response files can be found under `http://xmm2.esac.esa.int/external/xmm_sw_cal/background/index.shtml`.

4.3 SAS Environmental Parameters and Invocation

There are a few parameters which need to be set for the proper operation of SAS. Two are taken care of by the initialization script, `SAS_DIR` and `SAS_PATH`. These both set the SAS directory path. The remaining parameters, listed below, still need to be set. (The commands should, of course, be modified to be appropriate for your specific setup.)

```
setenv SAS_CCFPATH /full/path/to/CCF        Sets the directory path to the CCF data
setenv SAS_ODF /full/path/to/ODF            Sets the directory path to the ODF data
setenv SAS_CCF /full/path/to/ODF/ccf.cif    Sets the Calibration Index File (CIF) path
                                              and file name
```

Please note that `SAS_CCF` can also be set after the creation of the ccf.cif file with *cifbuild* (see §5.1). Also, while not necessary to run SAS, the following parameters are useful to know about and should be set.

```
setenv SAS_VERBOSITY 3                      Sets the verbosity, 1 => little, 10 => lot
setenv SAS_SUPPRESS_WARNING 3               Sets the warning level, 1 => little, 10 => lot
setenv SAS_IMAGEVIEWER ds9                  Sets the default image viewer; in this case,
                                              it is ds9, but should be set to whatever
                                              the user prefers.)
```

Finally, SAS is invoked by sourcing the script that came with the SAS package:

```
source /full/path/to/xmmsas_20090615_1801/setsas.csh      Initializes SAS
```

or

```
source /full/path/to/xmmsas_20090615_1801/setsas.sh       Alternate SAS initialization
```

At this point, SAS can be used on the command line or GUI; an introduction to the GUI can be found in §5.3. To verify the SAS-specific settings, type `env | grep SAS`.

It is strongly recommended that users include these environmental settings and make an alias to the initialization script in their login shell file (.cshrc, .bashrc, etc.)! It will save a lot of typing and lower the potential for frustration.

SAS tasks can be run equally well from the command line and from the SAS GUI. In this document we will demonstrate the use of some of the more commonly used tasks from both the GUI and command line, although in some instances, we only give command line examples. In these cases, the GUI can still be used – the user need only set the parameters there.

The MPE Analysis Guide, `http://www.mpe.mpg.de/xray/wave/xmm/cookbook/preparation/index.php` demonstrates many of the common tasks using GUIs.

4.3.1 SAS Helpful Hints

Command lines can often be quite long with a variety of parameters. To avoid considerable typing when creating command scripts, a feature of the GUI interface can be of assistance. When invoking a task through the GUI a copy of the full command appears in the dialog box, from where it can then be cut and pasted.

There are several useful features of the command-line interface that users should be aware of. 1) If the `dialog` parameter is included in the command line, the task GUI will pop up with all parameters in the command line preset. This allows the use of the GUI interfaces at the task level without having to go through the main SAS GUI. 2) If the `manpage` parameter is included in the command line, the task documentation will pop up in a web browser window. 3) In addition, the command `sashelp doc=`*sas_task* will pop up a web browser window with the documentation for the task *sas_task* as well.

The command documentation (i.e., the pages brought up by `sashelp doc=`*sas_task* or *sas_task* `manpage`) has an Errors section. Common warning messages produced by the tasks and their meanings are listed here. This feature is *very* useful.

4.4 SAS Syntax and Logic

4.4.1 Command Line Syntax

There is some flexibility in command line syntax in SAS. The following are all valid task calls on the command line that result in identical operations:

```
rgsproc withsrc=F
rgsproc withsrc=no
rgsproc withsrc='no'
rgsproc withsrc="no"
rgsproc --withsrc=no
rgsproc --withsrc='no'
rgsproc --withsrc="no"
```

However,

```
rgsproc -withsrc=F
rgsproc -withsrc=no
rgsproc -withsrc='no'
rgsproc -withsrc="no"
```

are not correct syntax.
One format is not "more correct" than another, and the choice of which to use is left to user preference. In this Guide, we adopt the simplest format, and use no dashes and only single quotation marks only when required, e.g.,

```
rgsproc withsrc=no orders='1 2 3'
```

where, in this case, the quotes provide the task with a list.

4.4.2 Table Syntax

When a task requires the use of a table within a file there are also several valid syntaxes, e.g.,

```
xmmselect table=filtered.fits:EVENTS
xmmselect table="filtered.fits:EVENTS"
xmmselect table=filtered.fits%EVENTS
```

do an identical operation in opening the `EVENTS` table inside the file `filtered.fits`.

4.4.3 Filtering Logic

Filtering event files requires some command of the SAS logical language which consists of familiar arithmetic and Boolean operators and functions. These, and their syntax, are described within the on-line documentation supplied with the software. Pull up the help document using:

```
sashelp doc=selectlib
```

Briefly, here are some commonly used terms:

Table 4.1: Commonly Used Filtering Logic

Symbol	Meaning
&&	and
!	not
||	or
==	equal
!=	not equal
&&!	and not
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
in	in

Chapter 5

Preparing the Data for Processing

Throughout this Primer, data from the HEASARC archive are used to illustrate how to run tasks for each instrument on XMM-Newton; new users are encouraged to use these sample data, though in should be noted that any data from the relevant instrument will do. Information about the example datasets are in Table 5.1.

A number of SAS tasks, including those which rerun the pipeline, require an up-to-date list of calibration files (the CIF file) and extended Observation Data File (ODF) summary file. These are obtained by running the tasks *cifbuild* and *odfingest*, respectively. Because of these dependencies, **it is strongly recommended that users run these tasks**, whether they plan to rerun the pipeline or not. Since these tasks are very straightforward to run, we will just call them from the command line and save the SAS GUI for more interesting tasks.

First, gunzip the all gzipped files in the ODF directory. If necessary, rename all files in the ODF directory to upper case. This can be done using the script provided by the NASA/GSFC XMM-Newton GOF.

```
gunzip ODF/*.gz
```

5.1 cifbuild

Many SAS tasks require calibration information from the Calibration Access Layer (CAL). Relevant files are accessed from the set of Current Calibration File (CCF) data using a CCF Index File (CIF). A CIF is included in the pipeline products (`*CALIND*`), but if the CCF has been updated it can be recreated by the user by simply typing

```
cd ODF


cifbuild
```

The task *cifdiff* can be used to compare the new CIF with the old:

```
cifdiff calindex1set=ccf.cif calindex2set=CALIND.FIT
```

where

- `calindex1set` – name of the first file to be compared, in this case the output from the current run of *cifbuild*
- `calindex2set` – name of the second file to be compared, in this case the (renamed) pipepine product file

If the CAL has changed, it is a good idea to repipeline the data using the new CIF. To help determine whether it is reasonable to repipeline the data, the CCF release notes should be examined (see §2.5). CCF files can be downloaded directly from the SOC website see (see §4.2).

To use the updated CIF file in further processing, you will need to reset the environment variable SAS_CCF:

```
setenv SAS_CCF /full/path/to/ODF/ccf.cif
```

Table 5.1: Example datasets used in this Guide.

Instrument	Chapter (Command line, SAS GUI)	ObsID	Object
EPIC (Imaging mode)	6, 7	0123700101	Lockman Hole
EPIC-PN (Timing mode)	8, 9	0400550201	Cen X-3
RGS	10, 11	0153950701	Mkn 421
OM (Image mode)	12.2	0123700101	Lockman Hole
OM (Fast mode)	12.3	0411081601	Mkn 421
OM (Grism mode)	12.4	0125320801	BPM 16274

5.2 odfingest

The task *odfingest* extends the Observation Data File (ODF) summary file with data extracted from the instrument housekeeping data files and the calibration database. It is only necessary to run it once on any dataset, and will cause problems if it is run a second time. If for some reason *odfingest* must be rerun, you must first delete the earlier file it produced. This file largely follows the naming convention described in §3.3.3, but has `SUM.SAS` appended to it. After running *odfingest*, you will need to reset the environment variable SAS_ODF to its output file. To run *odfingest* and reset the environment variable, simply type:

```
odfingest

setenv SAS_ODF /full/path/to/ODF/full_name_of_*SUM.SAS
```

The user will likely find it useful to alias these environment variable resets in their login shell (.cshrc, .bashrc, etc.).

The rest of this guide consists of step-by-step examples of how to reprocess and analyze data for the EPIC, RGS, and OM; the chapters for each instrument, and which way that SAS is used (on the command line or through a graphical user interface) are listed in Table 5.1. At the start of each chapter, it is assumed that the user has downloaded the data, initialized SAS, and set the environment variables, as listed in 4.3. It is also assumed that the CIF file is up-to-date, the `*SUM.SAS` file exists, and the appropriate environment variables were reset as needed.

5.3 Welcome to the SAS GUI

Many first time users will probably find the GUI more useful than the command line. To invoke it, simply type

```
sas &
```

A window will appear with two panels. The upper one will contain task names, what group they belong to (such as utility, epic, timing, calibration, etc.) and a short description of each. The lower one will contain information about environment variables, and as tasks are invoked, feedback from the tasks. The interface is shown in Fig. 5.1. Remember that tasks place output files in whatever directory you happened to be in when the SAS GUI was called, so either open and close the GUI in the directory where you want the output or move the files to the directory they should be in.

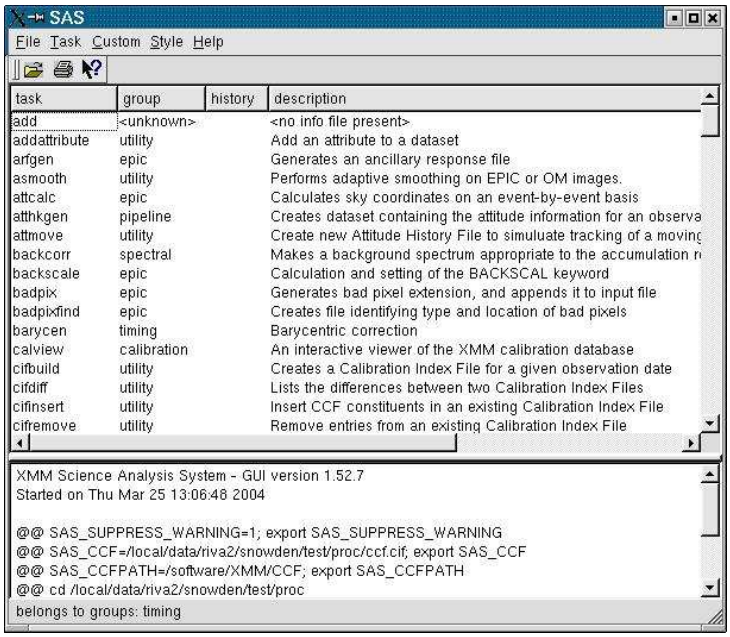


Figure 5.1: The SAS GUI. To locate and invoke a task one need only start typing the task name, and when it is highlighted, press carriage return. Alternatively, double-click on the task name.

Chapter 6

An EPIC Data Processing and Analysis Primer (Imaging Mode, Command Line)

So, you've received an *XMM-Newton* EPIC data set. What are you going to do with it? After checking what the observation consists of (see § 3.2), you should note when the observation was taken. If it is a recent observation, it was likely processed with the most recent calibrations and SAS, and you can immediately start to analyze the Pipeline Processed data. However, if it is more than a year old, it was probably processed with older versions of CCF and SAS prior to archiving, and the pipeline should be rerun to generate event files with the latest calibrations.

As noted in Chapter 4, a variety of analysis packages can be used for the following steps. However, as the SAS was designed for the basic reduction and analysis of *XMM-Newton* data (extraction of spatial, spectral, and temporal data), it will be used here for demonstration purposes. SAS will be required at any rate for the production of detector response files (RMFs and ARFs) and other observatory-specific requirements, though for the simple case of on-axis point sources the canned response files provided by the SOC can be used.

NOTE: For PN observations with very bright sources, out-of-time events can provide a serious contamination of the image. Out-of-time events occur because the read-out period for the CCDs can be up to $\sim 6.3\%$ of the frame time. Since events that occur during the read-out period can't be distinguished from others events, they are included in the event files but have invalid locations. For observations with bright sources, this can cause bright stripes in the image along the CCD read-out direction. For a more detailed description of this issue, check: `http://www.mpe.mpg.de/xray/wave/xmm/cookbook/preparation/index.php`

It is **strongly** recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If your data are recent, you need only to gunzip the files and prepare the data for processing (see §5. Feel free to skip the section on repipelining (§6.1 and proceed to later discussions. In any case, for simplicity, it is recommended that you change the name of the unzipped event file to something easy to type. For example, an MOS1 event list:

```
cp PPS/Piiiiiijjkk M1SlllMIEVLI0000.FTZ PROC/mos1.fits
```

where

`iiiiiijjkk` – observation number
`lll` – exposure number within the observation

Various analysis procedures are demonstrated using the Lockman Hole SV1 dataset, ObsID 0123700101, which definitely needs to be repipelined. The following procedures are applicable to all XMM-Newton datasets, so it is not required that you use this particular dataset; any observation should be sufficient.

If you simply want to have a quick look at your data, the ESKYIM files contain EPIC sky images in different energy bands whose ranges are listed in Table 3.3. While the zipped FITS files may need to be unzipped before display in *ds9* (depending on the version of *ds9*), they can be displayed when zipped using *fv* (*fv* is FITS file viewer available in the HEASoft package). In addition, the image of the total band pass for all three EPIC detectors is also provided in PNG format which can be displayed with a web browser. Also, the PP source list is provided in both zipped FITS format (readable by *fv*) and as an HTML file.

For detailed descriptions of PP data nomenclature, file contents, and which tasks can be used to view them, see Tables 3.2 and 3.3. For detailed descriptions of ODF data nomenclature and file contents, see Table 3.1.

6.1 Rerun the Pipeline

We assume that the data was prepared and environment variables were set according to §5. In the window where SAS was initialized, in your "processing directory" PROC, run *emchain* or *emproc* to produce calibrated photon event files for the MOS cameras, and *epchain* or *epproc* to do the same for the PN camera.

Note that *emproc* and *epproc* will automatically detect what mode the data were taken in. However, if you are using data that was not taken in Imaging mode and want to run *emchain* or *epchain*, you will need to set the relevant parameter, as shown below.

To process the MOS data, type

```
emchain
```

or

```
emproc
```

Similarly, to process the PN data, type

```
epchain
```

or

```
epproc
```

If the dataset has more than one exposure, a specific exposure can be accessed using the `exposure` parameter, e.g.:

```
emchain exposure=n
```

where n is the exposure number. To create an out-of-time event file for your PN data, add the parameter `withoutoftime` to your *epchain* invocation:

```
epchain withoutoftime=yes
```

By default, none of these tasks keep any intermediate files they generate. *Emchain* and *epchain* maintain the naming convention described in §3.3.3. *Emproc* and *epproc* designate their output event files with "*ImagingEvts.ds". In any case, you may want to name the new files something easy to type. For example, to rename one of the new MOS1 event files output from *emchain* or *emproc*, respectively, type

```
mv P0123700101M1S001MIEVLI0000.FIT mos1.fits
```

or

```
mv 0070_0123700101_EMOS1_S001_ImagingEvts.ds mos1.fits
```

6.2 Create and Display an Image

To create an image in sky coordinates, type

```
evselect table=mos1.fits withimageset=yes imageset=image.fits \
   xcolumn=X ycolumn=Y imagebinning=imageSize ximagesize=600 yimagesize=600
```

where

`table` – input event table
`withimageset` – make an image
`imageset` – name of output image
`xcolumn` – event column for X axis
`ycolumn` – event column for Y axis
`imagebinning` – form of binning, force entire image into a given size or bin by a specified number of pixels
`ximagesize` – output image pixels in X
`yimagesize` – output image pixels in Y

The output file `image.fits` can be viewed by using a standard FITS display, such as *ds9* (see Figure 6.1) :

```
ds9 image.fits &
```

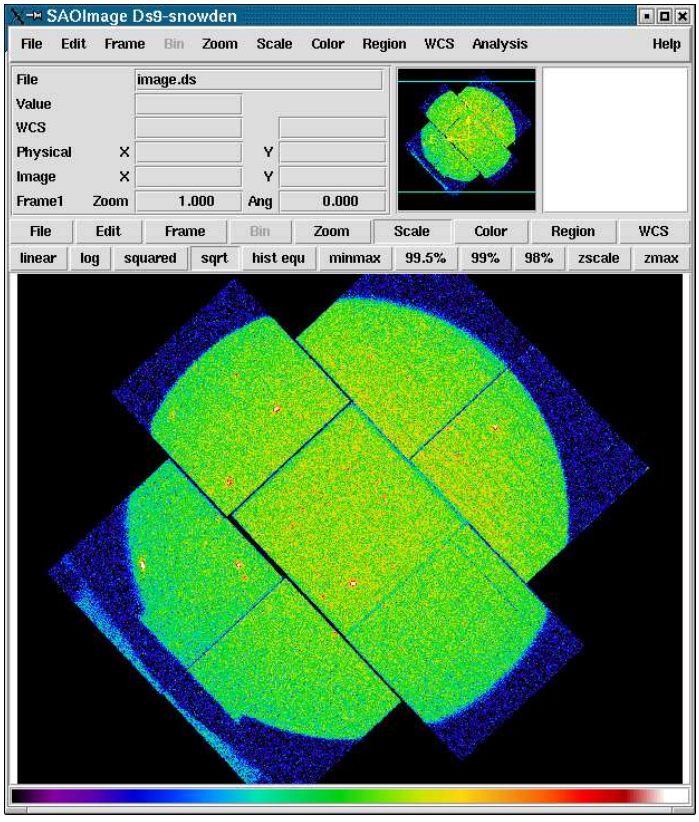


Figure 6.1: The MOS1 image, displayed in *ds9*.

6.3 Applying Standard Filters the Data

The filtering expressions for the MOS and PN are:

```
(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM
```

and

```
(PATTERN <= 12)&&(PI in [200:15000])&&#XMMEA_EP
```

The first two expressions will select good events with `PATTERN` in the 0 to 12 range. The `PATTERN` value is similar the `GRADE` selection for *ASCA* data, and is related to the number and pattern of the CCD pixels triggered for a given event.The `PATTERN` assignments are: single pixel events: `PATTERN == 0`, double pixel events: `PATTERN in [1:4]`, triple and quadruple events: `PATTERN in [5:12]`.

The second keyword in the expressions, `PI`, selects the preferred pulse height of the event; for the MOS, this should be between 200 and 12000 eV. For the PN, this should be between 200 and 15000 eV. This should clean up the image significantly with most of the rest of the obvious contamination due to low pulse height events. Setting the lower PI channel limit somewhat higher (e.g., to 300 eV) will eliminate much of the rest.

Finally, the `#XMMEA_EM` (`#XMMEA_EP` for the PN) filter provides a canned screening set of FLAG values for the event. (The FLAG value provides a bit encoding of various event conditions, e.g., near hot pixels or outside of the field of view.) Setting `FLAG == 0` in the selection expression provides the most conservative screening criteria and should always be used when serious spectral analysis is to be done on the PN. It typically is not necessary for the MOS.

It is a good idea to keep the output filtered event files and use them in your analyses, as opposed to re-filtering the original file with every task. This will save much time and computer memory. As an example, the Lockman Hole data's original event file is 48.4 Mb; the fully filtered list (that is, filtered spatially, temporally, and spectrally) is only 4.0Mb!

To filter the data, type

```
evselect table=mos1.fits withfilteredset=yes \
   expression='(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM' \
   filteredset=mos1_filt.fits filtertype=expression keepfilteroutput=yes \
   updateexposure=yes filterexposure=yes
```

where

`table` – input event table
`filtertype` – method of filtering
`expression` – filtering expression
`withfilteredset` – create a filtered set
`filteredset` – output file name
`keepfilteroutput` – save the filtered output
`updateexposure` – update exposure information in event list and in spectrum files
`filterexposure` – filter exposure extensions of event list with same time

6.4 Create and Display a Light Curve

Sometimes, it is necessary to use filters on time in addition to those mentioned above. This is because of soft proton background flaring, which can have count rates of 100 counts/sec or higher across the entire bandpass.

It should be noted that the amount of flaring that needs to be removed depends in part on the object observed; a faint, extended object will be more affected than a very bright X-ray source.

To determine if our observation is affected by background flaring, we can examine the light curve:

```
evselect table=mos1.fits withrateset=yes rateset=mos1_ltcrv.fits \
   maketimecolumn=yes timecolumn=TIME timebinsize=100 makeratecolumn=yes
```

```
fv mos1_ltcrv.fits &
```

where

`table` – input event table
`withrateset` – make a light curve
`rateset` – name of output light curve file
`maketimecolumn` – control to create a time column
`timecolumn` – time column label
`timebinsize` – time binning (seconds)
`makeratecolumn` – control to create a count rate column, otherwise a count column will be created

The output file `mos1_ltcrv.fits` can be viewed by using *fv*:

```
fv mos1_ltcrv.fits &
```

In the pop-up window, the RATE extension will be available in the second row (index 1, as numbering begins with 0). Select "PLOT" from this row, and select the column name and axis on which to plot it. The light curve is shown in Fig. 6.2.

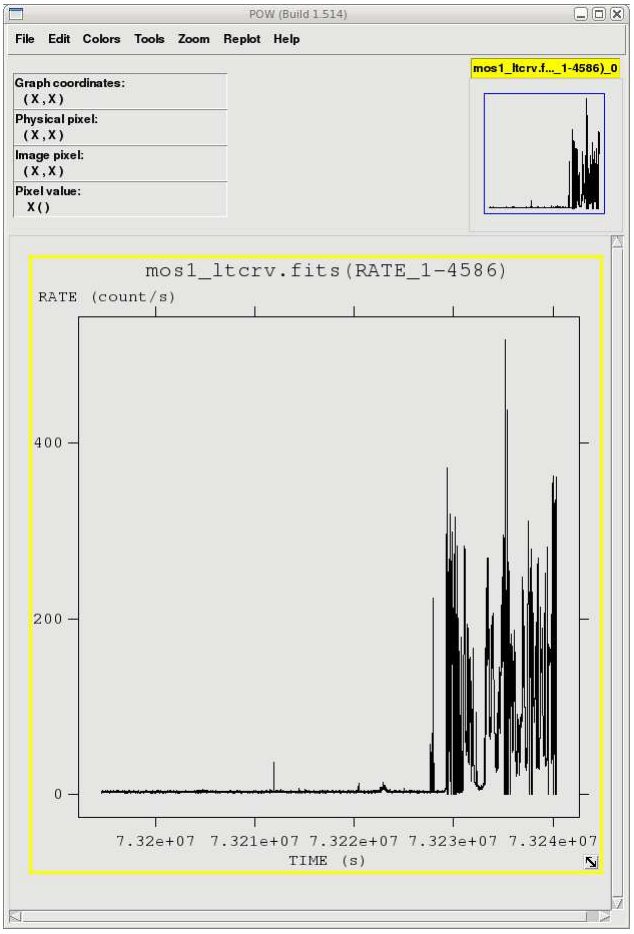


Figure 6.2: The light curve, displayed in *fv*.

6.5 Applying Time Filters the Data

Taking a look at the light curve, we can see that there is a very large flare toward the end of the observation and two much smaller ones in the middle of the exposure.

There are many ways to filter on time: with an explicit reference to the TIME parameter in the filtering expression; by making a secondary Good Time Interval (GTI) file with the task *tabgtigen*, which will allow you to filter on TIME or RATE; or by making a new GTI file with the task *gtibuild* using TIME as the filter. All of these will get the job done, so which to use is a matter of the user's preference. All of these are demonstrated below.

Filter on RATE with *tabgtigen*

Examining the light curve shows us that during non-flare times, the count rate is quite low, about 1.3 ct/s, with a small increase at 7.3223e7 seconds to about 6 ct/s. We can use that to generate the GTI file:

```
tabgtigen table=mos1_ltcrv.fits gtiset=gtiset.fits timecolumn=TIME \
   expression='(RATE <= 6)'
```

where

- `table` – input count rate table
- `expression` – filtering expression
- `gtiset` – output file name for selected GTI intervals
- `timecolumn` – time column

We can use *evselect* to apply it:

```
evselect table=mos1_filt.fits withfilteredset=yes \
   expression='GTI(gtiset.fits,TIME)' filteredset=mos1_filt_time.fits \
   filtertype=expression keepfilteroutput=yes \
   updateexposure=yes filterexposure=yes
```

where the parameters are as defined in §6.3.

Filter on TIME with *tabgtigen*

Alternatively, we could have chosen to make a new GTI file by noting the times of the flaring in the light curve and using that as a filtering parameter. The big flare starts around 7.32276e7 s, and the smaller ones are at 7.32119e7 s and 7.32205e7 s. The expression to remove these would be `(TIME <= 73227600)&&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])`. The syntax `&&(TIME < 73227600)` includes only events with times less than 73227600, and the "!" symbol stands for the logical "not", so use `&&!(TIME in [7.32118e7:7.3212e7])` to exclude events in that time interval. Once the new GTI file is made, we apply it with *evselect*.

```
tabgtigen table=mos1_ltcrv.fits gtiset=gtiset.fits timecolumn=TIME \
   expression= \
   '(TIME <= 73227600)&&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])'

evselect table=mos1_filt.fits withfilteredset=yes \
   expression='GTI(gtiset.fits,TIME)' filteredset=mos1_filt_time.fits \
   filtertype=expression keepfilteroutput=yes \
   updateexposure=yes filterexposure=yes
```

where the parameters are as defined above.

Filter on TIME with *gtibuild*

This method requires a text file as input. In the first two columns, enter the start and end times (in seconds) that you are interested in, and in the third column, indicate with either a + or - sign whether that region should be kept or removed. Each good (or bad) time interval should get its own line, with any optional comments preceeded by a "#". In the example case, we would write in our ASCII file (named gti.txt):

```
0 73227600 + # Good time from the start of the observation...
```

```
7.32118e7 7.3212e7 - # but without a small flare here,

7.32204e7 7.32206e7 - # and here.
```

and proceed to *gtibuild*:

```
gtibuild file=gti.txt table=gti.fits
```

where

`file` – input text file
`table` – output GTI file

And we apply it in the usual manner:

```
evselect table=mos1_filt.fits withfilteredset=yes \
   expression='GTI(gtiset.fits,TIME)' filteredset=mos1_filt_time.fits \
   filtertype=expression keepfilteroutput=yes \
   updateexposure=yes filterexposure=yes
```

where the parameters are as described in §6.3.

Filter on TIME by Explicit Reference

Finally, we could have chosen to forgo making a secondary GTI file altogether, and simply filtered on TIME with the standard filtering expression (see §6.3). In that case, the full filtering expression would be:

```
(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM
   &&(TIME <= 73227600) &&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])
```

This expression can then be used to filter the original event file, as shown in §6.3, or only the times can be used to filter the file that has already had the standard filters applied:

```
evselect table=mos1_filt.fits withfilteredset=yes \
  expression= \
 '(TIME<=73227600)&&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])' \
  filteredset=mos1_filt_time.fits filtertype=expression keepfilteroutput=yes \
  updateexposure=yes filterexposure=yes
```

where the keywords are as described in §6.3.

6.6 Source Detection with *edetect_chain*

The *edetect_chain* task does nearly all the work involved with EPIC source detection. It can process up to three intruments (both MOS cameras and the PN) with up to five images in different energy bands simultaneously. All images must have identical binning and WCS keywords. For this example, we will perform source detection on MOS1 images in two bands ("soft" X-rays with energies between 300 and 2000 eV, and "hard" X-rays, with energies between 2000 and 10000 eV) using the filtered event files produced here.

We will start by generating some files that *edetect_chain* needs: an attitude file and images of the sources in the desired energy bands, with the image binning sizes as needed according to the detector. For the MOS, the we'll let the binsize be 22.

The example uses the filtered event file produced in §6.5, with the assumption that it is located in the current directory.

First, make the attitude file by typing

```
atthkgen atthkset=attitude.fits timestep=1
```

where

`atthkset` – output file name
`timestep` – time step in seconds for attitude file

Next, make the soft and hard X-ray images with *evselect* by typing

```
evselect table=mos1_filt_time.fits withimageset=yes imageset=mos1-s.fits \
   imagebinning=binSize xcolumn=X ximagebinsize=22 ycolumn=Y yimagebinsize=22 \
   filtertype=expression expression='(FLAG == 0)&&(PI in [300:2000])'
```

and

```
evselect table=mos1_filt_time.fits withimageset=yes imageset=mos1-h.fits \
   imagebinning=binSize xcolumn=X ximagebinsize=22 ycolumn=Y yimagebinsize=22 \
   filtertype=expression expression='(FLAG == 0)&&(PI in [2000:10000])'
```

We will also make an image with both soft and hard X-rays for display purposes:

```
evselect table=mos1_filt_time.fits withimageset=yes imageset=mos1-all.fits \
   imagebinning=binSize xcolumn=X ximagebinsize=22 ycolumn=Y yimagebinsize=22 \
   filtertype=expression expression='(FLAG == 0)&&(PI in [300:10000])'
```

where the parameters are

`table` – event list
`withimageset` – flag to create an image
`imageset` – fits image name to be created
`imagebinning` – how to bin the image
`xcolumn` – table column to use for the X axis
`ximagebinsize` – binning in X axis
`ycolumn` – table column to use for the Y axis
`yimagebinsize` – binning in Y axis
`filtertype` – type of filtering
`expression` – filtering expression

Now we can run *edetect_chain*:

```
edetect_chain imagesets='mos1-s.fits mos1-h.fits' \
   eventsets='mos1_filt_time.fits' attitudeset=attitude.fits \
   pimin='300 2000' pimax='2000 10000' likemin=10 witheexpmap=yes \
   ecf='0.878 0.220' eboxl_list=eboxlist_l.fits \
   eboxm_list=eboxlist_m.fits eml_list=emllist.fits esp_withootset=no
```

where

`imagesets` – list of count images
`eventsets` – list of event files
`attitudeset` – attitude file name
`pimin` – list of minimum PI channels for the bands
`pimax` – list of maximum PI channels for the bands
`likemin` – maximum likelihood threshold
`witheexpmap` – create and use exposure maps
`ecf` – energy conversion factors for the bands
`eboxl_list` – output file name for the local sliding box source detection list
`eboxm_list` – output file name for the sliding box source detection in

background map mode list
`eml_list` – output file name for maximum likelihood source detection list
`esp_withootset` – Flag to use an out-of-time processed PN event file,
useful in cases where bright point sources have left streaks in the PN data
`esp_ooteventset` – The out-of-time processed PN event file

The energy conversion factors (ECFs) convert the source count rates into fluxes. The ECFs for each detector and energy band depend on the pattern selection and filter used during the observation. For more information, please consult the calibration paper "SSC-LUX-TN-0059", available at the XMM-Newton Science Operations Center or see Table 8 in the 3XMM Catalogue User Guide. Those used here are derived from PIMMS using the flux in the 0.1-10.0 keV band, a source power-law index of 1.9, an absorption of 0.5×10^{20} cm^{-2}.

We can display the results of *eboxdetect* using the task *srcdisplay* and produce a region file for the sources.

```
srcdisplay boxlistset=emllist.fits imageset=mos1-all.fits \
   regionfile=regionfile.txt sourceradius=0.01 withregionfile=yes
```

where

`boxlistset` – *eboxdetect* source list
`imageset` – image file name over which the source circles are to be plotted
`includesources` – flag to include the source positions on the display
`regionfile` – file name of output file containing source regions
`sourceradius` – radius of circle plotted to locate sources
`withregionfile` – flag to create a region file

Figure 6.3 shows the MOS1 event file overlayed with the detected sources.

6.7 Extract the Source and Background Spectra

Throughout the following, please keep in mind that some parameters are instrument-dependent. The parameter `specchannelmax` should be set to 11999 for the MOS, or 20479 for the PN. Also, for the PN, the most stringent filters, `(FLAG==0)&&(PATTERN<=4)`, must be included in the `expression` to get a high-quality spectrum.

For the MOS, the standard filters should be appropriate for many cases, though there are some instances where tightening the selection requirements might be needed. For example, if obtaining the best-possible spectral resolution is critical to your work, and the corresponding loss of counts is not important, only the single pixel events should be selected (`PATTERN==0`). If your observation is of a bright source, you again might want to select only the single pixel events to mitigate pile up (see §6.8 and §6.9 for a more detailed discussion).

In any case, you'll need to know spatial information about the area over which you want to extract the spectrum, so display the filtered event file with *ds9*:

```
ds9 mos1_filt_time.fits &
```

Select the object whose spectrum you wish to extract. This will produce a circle (extraction region), centered on the object. The circle's radius can be changed by clicking on it and dragging to the desired size. Adjust the size and position of the circle until you are satisfied with the extraction region; then, double-click on the region to bring up a window showing the center coordinates and radius of the circle. For this example, we will choose the source at (26188.5,22816.5) and set the extraction radius to 300 (in physical units).

To extract the source spectrum, type

```
evselect table='mos1_filt_time.fits' energycolumn='PI' withfilteredset=yes \
   filteredset='mos1_filtered.fits' keepfilteroutput=yes filtertype='expression' \
   expression='((X,Y) in CIRCLE(26188.5,22816.5,300))' \
   withspectrumset=yes spectrumset='mos1_pi.fits' spectralbinsize=5 \
   withspecranges=yes specchannelmin=0 specchannelmax=11999
```

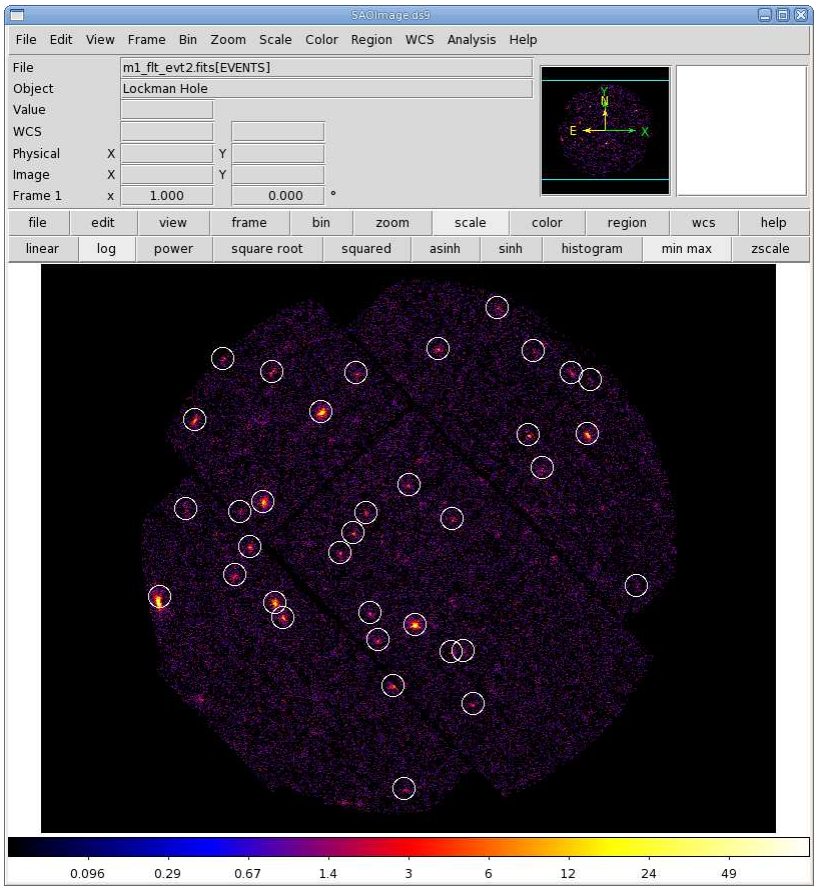


Figure 6.3: MOS1 event file overlayed with the detected sources.

where

`table` – the event file
`energycolumn` – energy column
`withfilteredset` – make a filtered event file
`keepfilteroutput` – keep the filtered file
`filteredset` – name of output file
`filtertype` – type of filter
`expression` – expression to filter by
`withspectrumset` – make a spectrum
`spectrumset` – name of output spectrum
`spectralbinsize` – size of bin, in eV
`withspecranges` – covering a certain spectral range
`specchannelmin` – minimum of spectral range
`specchannelmax` – maximum of spectral range

When extracting the background spectrum, follow the same procedures, but change the extraction area. For example, make an annulus around the source; this can be done using two circles, each defining the inner and outer edges of the annulus, then change the filtering expression (and output file name) as necessary.

To extract the background spectrum, type

```
evselect table=mos1_filt_time.fits energycolumn='PI' withfilteredset=yes \
   filteredset='bkg_filtered.fits' keepfilteroutput=yes filtertype='expression' \
```

```
   expression='((X,Y) in CIRCLE(26188.5,22816.5,1500))&&!((X,Y) in CIRCLE(26188.5,22816.5,500))'
\
   withspectrumset=yes spectrumset='bkg_pi.fits' spectralbinsize=5 \
   withspecranges=yes specchannelmin=0 specchannelmax=11999
```

where the keywords are as described above.

6.8 Check for Pile Up

Depending on how bright the source is and what modes the EPIC detectors are in, event pile up may be a problem. Pile up occurs when a source is so bright that incoming X-rays strike two neighboring pixels or the same pixel in the CCD more than once in a read-out cycle. In such cases the energies of the two events are in effect added together to form one event. If this happens sufficiently often, 1) the spectrum will appear to be harder than it actually is, and 2) the count rate will be underestimated, since multiple events will be undercounted. To check whether pile up may be a problem, use the SAS task *epatplot*. Heavily piled sources will be immediately obvious, as they will have a "hole" in the center, but pile up is not always so conspicuous. Therefore, **we recommend to always check for it**.

Note that this procedure requires as input the event files created when the spectrum was made, not the usual time-filtered event file.

To check for pile up in our Lockman Hole example, type

```
epatplot set=mos1_filtered.fits plotfile=mos1_epat.ps useplotfile=yes \
   withbackgroundset=yes backgroundset=bkg_filtered.fits
```

where

`set` – input events file \
`plotfile` – output postscript file \
`useplotfile` – flag to use file name from "plotfile" \
`withbackgroundset` – use background event set for background subtraction? \
`backgroundset` – name of background event file

The output of *epatplot* is a postscript file, `mos1_epat.ps`, which may be viewed with viewers such as *gv*, containing two graphs describing the distribution of counts as a function of PI channel; see Figure 8.4.

A few words about interpretting the plots are in order. The top is the distribution of counts versus PI channel for each pattern class (single, double, triple, quadruple), and the bottom is the expected pattern distribution (**smooth lines**) plotted over the observed distribution (**histogram**). The lower plot shows the model distributions for single and double events and the observed distributions. It also gives the ratio of observed-to-modeled events with 1-σ uncertainties for single and double pattern events over a given energy range. (The default is 0.5-2.0 keV; this can be changed with the `pileupnumberenergyrange` parameter.) If the data is not piled up, there will be good agreement between the modeled and observed single and double event pattern distributions. Also, the observed-to-modeled fractions for both singles and doubles in the 0.5-2.0 keV range will be unity, within errors. In contrast, if the data is piled up, there will be clear divergence between the modeled and observed pattern distributions, and the observed-to-modeled fraction for singles will be less than 1.0, and for doubles, it will be greater than 1.0.

Finally, when examining the plots, it should noted that the observed-to-modeled fractions can be inaccurate. Therefore, the agreement between the modeled and observed single and double event pattern distributions should be the main factor in determining if an observation is affected by pile up or not.

The source used in our Lockman Hole example is too faint to provide reasonable statistics for *epatplot* and is far from being affected by pile up. For comparison, an example of a bright source (from a different observation)

which is strongly affected by pileup is shown in Figure 6.5. Note that the observed-to-model fraction for doubles is over 1.0, and there is severe divergence between the model and the observed pattern distribution.

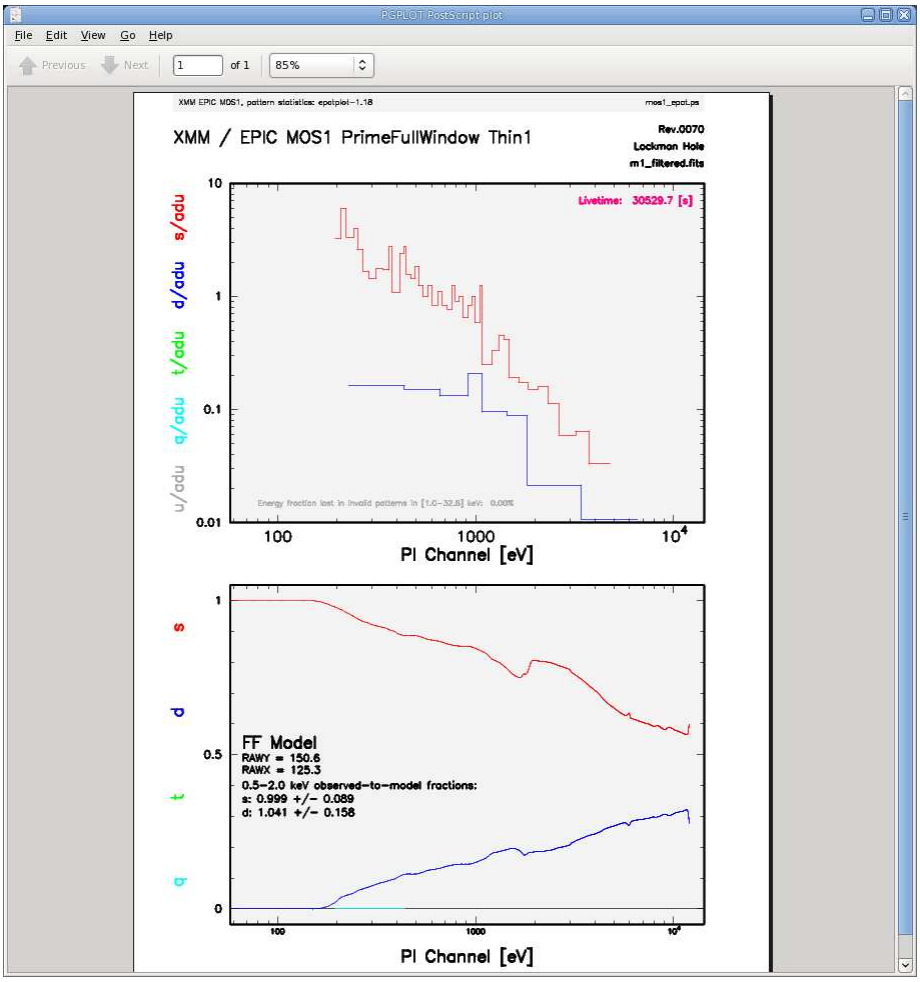


Figure 6.4: The output of *epatplot* for a very faint source without pileup. Note that in the lower plot, there are too few X-rays for *epatplot* to model.

6.9 My Observation is Piled Up! Now What?

If you're working with a different (much brighter) dataset that does show signs of pile up, there are a few ways to deal with it. First, using the region selection and event file filtering procedures demonstrated in earlier sections, you can excise the inner-most regions of a source (as they are the most heavily piled up), re-extract the spectrum, and continue your analysis on the excised event file. For this procedure, it is recommended that you take an iterative approach: remove an inner region, extract a spectrum, check with epatplot, and repeat, each time removing a slightly larger region, until the model and observed distribution functions agree. If you do this, be aware that removing too small a region with respect to the instrumental pixel size (1.1" for the MOS, 4.1" for the PN) can introduce systematic inaccuracies when calculating the source flux; these are less than 4%, and decrease to less than 1% when the excised region is more than 5 times the instrumental pixel half-size. In any case, be certain that the excised region is larger than the instrumental pixel size!

You can also use the event file filtering procedures to include only single pixel events (`PATTERN==0`), as these events are less sensitive to pile up than other patterns.

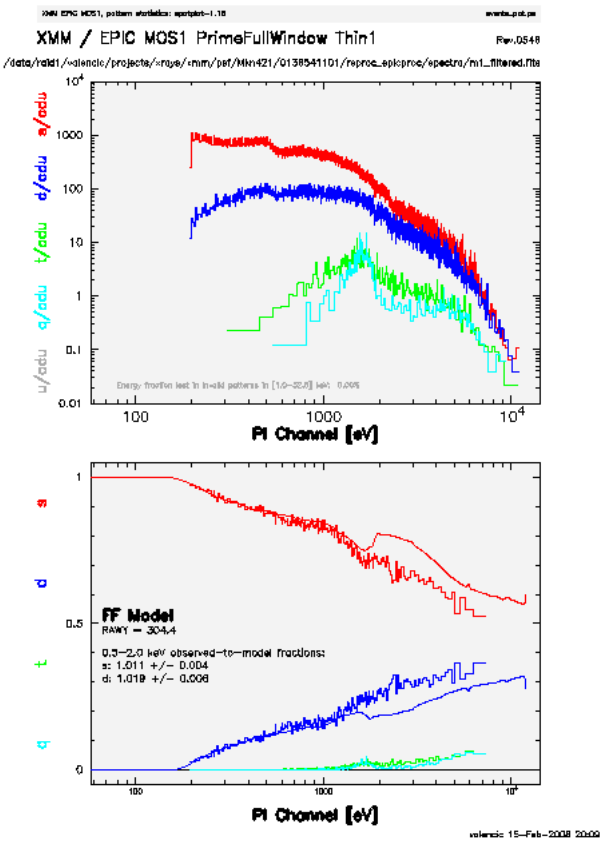


Figure 6.5: The output of *epatplot* for a heavily piled source. In the lower plot, there are large differences between the predicted and observed pattern distribution at energies above $\sim$ 1000 eV.

6.10 Determine the Spectrum Extraction Areas

Now that we are confident that our spectrum is not piled up, we can continue by finding the source and background region areas. This is done with the task *backscale*, which takes into account any bad pixels or chip gaps, and writes the result into the `BACKSCAL` keyword of the spectrum table. Alternatively, we can skip running *backscale*, and use a keyword in *arfgen* below. We will show both options for the curious.

To find the source and background extraction areas explicitly,

```
backscale spectrumset=mos1_pi.fits badpixlocation=mos1_filt_time.fits

backscale spectrumset=bkg_pi.fits badpixlocation=mos1_filt_time.fits
```

where

`spectrumset` – spectrum file
`badpixlocation` – event file containing the bad pixels

6.11 Create the Photon Redistribution Matrix (RMF) and Ancillary File (ARF)

Now that a source spectrum has been extracted, we need to reformat the detector response by making a redistribution matrix file (RMF) and ancillary response file (ARF). To make the RMF:

```
rmfgen rmfset=mos1_rmf.fits spectrumset=mos1_pi.fits
```

where

`rmfset` – output file
`spectrumset` – spectrum file

Now use the RMF, spectrum, and event file to make the ancillary file:

```
arfgen arfset=mos1_arf.fits spectrumset=mos1_pi.fits withrmfset=yes \
   rmfset=mos1_rmf.fits withbadpixcorr=yes badpixlocation=mos1_filt_time.fits
```

If we had not run *backscale*, we could set a keyword in *arfgen* to find the region area:

```
arfgen arfset=mos1_arf.fits spectrumset=mos1_pi.fits withrmfset=yes \
   rmfset=mos1_rmf.fits withbadpixcorr=yes badpixlocation=mos1_filt_time.fits \
   setbackscale=yes
```

where

`arfset` – output ARF file name
`spectrumset` – input spectrum file name
`withrmfset` – flag to use the RMF
`rmfset` – RMF file created by *rmfgen*
`withbadpixcorr` – flag to include the bad pixel correction
`badpixlocation` – file containing the bad pixel information; should be set to the event file from which the spectrum was extracted.
`setbackscale` – flag to calculate the area of the source region and write it to the BACKSCAL keyword in the spectrum header

At this point, the spectrum is ready to be analyzed, so skip ahead to prepare the spectrum for fitting (§13).

Chapter 7

An EPIC Data Processing and Analysis Primer (Imaging Mode, GUI)

So, you've received an *XMM-Newton* EPIC data set. What are you going to do with it? After checking what the observation consists of (see § 3.2), you should note when the observation was taken. If it is a recent observation, it was likely processed with the most recent calibrations and SAS, and you can immediately start to analyze the Pipeline Processed data. However, if it is more than a year old, it was probably processed with older versions of CCF and SAS prior to archiving, and the pipeline should be rerun to generate event files with the latest calibrations.

As noted in Chapter 4, a variety of analysis packages can be used for the following steps. However, as the SAS was designed for the basic reduction and analysis of *XMM-Newton* data (extraction of spatial, spectral, and temporal data), it will be used here for demonstration purposes. SAS will be required at any rate for the production of detector response files (RMFs and ARFs) and other observatory-specific requirements. (Although for the simple case of on-axis point sources the canned response files provided by the SOC can be used.)

NOTE: For PN observations with very bright sources, out-of-time events can provide a serious contamination of the image. Out-of-time events occur because the read-out period for the CCDs can be up to $\sim 6.3\%$ of the frame time. Since events that occur during the read-out period can't be distinguished from others events, they are included in the event files but have invalid locations. For observations with bright sources, this can cause bright stripes in the image along the CCD read-out direction. For a more detailed description of this issue, check: `http://www.mpe.mpg.de/xray/wave/xmm/cookbook/preparation/index.php`

It is **strongly** recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If your data are recent, you need only to gunzip the files and prepare the data for processing (see §5. Feel free to skip the discussion on repipelining (§7.1) and proceed to later discussions. In any case, for simplicity, it is recommended that you change the name of the unzipped event file to something easy to type. For example, an MOS1 event list:

```
cp PPS/Piiiiiijjkk M1SlllMIEVLI0000.FTZ PROC/mos1.fits
```

where

`iiiiiijjkk` – observation number
`lll` – exposure number within the observation

Various analysis procedures are demonstrated using the Lockman Hole SV1 dataset, ObsID 0123700101, which definitely needs to be repipelined. The following procedures are applicable to all XMM-Newton datasets, so it

is not required that you use this particular dataset; any observation should be sufficient.

If you simply want to have a quick look at your data, the ESKYIM files contain EPIC sky images in different energy bands whose ranges are listed in Table 3.3. While the zipped FITS files may need to be unzipped before display in *ds9* (depending on the version of *ds9*), they can be displayed when zipped using *fv* (*fv* is FITS file viewer available in the HEASoft package). In addition, the image of the total band pass for all three EPIC detectors is also provided in PNG format which can be displayed with a web browser. Also, the PP source list is provided in both zipped FITS format (readable by *fv*) and as an HTML file.

For detailed descriptions of PP data nomenclature, file contents, and which tasks can be used to view them, see Tables 3.2 and 3.3. For detailed descriptions of ODF data nomenclature and file contents, see Table 3.1.

7.1 Rerun the Pipeline

We assume that the data was prepared and environment variables were set according to §5, the GUI has been invoked (see §5.3), and we are in our working directory, "PROC".

From the upper window of the GUI, select *emproc* to process the MOS data, and *epchain* or *epproc* to process the PN data. Double-clicking the task will bring up pop-up windows that will allow you to change the (many) parameters; however, for most cases, the default settings are fine, so just click "Run".

If the dataset has more than one exposure, a specific exposure can be accessed with *epchain* by setting the *exposure* parameter in the "General" tab to the exposure number.

To create an out-of-time event file for your PN data, toggle the parameter *withoutoftime* on the *Epevents* tab to "yes".

By default, none of these tasks keep any intermediate files they generate. *Epchain* maintains the naming convention described in §3.3.3. *Emproc* and *epproc* designate their output event files with "*ImagingEvts.ds". In any case, you may want to name the new files something easy to type. For example, to rename one of the new MOS1 event files output from *emproc*, type

```
mv 0070_0123700101_EMOS1_S001_ImagingEvts.ds mos1.fits
```

Remember that tasks place output files in whatever directory you happened to be in when the SAS GUI was called, so either open and close the GUI in the directory where you want the output or move the files to the directory they should be in.

7.2 An Introduction to *xmmselect*

The task *xmmselect* is used for many procedures in the GUI. Like all tasks, it can easily be invoked by starting to type the name and pressing enter when it is highlighted.

When *xmmselect* is invoked a dialog box will first appear requesting a file name. You can either use the browser button or just type the file name in the entry area, "mos1.fits:EVENTS" in this case. To use the browser, select the file folder icon; this will bring up a second window for the file selection. Choose the desired event file, then the "EVENTS" extension in the right-hand column, and click "OK". The directory window will then disappear and you can click "Run" on the selection window.

When the file name has been submitted the *xmmselect* GUI (see Figure 7.1) will appear, along with a dialog box offering to display the selection expression. The selection expression will include the filtering done to this point on the event file, which for the pipeline processing includes for the most part CCD and GTI selections.

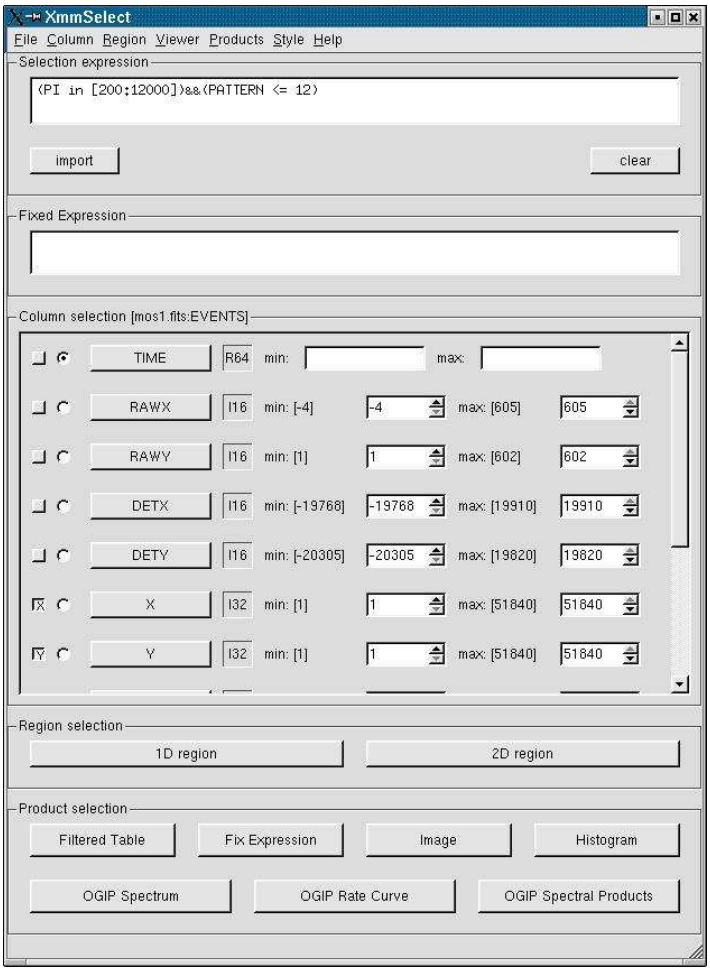


Figure 7.1: The *xmmselect* GUI.

7.3 Create and Display an Image

To create an image in sky coordinates by using the *xmmselect*, call *xmmselect* and load the event file as in §7.2. Then,

1) Check the square boxes to the left of the "X" and "Y" entries.
2) Click the "Image" button near the bottom of the page. This brings up the *evselect* GUI (see Figure 7.2).
3) In the `imageset` box, enter the name of the output file, in this case, `image.fits`.
4) Click the "Run" button on the lower left corner of the *evselect* GUI.

Different binnings and other selections can be invoked by accessing the "Image" tab at the top of the GUI. The default settings are reasonable, however, for a basic image. The resultant image is written to the file `image.fits`, and is automatically displayed using *ds9* (see Figure 7.3).

7.4 Applying Standard Filters the Data

The filtering expressions for the MOS and PN are:

```
(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM
```

and

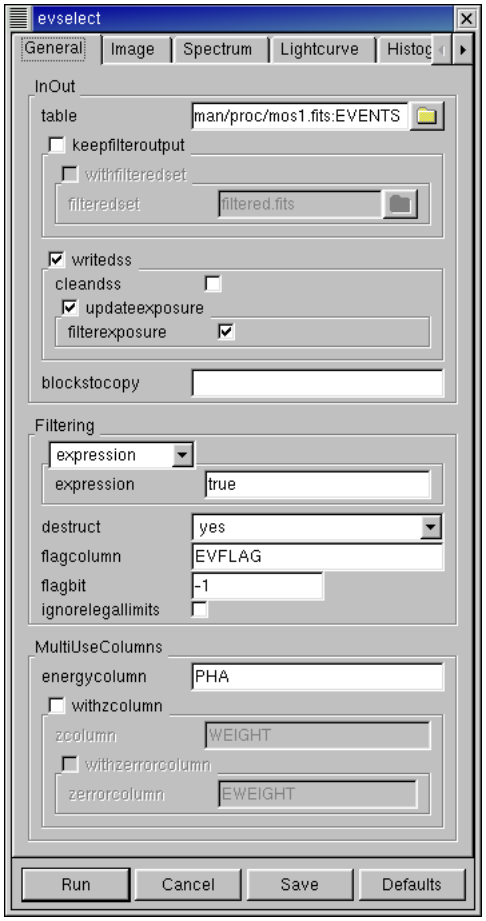


Figure 7.2: The *evselect* GUI.

```
(PATTERN <= 12)&&(PI in [200:15000])&&#XMMEA_EP
```

The first two expressions will select good events with `PATTERN` in the 0 to 12 range. The `PATTERN` value is similar the `GRADE` selection for *ASCA* data, and is related to the number and pattern of the CCD pixels triggered for a given event.The `PATTERN` assignments are: single pixel events: `PATTERN == 0`, double pixel events: `PATTERN in [1:4]`, triple and quadruple events: `PATTERN in [5:12]`.

The second keyword in the expressions, `PI`, selects the preferred pulse height of the event; for the MOS, this should be between 200 and 12000 eV. For the PN, this should be between 200 and 15000 eV. This should clean up the image significantly with most of the rest of the obvious contamination due to low pulse height events. Setting the lower PI channel limit somewhat higher (e.g., to 300 eV) will eliminate much of the rest.

Finally, the `#XMMEA_EM` (`#XMMEA_EP` for the PN) filter provides a canned screening set of FLAG values for the event. (The FLAG value provides a bit encoding of various event conditions, e.g., near hot pixels or outside of the field of view.) Setting `FLAG == 0` in the selection expression provides the most conservative screening criteria and should always be used when serious spectral analysis is to be done on the PN. It typically is not necessary for the MOS.

It is a good idea to keep the output filtered event files and use them in your analyses, as opposed to re-filtering the original file with every task. This will save much time and computer memory. As an example, the Lockman Hole data's original event file is 48.4 Mb; the fully filtered list (that is, filtered spatially, temporally, and spectrally) is only 4.0Mb!

To filter the data using *xmmselect*,

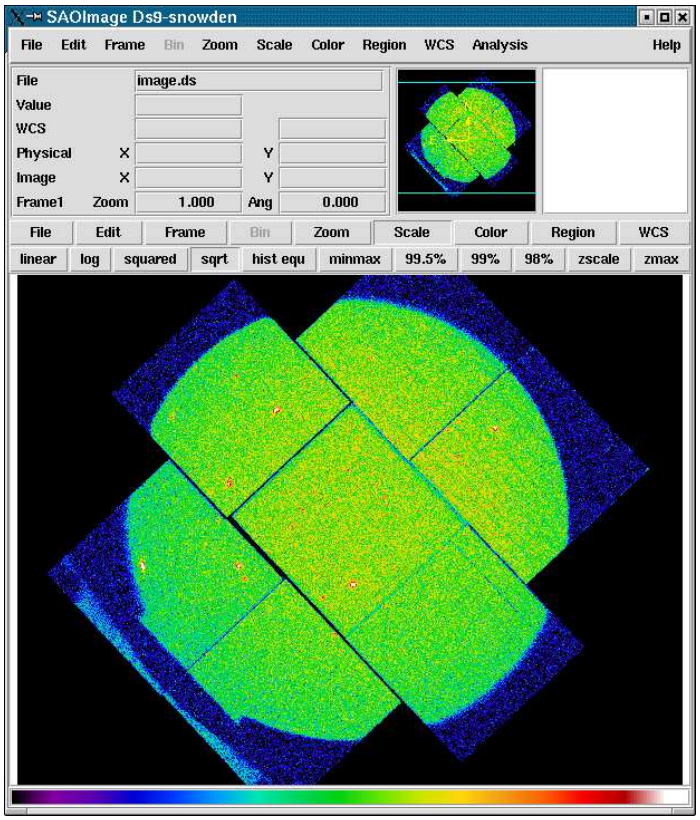


Figure 7.3: The MOS1 image, displayed in *ds9*.

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window:

   `(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM`

2) Click on the "Filtered Table" box at the lower left of the *xmmselect* GUI.

3) Change the *evselect* `filteredset` parameter, the output file name, to something useful, e.g., `mos1_filt.fits`

4) Click "Run".

7.5 Create and Display a Light Curve

Sometimes, it is necessary to use filters on time in addition to those mentioned above. This is because of soft proton background flaring, which can have count rates of 100 counts/sec or higher across the entire bandpass.

It should be noted that the amount of flaring that needs to be removed depends in part on the object observed; a faint, extended object will be more affected than a very bright X-ray source.

To determine if our observation is affected by background flaring, we can make a light curve with *xmmselect*. Load the event file as shown in §7.2. Then,

1) Check the round box to the left of the "Time" entry.

2) Click on the "OGIP Rate Curve" button near the bottom of the page. This brings up the *evselect* GUI (see Figure 7.2).

3) Click on the "Lightcurve" tab and change the "timebinsize" to a reasonable amount, e.g. 10 or 100 s. In the "rateset" textbox, enter the name of the output file, for example, `mos1_ltcrv.fits`.

4) Click on the "Run" button at the lower left corner of the *evselect* GUI.

The resultant light curve is displayed automatically using Grace (see Figure 7.4). It can also be viewed using *fv*; on the command line, type
or, alternatively,

```
fv mos1_ltcrv.fits &
```

In the *fv* pop-up window, the RATE extension will be available in the second row (index 1, as numbering begins with 0). Select "PLOT" from this row, and select the column name and axis on which to plot it.

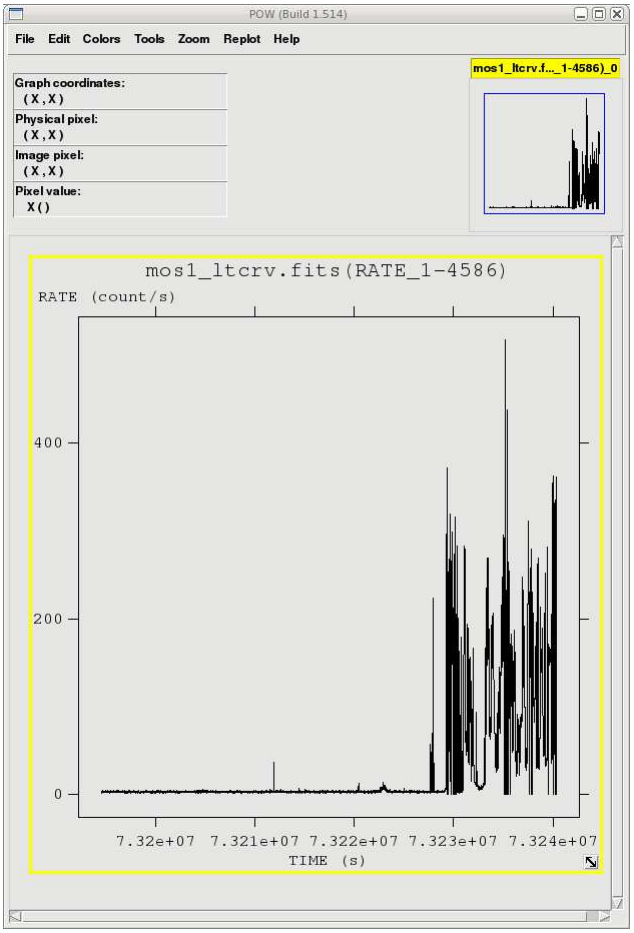


Figure 7.4: The light curve, displayed in *fv*.

7.6 Applying Time Filters the Data

Taking a look at the light curve, we can see that there is a very large flare toward the end of the observation and two much smaller ones in the middle of the exposure.

There are several ways to filter an event file: on TIME, with an explicit reference to the TIME parameter in the filtering expression or by creating a secondary Good Time Interval (GTI) file, or on RATE, which requires making a new GTI file. New GTI files are easily made with the task *tabgtigen* or *gtibuild*. These are discussed in detail below. Any of these methods will produce a cleaned file, so which one to use is a matter of the user's preference.

Filter on RATE With *tabgtigen*

Examining the light curve shows us that during non-flare times, the count rate is quite low, about 1.3 ct/s, with a small increase at 7.3223e7 seconds to about 6 ct/s. We can use that to generate the GTI file by calling *tabgtigen* from the SAS GUI and loading the light curve's RATE table as the input table. Then,

1) Edit the output file name, the `gtiset` parameter; here, we will use `gtiset.fits`.
2) Enter the filtering expression, `(RATE <= 6)`.
3) Click "Run".

The new GTI file can be applied with *xmmselect*. With the `mos1_filt.fits` event file loaded,

1) In the "Selection Expression" box, type `GTI(gtiset.fits,TIME)`.
2) Click on the "Filtered Table" box at the lower left of the *xmmselect* GUI.
3) Change the *evselect* `filteredset` parameter, the output file name, to something useful; here, we will use `mos1_filt_time.fits`.
4) Click "Run".

Filter on TIME With *tabgtigen*

Alternatively, we could have chosen to make a new GTI file by noting the times of the flaring in the light curve and using that as a filtering parameter. The big flare starts around 7.32276e7 s, and the smaller ones are at 7.32119e7 s and 7.32205e7 s. The expression to remove these would be `(TIME <= 73227600)&&! (TIME IN [7.32118e7:7.3212e7])&&(TIME IN [7.32204e7:7.32206e7])`. The syntax `&&(TIME < 73227600)` includes only events with times less than 73227600, and the "!" symbol stands for the logical "not", so use `&&!(TIME in [7.32118e7:7.3212e7])` to exclude events in that time interval. To use these filtering parameters, call *tabgtigen* from the SAS GUI and load the light curve's RATE table as the input table. Then,

1) Edit the output file name, the `gtiset` parameter; here, we will use `gtiset.fits`.
2) Enter the filtering expression, `(TIME <= 73227600)&&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])`.
3) Click "Run".

The new GTI file can be applied with *xmmselect*. With the `mos1_filt.fits` event file loaded,

1) In the "Selection Expression" box, type `GTI(gtiset.fits,TIME)`.
2) Click on the "Filtered Table" box at the lower left of the *xmmselect* GUI.
3) Change the *evselect* `filteredset` parameter, the output file name, to something useful; here, we will use `mos1_filt_time.fits`.
4) Click "Run".

Filter on TIME With *gtibuild*

This task requires a text file as input. In the first two columns, enter the start and end times (in seconds) that you are interested in, and in the third column, indicate with either a + or - sign whether that region should be kept or removed. Each good (or bad) time interval should get its own line. In the example case, we would write in our ASCII file (named `gti.txt`):

```
0 73227600 + # Good time from the start of the observation...
7.32118e7 7.3212e7 - # but without a small flare here,
7.32204e7 7.32206e7 - # and here.
```

and proceed to *gtibuild*. Invoke the task, then

1) Enter the name of the text file for the `file` parameter.

2) Enter the output name in the `table` parameter; we will use `gtiset.fits`

3) Click "Run".

Filter on TIME by Explicit Reference

Finally, we could have chosen to forgo using *tabgtigen* or *gtibuild* altogether, and simply filtered on TIME with the standard filtering expression, seen in §7.4. In that case, the full filtering expression would be:

```
(PATTERN <= 12)&&(PI in [200:12000])&&#XMMEA_EM&&
   (TIME <= 73227600)&&!(TIME IN [7.32118e7:7.3212e7])&&!(TIME IN [7.32204e7:7.32206e7])
```

This expression can then be used to filter the original event file, or only the times can be used to filter the file that has already had the standard filters applied. To do this, load the filtered event file `mos_filt.fits` in *xmmselect* by going to "File → New Table" at the top of the window. Then,

1) In the "Selection Expression" box, enter `(TIME <= 73227600) &&!`
   `(TIME IN [7.32118e7:7.3212e7]) &&!  (TIME IN [7.32204e7:7.32206e7])`

2) Click on the "Filtered Table" box at the lower left of the *xmmselect* GUI.

3) Change the *evselect* `filteredset` parameter, the output file name, to something useful; here, we will use `mos1_filt_time.fits`.

4) Click "Run".

7.7 Source Detection with *edetect_chain*

The *edetect_chain* task does nearly all the work involved with EPIC source detection. It can process up to three intruments (both MOS cameras and the PN) with up to five images in different energy bands simultaneously. All images must have identical binning and WCS keywords. For this example, we will perform source detection on MOS1 images in two bands ("soft" X-rays with energies between 300 and 2000 eV, and "hard" X-rays, with energies between 2000 and 10000 eV) using the filtered event file produced in §7.6.

We will start by generating some files that *edetect_chain* needs: an attitude file and images of the sources in the desired energy bands, with the image binning sizes as needed according to the detector. For the MOS, we'll let the binsize be 22.

First, make the attitude file by calling *atthkgen*. Then,

1) Verify that `timestep` is set to `1`. Set the `atthkset` keyword to the desired output file name, for example, `attitude.fits`.

2) Click "Run".

Next, make the soft and hard X-ray images. We'll also make an image that includes both bands, for display purposes. Call *evselect*, then

1) In the "General" tab, set the "Table" parameter to the MOS1 event file name (`mos1_filt_time.fits`). Confirm that "Filtertype" is set to "expression", and in the "Expression" text area, type `(FLAG == 0)&&(PI in [300:2000])`.

2) In the "Image" tab, check the `withimageset` box, and enter the desired output image name; we will use `mos1-s.fits`. Set `xcolumn` to X and `ycolumn` to Y. Set `Binning` to `binSize`, `ximagebinsize` to 22, and `yimagebinsize` to 22.

3) Click "Run".

Follow the same procedure to make the hard X-ray image, changing the output name to `mos1-h.fits` and the filtering expression to `(FLAG == 0)&&(PI in [2000:10000])`. For our combined band image, we'll set the output name to `mos1-all.fits` and the filtering expression to `(FLAG == 0)&&(PI in [300:10000])`.

Now we can run *edetect_chain*. Call the task, and then

1) In the "0" tab, in the `imagesets` text area, type: `mos1-s.fits mos1-h.fits` In the `eventsets` area, enter the names of the event file: `mos1_filt_time.fits`. In the `attitudeset` area, enter the name of the attitude file made by the task *atthkgen* (`attitude.fits`). Set the `pimin` keyword to the minimum PI values (in eV) for the input images by typing: `300 2000`, and do similar for the maximum values for `pimax` (`2000 10000`). Set the `likemin` parameter to 10, `witheexpmap` to `yes`, `ecf` to `0.878 0.220`.

2) In the "1" tab, set `eboxl_list` to `eboxlist_l.fits` and `eboxm_list` to `eboxlist_m.fits`.

3) In the "2" tab, set `esp_withootset` to `no` and `eml_list` to `emllist.fits`.

4) Click "Run".

The energy conversion factors (ECFs) convert the source count rates into fluxes. The ECFs for each detector and energy band depend on the pattern selection and filter used during the observation. For more information, please consult the calibration paper "SSC-LUX-TN-0059", available at the XMM-Newton Science Operations Center or see Table 8 in the 3XMM Catalogue User Guide. Those used here are derived from PIMMS using the flux in the 0.1-10.0 keV band, a source power-law index of 1.9, an absorption of 0.5×10^{20}.

We can display the results of *eboxdetect* using the task *srcdisplay* and produce a region file for the sources. Call *srcdisplay*, then

1) Set `boxlistset` to `emllist.fits`. Confirm that `withimageset` is checked, and set `imageset` to `mos1-all.fits`. Check `withregionfile`, and set `regionfile` to `regionfile.txt`. Confirm that `sourceradius` is set to 0.01.

2) Click "Run".

Figure 7.5 shows the MOS1 event file overlayed with the detected sources.

7.8 Extract the Source and Background Spectra

Throughout the following, please keep in mind that some parameters are instrument-dependent. The parameter `specchannelmax` should be set to 11999 for the MOS, or 20479 for the PN. Also, for the PN, the most stringent filters, `(FLAG==0)&&(PATTERN<=4)`, must be included in the `expression` to get a high-quality spectrum.

For the MOS, the standard filters should be appropriate for many cases, though there are some instances where tightening the selection requirements might be needed. For example, if obtaining the best-possible spectral resolution is critical to your work, and the corresponding loss of counts is not important, only the single pixel events should be selected (`PATTERN==0`). If your observation is of a bright source, you again might want to select only the single pixel events to mitigate pile up (see §7.9 and §7.10 for a more detailed discussion).

To extract the source spectrum, load the filtered file `mos1_filt_time.fits` into *xmmselect* if it isn't already loaded. Then,

1) Make an image (see §7.3). It will be displayed automatically in a *ds9* window.

2) In the "Image" tab, enter the name fothe output file in the `imageset` box; we will use `mos1_image.fits`. Different binnings and other selections can be invoked, but the defaults are reasonable for a basic image.

3) Click "Run". The image will be displayed automatically in a *ds9* window.

4) Click on the object whose spectrum you wish to extract. This will produce a circle (extraction region), centered on the object. The circle's radius can be changed by clicking on it. Adjust the size and position of the circle until you are satisfied with the extraction region. We will use the source at x=26188.5 and y=22816.

5) Click on "2D Region" in the *xmmselect* GUI. This transfers the region information into the "Selection Expression" text area, for example, `((X,Y) IN circle(26188.5,22816,300))`. Click the round button next the `PI` column on the *xmmselect* GUI, then click on "OGIP Spectrum". This will bring up the *evselect* GUI.

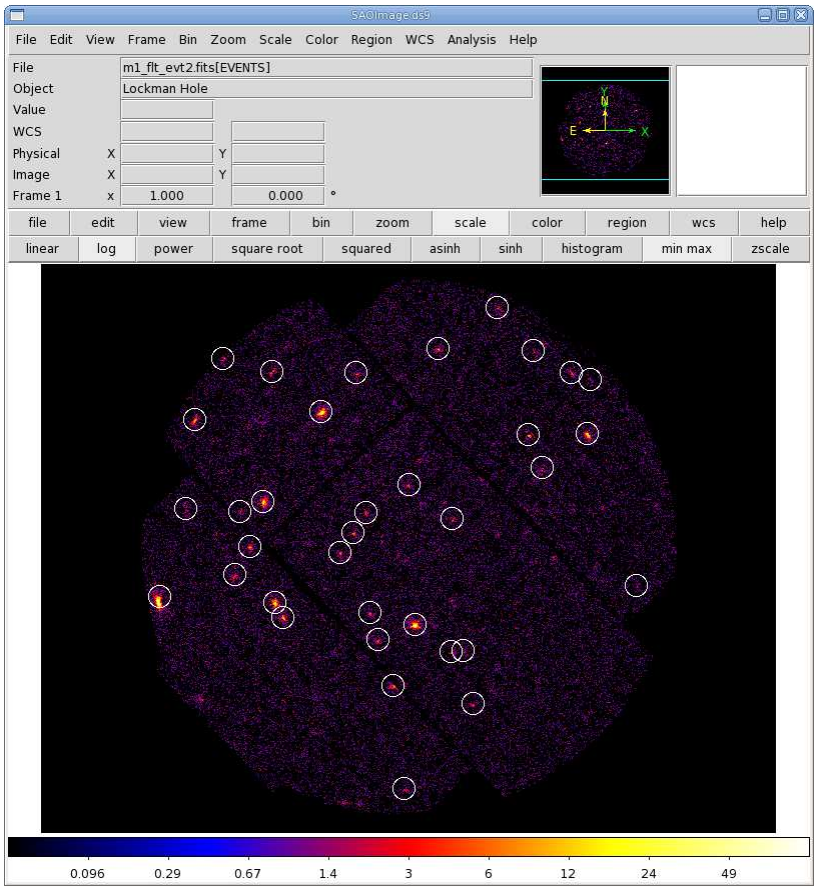


Figure 7.5: MOS1 event file with the detected sources.

6) In the "General" tab, check `keepfilteroutput` and `withfilteredset`. In the `filteredset` box, enter the name of the event file output. We will use `mos1_filtered.fits`.

7) Select the "Spectrum" tab of the *evslect* GUI to set the file name and binning parameters for the spectrum. Confirm that `withspectrumset` is checked. Set `spectrumset` to the desired output name, in this case, `mos1_pi.fits`. Confirm that `withspecranges` is checked. Confirm that `specchannelmin` is 0 and `specchannelmax` is `11999` for the MOS, or `20479` for the PN.

8) Click "Run".

The background spectrum can be extracted following the same method, setting the region to an annulus around the source: `((X,Y) in CIRCLE(26188.5,22816.5,1500))&&!((X,Y) in CIRCLE(26188.5,22816.5,500))`. We will call the filtered event file `bkg_filtered.fits` and the output spectrum `bkg_pi.fits`.

7.9 Check for Pile Up

Depending on how bright the source is and what modes the EPIC detectors are in, event pile up may be a problem. Pile up occurs when a source is so bright that incoming X-rays strike two neighboring pixels or the same pixel in the CCD more than once in a read-out cycle. In such cases the energies of the two events are in effect added together to form one event. If this happens sufficiently often, 1) the spectrum will appear to be harder than it actually is, and 2) the count rate will be underestimated, since multiple events will be undercounted. To check whether pile up may be a problem, use the SAS task *epatplot*. Heavily piled sources will be immediately obvious, as they will have a "hole" in the center of their image, but pile up is not always so

conspicuous. Therefore, we recommend to always check for it.

Note that this procedure requires as input the event file created when the spectrum was made, not the usual time-filtered event file.

To check for pile up, invoke *epatplot*. Then,

1) In the "0" tab, enter the name of the event file that was made when the spectrum was extracted, `mos1_filtered.fits`, in the `set` box. We want to send the output to a ps file, so set `useplotfile` to `yes`, and enter the file name in the `plotfile` box. We will use `mos1_epat.ps`.
2) In the "1" tab, set `withbackgroundset` to `yes` and enter the name of the event file that was made when the background spectrum was made, `bkg_filtered.fits`.
3) Click "Run".

The output of *epatplot* is a postscript file, `mos1_epat.ps`, which may be viewed with viewers such as *gv*, containing two graphs describing the distribution of counts as a function of PI channel, as seen in Figure 7.6.

A few words about interpretting the plots are in order. The top is the distribution of counts versus PI channel for each pattern class (single, double, triple, quadruple), and the bottom is the expected pattern distribution (smooth lines) plotted over the observed distribution (histogram). The lower plot shows the model distributions for single and double events and the observed distributions. It also gives the ratio of observed-to-modeled events with 1-σ uncertainties for single and double pattern events over a given energy range. (The default is 0.5-2.0 keV; this can be changed with the `pileupnumberenergyrange` parameter.) If the data is not piled up, there will be good agreement between the modeled and observed single and double event pattern distributions. Also, the observed-to-modeled fractions for both singles and doubles in the specified energy range will be unity, within errors. In contrast, if the data is piled up, there will be clear divergence between the modeled and observed pattern distributions, and the observed-to-modeled fraction for singles will be less than 1.0, and for doubles, it will be greater than 1.0.

Finally, when examining the plots, it should noted that the observed-to-modeled fractions can be inaccurate. Therefore, the agreement between the modeled and observed single and double event pattern distributions should be the main factor in determining if an observation is affected by pile up or not.

The source used in our Lockman Hole example is too faint to provide reasonable statistics for *epatplot* and is far from being affected by pile up. For comparison, an example of a bright source (from a different observation) which is strongly affected by pileup is shown in Figure 7.7. Note that the observed-to-model fraction for doubles is over 1.0, and there is severe divergence between the model and the observed pattern distribution.

7.10 My Observation is Piled Up! Now What?

If you are working with a different (much brighter) dataset that does show signs of pie up, there are a few ways to deal with it. First, using the region selection and event file filtering procedures demonstrated in earlier sections, you can excise the inner-most regions of a source (as they are the most heavily piled up), re-extract the spectrum, and continue your analysis on the excised event file. For this procedure, it is recommended that you take an iterative approach: remove an inner region, extract a spectrum, check with *epatplot*, and repeat, each time removing a slightly larger region, until the model and observed distribution functions agree. If you do this, be aware that removing too small a region with respect to the instrumental pixel size (1.1" for the MOS, 4.1" for the PN) can introduce systematic inaccuracies when calculating the source flux; these are less than 4%, and decrease to less than 1% when the excised region is more than 5 times the instrumental pixel half-size. In any case, be certain that the excised region is larger than the instrumental pixel size!

You can also use the event file filtering procedures to include only those events with `PATTERN==0`, as these events are less sensitive to pile up than other patterns.

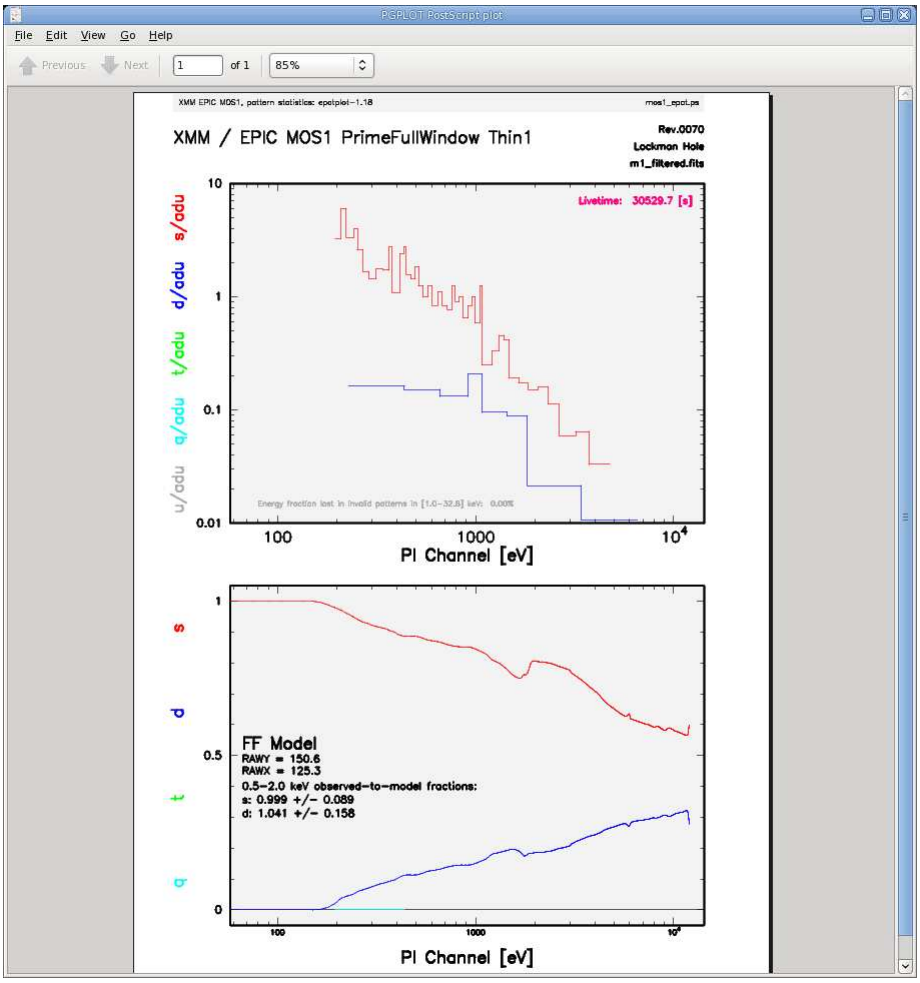


Figure 7.6: The output of *epatplot* for a very faint source without pileup. Note that in the lower plot, there are too few X-rays for *epatplot* to model.

7.11 Determine the Spectrum Extraction Areas

Now that we are confident that our spectrum is not piled up, we can continue by finding the source and background region areas. This is done with the task *backscale*, which takes into account any bad pixels or chip gaps, and writes the result into the BACKSCAL keyword of the spectrum table. Alternatively, we can skip running *backscale*, and use a keyword in *arfgen* below. We will show both options for the curious.

To find the source extraction area explicitly, call *backscale* and then

1) In the "Main" tab, enter the name of the spectrum, `mos1_pi.fits`.
2) In the "Effects" tab, confirm that `withbadpixcorr` is checked, and enter the name of the event file in `badpixlocation`.
3) Click "Run".

Follow the same steps to find the background spectrum area, changing the input spectrum file to `bkg_pi.fits`.

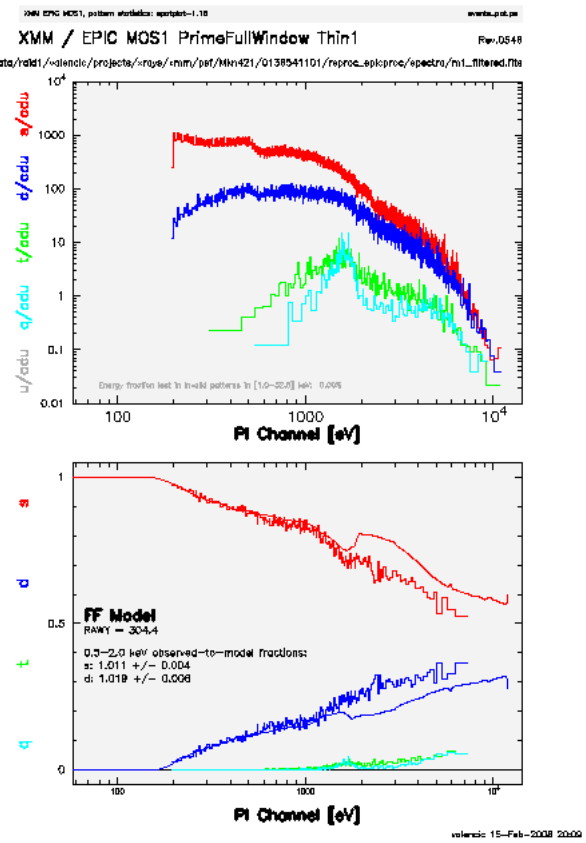


Figure 7.7: The output of *epatplot* for a heavily piled source. In the lower plot, there are large differences between the predicted and observed pattern distribution at energies above $\sim$ 1000 eV.

7.12 Create the Photon Redistribution Matrix (RMF) and Ancillary File (ARF)

The following assumes that an appropriate source spectrum, named `mos1_pi.fits`, has been extracted as in §7.8.

To make the RMF,

1) Invoke the task *rmfgen* in the SAS GUI.
2) In the "Main" tab, set the `spectrumset` keyword to the spectrum file name, e.g., `mos1_pi.fits`. Set the `rmfset` keyword to the RMF file name, e.g., `mos1_rmf.fits`.
3) Click "Run".

To make the ARF,

1) Invoke the task *arfgen* in the SAS GUI.
2) In the "Main" tab, set the `arfset` parameter to the ARF file name, for example, `mos1_arf.fits`. Set the `spectrumset` parameter to the spectrum file name, in this case, `mos1_pi.fits`.
3) In the "Effects" tab, confirm that the `withbadpixcorr` box is checked. Set the `badpixlocation` keyword to the event file name from which the spectrum was extracted, in this case, `mos1_filt_time.fits`.
4) In the "Calibration" tab, check the `withrmfset` box and set the `rmfset` keyword to the RMF file name, in this case, `mos1_rmf.fits`.

5) Click “Run”.

At this point, the spectrum is ready to be analyzed, so skip ahead to prepare the spectrum for fitting §13.

Chapter 8

An EPIC Data Processing and Analysis Primer (Timing Mode, Command Line)

So, you've received an *XMM-Newton* EPIC data set. What are you going to do with it? After checking what the observation consists of (see § 3.2), you should note when the observation was taken. If it is a recent observation, it was likely processed with the most recent calibrations and SAS, and you can immediately start to analyze the Pipeline Processed data. However, if it is more than a year old, it was probably processed with older versions of CCF and SAS prior to archiving, and the pipeline should be rerun to generate event files with the latest calibrations.

As noted in Chapter 4, a variety of analysis packages can be used for the following steps. However, as the SAS was designed for the basic reduction and analysis of *XMM-Newton* data (extraction of spatial, spectral, and temporal data), it will be used here for demonstration purposes. SAS will be required at any rate for the production of detector response files (RMFs and ARFs) and other observatory-specific requirements. (Although for the simple case of on-axis point sources the canned response files provided by the SOC can be used.)

It is **strongly** recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If your data are recent, you need only to gunzip the files and prepare the data for processing (see §5. Feel free to skip the section on repipelining and proceed to the later discussions. In any case, for simplicity, it is recommended that you change the name of the unzipped event file to something easy to type. For example, a PN event list:

```
cp PPS/Piiiiiijjkk PNSlllTIEVLI0000.FTZ PROC/pn.fits
```

where

`iiiiiijjkk` – observation number
`lll` – exposure number within the observation

Various analysis procedures are demonstrated using the Cen X-3 dataset, ObsID 0400550201. The following procedures are applicable to all XMM-Newton datasets, so it is not required that you use this particular dataset; any Timing Mode observation should be sufficient.

For detailed descriptions of PP data nomenclature, file contents, and which tasks can be used to view them, see Tables 3.2 and 3.3. For detailed descriptions of ODF data nomenclature and file contents, see Table 3.1.

8.1 Rerun the Pipeline

We assume that the data was prepared and environment variables were set according to §5. In the window where SAS was initialized, in your "processing directory" PROC, run *epchain* or *epproc* to produce calibrated photon event files for the PN camera.

Note that *epproc* will automatically detect what mode the data were taken in. However, if we want to use *epchain*, we will need to set the relevant parameter, since this dataset was not taken in Imaging Mode; this is shown below. (Please note that SAS is case-sensitive.)

On the command line, just type:

```
epproc
```

or, alternatively,

```
epchain datamode=TIMING
```

If your data is in Burst Mode:

```
epchain datamode=BURST
```

By default, neither of these tasks keep any intermediate files they generate. *Epchain* maintains the naming convention described in §3.3.3. *Epproc* designates its output event files with "Evts.ds", so "*TimingEvts.ds" and "*BurstEvts.ds" denote the timing and burst mode event lists, respectively. In any case, you may want to name the new files something easy to type. For example, to rename the new PN event files output from *epchain* or *epproc*, respectively, type

```
mv P0403530301PNS003TIEVLI0000.FIT pn.fits
```

or

```
mv 1206_0403530301_EPN_S003_TimingEvts.ds pn.fits
```

8.2 Create and Display an Image

To create an image of your timing data, type

```
evselect table=pn.fits withimageset=yes imageset=image.fits \
   xcolumn=RAWX ycolumn=RAWY imagebinning=binSize ximagebinsize=1 yimagebinsize=1
```

where

`table` – input event table
`withimageset` – make an image
`imageset` – name of output image
`xcolumn` – event column for X axis
`ycolumn` – event column for Y axis
`imagebinning` – form of binning, force entire image into a given size or bin by a specified number of pixels
`ximagesize` – output image pixels in X
`yimagesize` – output image pixels in Y

The output file `image.fits` can be viewed by using a standard FITS display, such as *ds9* (see Figure 8.1) :

```
ds9 image.fits &
```

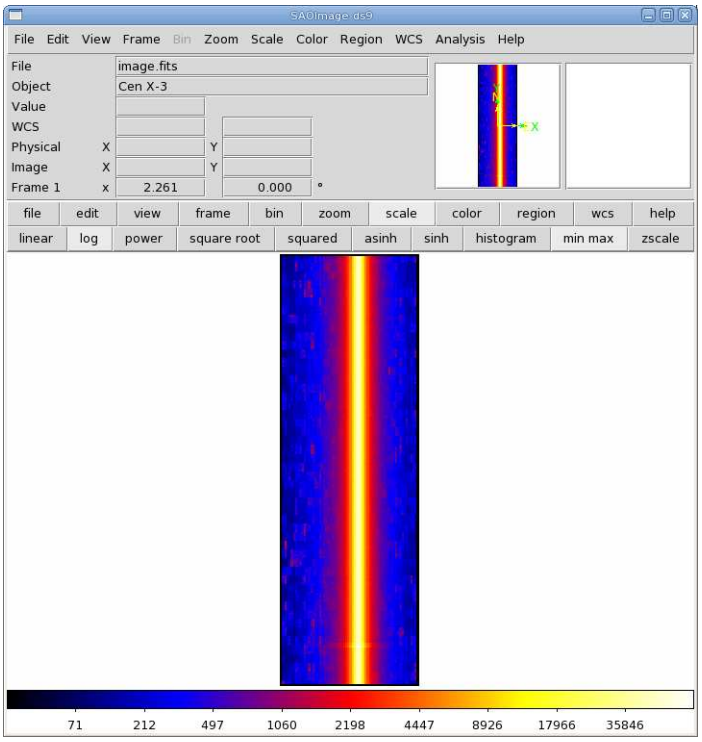


Figure 8.1: The PN Timing image, displayed in *ds9*.

8.3 Applying Standard Filters to the Data

The filtering expression for the PN in Timing mode is:

```
(PATTERN <= 4)&&(PI in [200:15000])&&#XMMEA_EP
```

The first two expressions will select good events with `PATTERN` in the 0 to 4 range. The `PATTERN` value is similar the `GRADE` selection for *ASCA* data, and is related to the number and pattern of the CCD pixels triggered for a given event. Single pixel events have `PATTERN == 0`, while double pixel events have `PATTERN in [1:4]`.

The second keyword in the expressions, `PI`, selects the preferred pulse height of the event; for the PN, this should be between 200 and 15000 eV. This should clean up the image significantly with most of the rest of the obvious contamination due to low pulse height events. Setting the lower PI channel limit somewhat higher (e.g., to 300 or 400 eV) will eliminate much of the rest.

Finally, the `#XMMEA_EP` filter provides a canned screening set of FLAG values for the event. (The FLAG value provides a bit encoding of various event conditions, e.g., near hot pixels or outside of the field of view.) Setting `FLAG == 0` in the selection expression provides the most conservative screening criteria and should always be used when serious spectral analysis is to be done on PN data.

To filter the data, type

```
evselect table=pn.fits withfilteredset=yes \
   expression='(PATTERN <= 4)&&(PI in [200:15000])&&#XMMEA_EP' \
   filteredset=pn_filt.fits filtertype=expression keepfilteroutput=yes \
   updateexposure=yes filterexposure=yes
```

where

`table` – input event table
`filtertype` – method of filtering

`expression` – filtering expression
`withfilteredset` – create a filtered set
`filteredset` – output file name
`keepfilteroutput` – save the filtered output
`updateexposure` – update exposure information in event list and in spectrum files
`filterexposure` – filter exposure extensions of event list with same time

8.4 Create and Display a Light Curve

Sometimes, it is necessary to use filters on time in addition to those mentioned above. This is because of soft proton background flaring, which can have count rates of 100 counts/sec or higher across the entire bandpass.

To determine if our observation is affected by background flaring, we can examine the light curve. For the time binning, we will set it to something reasonable (usually between 10 and 100 s):

```
evselect table=pn.fits withrateset=yes rateset=pn_ltcrv.fits \
   maketimecolumn=yes timecolumn=TIME timebinsize=50 makeratecolumn=yes
```

where

`table` – input event table
`withrateset` – make a light curve
`rateset` – name of output light curve file
`maketimecolumn` – control to create a time column
`timecolumn` – time column label
`timebinsize` – time binning (seconds)
`makeratecolumn` – control to create a count rate column, otherwise a count column will be created

The output file `pn_ltcrv.fits` can be viewed by using *fv*:

```
fv pn_ltcrv.fits &
```

In the *fv* pop-up window, the RATE extension will be available in the second row (index 1, as numbering begins with 0). Select "PLOT" from this row, and select the column name and axis on which to plot it. The light curve is shown in Fig. 8.2. No flares are evident, so we will continue to the next section. However, if a dataset does contain flares, they should be removed in the same way as shown for EPIC Imaging mode data in §6.5.

8.5 Extract the Source and Background Spectra

The first step in extracting a spectrum from PN Timing data is to make an image of the event file over the energy range we are interested in; for this example, we'll say 0.5-15 keV. And since this is the PN, we need to remember to set `(FLAG==0)` to get a high-quality spectrum. Thus, our `expression` parameter would be set to `(FLAG==0) && (PI in [500:15000])`, and the entire command would be:

```
evselect table=pn_filt.fits withimageset=yes imageset=image.fits \
   xcolumn=RAWX ycolumn=RAWY imagebinning=binSize ximagebinsize=1 \
   yimagebinsize=1 expression='(FLAG==0) && (PI in [500:15000])'
```

where the parameters are as described in §8.2.

The image can be displayed with *ds9* and is shown in Fig. 8.3 (top). The source is centered on RAWX=37; we will extract this and the 10 pixels on either side of it:

```
evselect table=pn_filt.fits spectrumset=source_pi_WithBore.fits \
   energycolumn=PI spectralbinsize=5 specchannelmin=0 specchannelmax=20479 \
   filteredset=pn_filt_source_WithBore.fits \
   expression='(FLAG==0) && (PI in [500:15000]) && (RAWX in [27:47])'
```

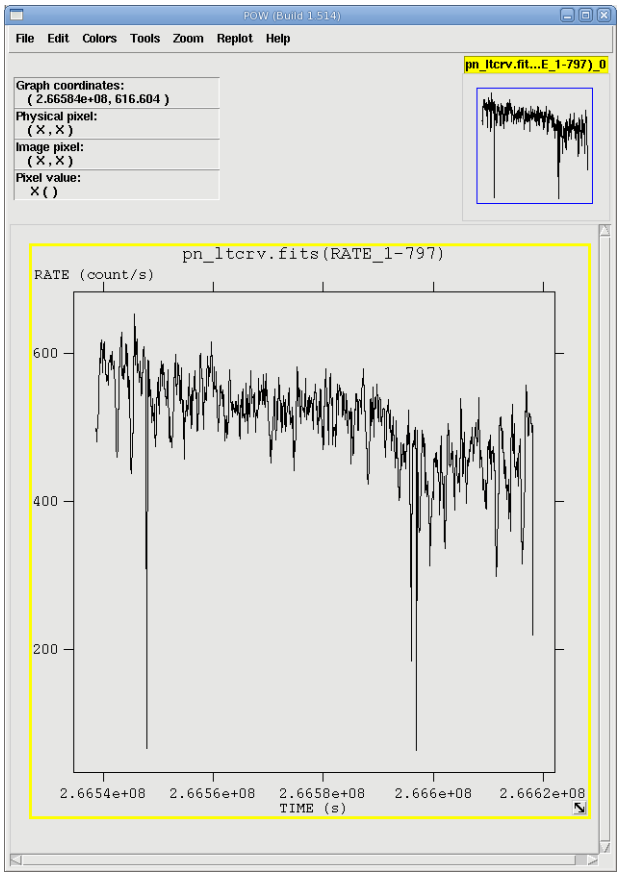


Figure 8.2: The light curve of Cen X-3 from 0.2-15 keV, displayed in *fv*.

where the keywords are the same as in §8.3, and

`energycolumn` – energy column
`spectrumset` – name of output spectrum
`spectralbinsize` – size of bin, in eV
`specchannelmin` – minimum of spectral range
`specchannelmax` – maximum of spectral range

For the background, the extraction area should be as far from the source as possible. However, sources with > 200 ct/s (like our example!) are so bright that they dominate the entire CCD area, and there is no source-free region from which to extract a background. (It goes without saying that this is highly energy-dependent.) In such a case, it may be best not to subtract a background. Users are referred to Ng et al. (2010, A&A, 522, 96) for an in-depth discussion. While this observation is too bright to have a good background extraction region, the process is shown below nonetheless for the sake of demonstration:

```
evselect table=pn_filt.fits withspectrumset=yes spectrumset=bkg_pi.fits \
   energycolumn=PI spectralbinsize=5 withspecranges=yes specchannelmin=0 \
   expression='(FLAG==0) && (PI in [500:15000]) && (RAWX in [3:5])' \
   specchannelmax=20479 withfilteredset=y filteredset=pn_filt_bkg.fits
```

8.6 Check for Pile Up

Depending on how bright the source is and what modes the EPIC detectors are in, event pile up may be a problem. Pile up occurs when a source is so bright that incoming X-rays strike two neighboring pixels or the same

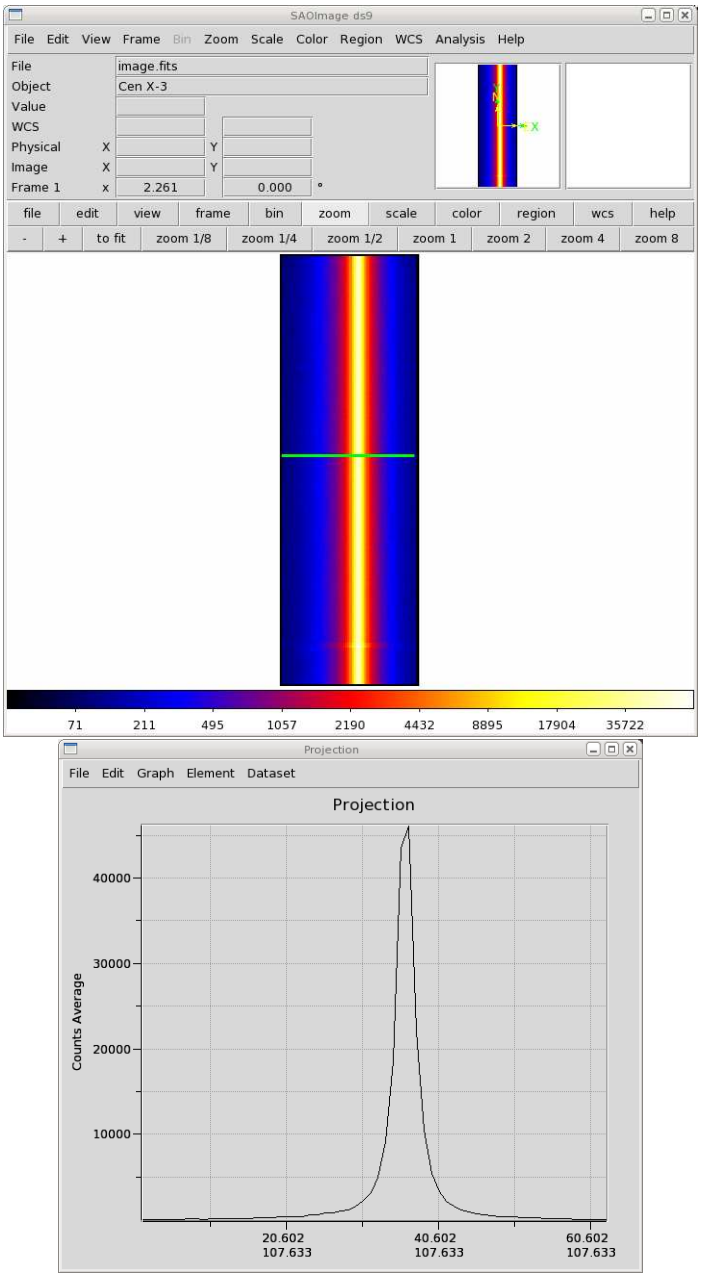


Figure 8.3: TOP: The Cen X-3 Timing Mode image, from 0.5-15 keV. The green line is a projection cut. BOTTOM: The average counts in the cut across the CCD.

pixel in the CCD more than once in a read-out cycle. In such cases the energies of the two events are in effect added together to form one event. If this happens sufficiently often, 1) the spectrum will appear to be harder than it actually is, and 2) the count rate will be underestimated, since multiple events will be undercounted. Briefly, we deal with it in PN Timing data essentially the same way as in Imaging data, that is, by using only single pixel events, and/or removing the regions with very high count rates, checking the amount of pile up, and repeating until it is no longer a problem. We recommend to always check for it.

Note that this procedure requires as input the event files created when the spectrum was made, not the usual time-filtered event file.
To check for pile up:

```
epatplot set=pn_filt_source_WithBore.fits plotfile=pn_epat.ps useplotfile=yes \
```

```
withbackgroundset=yes backgroundset=pn_flt_bkg.fits
```

where

`set` – input events file
`plotfile` – output postscript file
`useplotfile` – flag to use file name from "plotfile"
`withbackgroundset` – use background event set for background subtraction?
`backgroundset` – name of background event file

The output of *epatplot* is a postscript file, `pn_epat.ps`, which may be viewed with viewers such as *gv*, containing two graphs describing the distribution of counts as a function of PI channel; see Figure 8.4.

A few words about interpretting the plots are in order. The top is the distribution of counts versus PI channel for each pattern class (single, double, triple, quadruple), and the bottom is the expected pattern distribution (**smooth lines**) plotted over the observed distribution (**histogram**). The lower plot shows the model distributions for single and double events and the observed distributions. It also gives the ratio of observed-to-modeled events with 1-σ uncertainties for single and double pattern events over a given energy range. (The default is 0.5-2.0 keV; this can be changed with the `pileupnumberenergyrange` parameter.) If the data is not piled up, there will be good agreement between the modeled and observed single and double event pattern distributions. Also, the observed-to-modeled fractions for both singles and doubles in the 0.5-2.0 keV range will be unity, within errors. In contrast, if the data is piled up, there will be clear divergence between the modeled and observed pattern distributions, and the observed-to-modeled fraction for singles will be less than 1.0, and for doubles, it will be greater than 1.0.

Finally, when examining the plots, it should noted that the observed-to-modeled fractions can be inaccurate. Therefore, the agreement between the modeled and observed single and double event pattern distributions should be the main factor in determining if an observation is affected by pile up or not.

Examining the plots, we see that there is a large difference between the modeled and observed single and double pattern events, and that the observed-to-model fraction for doubles is over 1.0, indicating that the observation is piled up.

8.7 My Observation is Piled Up! Now What?

There are a couple ways to deal with pile up. First, you can use event file filtering procedures to include only single pixel events (`PATTERN==0`), as these events are less sensitive to pile up than other patterns.

You can also excise areas of high count rates, i.e., the boresight column and several columns to either side of it. (This is analogous to removing the inner-most regions of a source in Imaging data.) The spectrum can then be re-extracted and you can continue your analysis on the excised event file. As with Imaging data, it is recommended that you take an iterative approach: remove an inner region, extract a spectrum, check with *epatplot*, and repeat, each time removing a slightly larger region, until the model and observed pattern distributions agree.

To extract only the columns to either side of the boresight:

```
evselect table=pn_filt.fits withspectrumset=yes spectrumset=source_pi_NoBore.fits \
   energycolumn=PI spectralbinsize=5 withspecranges=yes specchannelmin=0 \
   specchannelmax=20479 \
   expression='(FLAG ==0)&&(PI in [500:15000])&&(RAWX in [27:47])&&!(RAWX in [29:45]) \
   withfilteredset=yes filteredset=pn_filt_source_NoBore.fits
```

Be aware that if you do this and are using SAS v. 13.x or older, you will need to use a non-standard way to make the ancillary files (ARFs) for your spectrum! This is discussed further in §8.9.

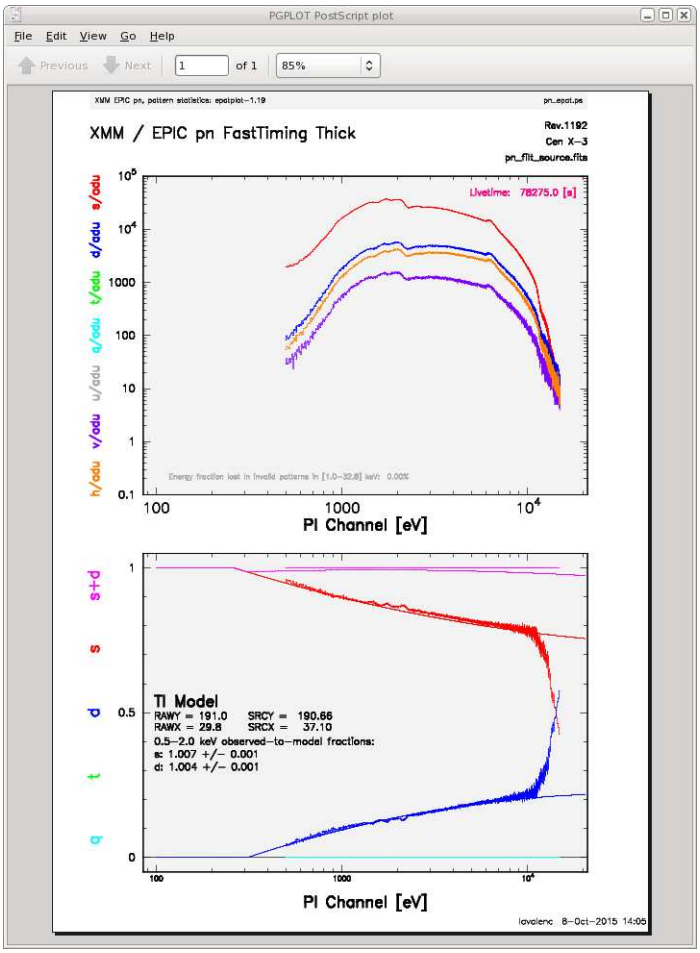


Figure 8.4: The output of *epatplot*.

8.8 Determine the Spectrum Extraction Areas

Now that we are confident that our spectrum is not piled up, we can continue by finding the source and background region areas. (This process is identical to that used for IMAGING data.) This is done with the task *backscale*, which takes into account any bad pixels or chip gaps, and writes the result into the `BACKSCAL` keyword of the spectrum table.

To find the source and background extraction areas:

```
backscale spectrumset=source_pi_NoBore.fits badpixlocation=pn_filt.fits
```

```
backscale spectrumset=bkg_pi.fits badpixlocation=pn_filt.fits
```

where

`spectrumset` – spectrum file
`badpixlocation` – event file containing the bad pixels

8.9 Create the Photon Redistribution Matrix (RMF) and Ancillary File (ARF)

If you are using SAS v. 14 or higher, making the RMF and ARF for PN data in TIMING mode is exactly the same as in IMAGING mode, *even if you had to excise piled up areas.* This is a change from earlier SAS

versions; if you are working with an older SAS, you will need to use the special recipe below to generate the ARF (the method to make a RMF file is the same as shown here.)

To make the RMF:

```
rmfgen rmfset=source_rmf_NoBore.fits spectrumset=source_pi_NoBore.fits
```

where

`rmfset` – output file
`spectrumset` – spectrum file

To make the ARF:

```
arfgen arfset=source_arf_NoBore.fits spectrumset=source_pi_NoBore.fits \
   detmaptype=psf withrmfset=yes rmfset=source_rmf_NoBore.fits badpixlocation=pn_filt.fits
```

where

`arfset` – output file
`spectrumset` – spectrum file
`arfset` – output file
`detmaptype` – origin of the detector map
`withrmfset` – use the RMF dataset to define the ARF energy grid?
`rmfset` – RMF file
`badpixlocation` – the file containing the bad pixel locations

If you excised regions to make a spectrum and are using SAS v. 13.x or older, you will need the spectra of the full extraction area and the excised area. We already have it for the full extraction area, so for the excised area,

```
evselect table=pn_filt.fits withspectrumset=yes spectrumset=source_pi_Excised.fits \
   energycolumn=PI spectralbinsize=5 withspecranges=yes specchannelmin=0 \
   specchannelmax=20479 \
   expression='(FLAG ==0)&&(PI in [500:15000])&&(RAWX in [27:47])&&!(RAWX in [29:45]) \
   withfilteredset=yes filteredset=pn_filt_source_Excised.fits
```

Now we can use the spectra to make the ARFs:

```
arfgen arfset=source_arf_WithBore.fits spectrumset=source_pi_WithBore.fits \
   detmaptype=psf badpixlocation=pn_filt.fits
```

```
arfgen arfset=source_arf_Excised.fits spectrumset=source_pi_Excised.fits \
   detmaptype=psf badpixlocation=pn_filt.fits
```

where the parameters are as previously described. Now we can subtract them:

```
addarf "source_arf_WithBore.fits source_arf_Excised.fits" "1.0 -1.0" \
   source_arf_NoBore.fits
```

At this point, the spectrum is ready to be analyzed, so skip ahead to prepare the spectrum for fitting (§13).

The timing data can also be examined in Xronos; this is discussed in §15.

Chapter 9

An EPIC Data Processing and Analysis Primer (Timing Mode, GUI)

So, you've received an *XMM-Newton* EPIC data set. What are you going to do with it? After checking what the observation consists of (see § 3.2), you should note when the observation was taken. If it is a recent observation, it was likely processed with the most recent calibrations and SAS, and you can immediately start to analyze the Pipeline Processed data. However, if it is more than a year old, it was probably processed with older versions of CCF and SAS prior to archiving, and the pipeline should be rerun to generate event files with the latest calibrations.

As noted in Chapter 4, a variety of analysis packages can be used for the following steps. However, as the SAS was designed for the basic reduction and analysis of *XMM-Newton* data (extraction of spatial, spectral, and temporal data), it will be used here for demonstration purposes. SAS will be required at any rate for the production of detector response files (RMFs and ARFs) and other observatory-specific requirements. (Although for the simple case of on-axis point sources the canned response files provided by the SOC can be used.)

It is **strongly** recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If your data are recent, you need only to gunzip the files and prepare the data for processing (see §5. Feel free to skip the section on repipelining and proceed to the later discussions. In any case, for simplicity, it is recommended that you change the name of the unzipped event file to something easy to type. For example, a PN event list:

```
cp PPS/Piiiiiijjkk PNSlllTIEVLI0000.FTZ PROC/pn.fits
```

where

`iiiiiijjkk` – observation number
`lll` – exposure number within the observation

Various analysis procedures are demonstrated using the Cen X-3 dataset, ObsID 0400550201. The following procedures are applicable to all XMM-Newton datasets, so it is not required that you use this particular dataset; any observation should be sufficient.

For detailed descriptions of PP data nomenclature, file contents, and which tasks can be used to view them, see Tables 3.2 and 3.3. For detailed descriptions of ODF data nomenclature and file contents, see Table 3.1.

9.1 Rerun the Pipeline

We assume that the data was prepared and environment variables were set according to §5, the GUI has been invoked (see §5.3), and we are in our working directory, "PROC".

From the upper window of the GUI, select *epchain* or *epproc*. *Epproc* will automatically detect the science mode of the data, so just click "Run". (It is safe to ignore the warnings.) However, *epchain* does not, so in the "General" tab, toggle the datamode parameter to TIMING and click "Run".

By default, none of these tasks keep any intermediate files they generate. *Epchain* maintains the naming convention described in §3.3.3. *Epproc* designates its output event file with "*TimingEvts.ds". In any case, you may want to name the new file something easy to type:

```
mv 1206_0403530301_EPN_S003_TimingEvts.ds pn.fits
```

or

```
mv P0400550201PNS003TIEVLI0000.FIT pn.fits
```

9.2 Create and Display an Image

Many popular data products, such as images and light curves, are made with the task *xmmselect*. The following sections detail how to make them. For an introduction to *xmmselect* and a discussion on how to load an event file in it, please see §7.2.

Once the event file has been loaded,

1) Check the square boxes to the left of the "RAWX" and "RAWY" entries.
2) Click the "Image" button near the bottom of the page. This brings up the *evselect* GUI (see Figure 7.2).
3) In the "General" tab, in the `imageset` box, enter the name of the output file, in this case, `image.fits`.
4) In the "Image" tab, toggle Binning to binSize, and confirm that `ximagebinsize` and `yimagebinsize` are set to 1.
5) Click the "Run" button on the lower left corner of the *evselect* GUI.

The output file `image.fits` can be viewed by using a standard FITS display such as *ds9* (see Fig. 9.1).

9.3 Applying Standard Filters to the Data

The filtering expressions for PN in Timing Mode is:

```
(PATTERN <= 4)&&(PI in [200:15000])&&#XMMEA_EP
```

The first expression will select good events with `PATTERN` between 0 and 4. The `PATTERN` value is similar the `GRADE` selection for *ASCA* data, and is related to the number and pattern of the CCD pixels triggered for a given event. The `PATTERN` assignments are: single pixel events: `PATTERN == 0`, double pixel events: `PATTERN in [1:4]`, triple and quadruple events: `PATTERN in [5:12]`.

The second keyword in the expression, `PI`, selects the preferred pulse height of the event. Since we are working with PN data, this should be between 200 and 15000 eV. This should clean up the image significantly with most of the rest of the obvious contamination due to low pulse height events. Setting the lower PI channel limit somewhat higher (e.g., to 300 eV) will eliminate much of the rest.

Finally, the `#XMMEA_EP` filter provides a canned screening set of FLAG values for the event. (The FLAG value provides a bit encoding of various event conditions, e.g., near hot pixels or outside of the field of view.) Setting `FLAG == 0` in the selection expression provides the most conservative screening criteria and should always be

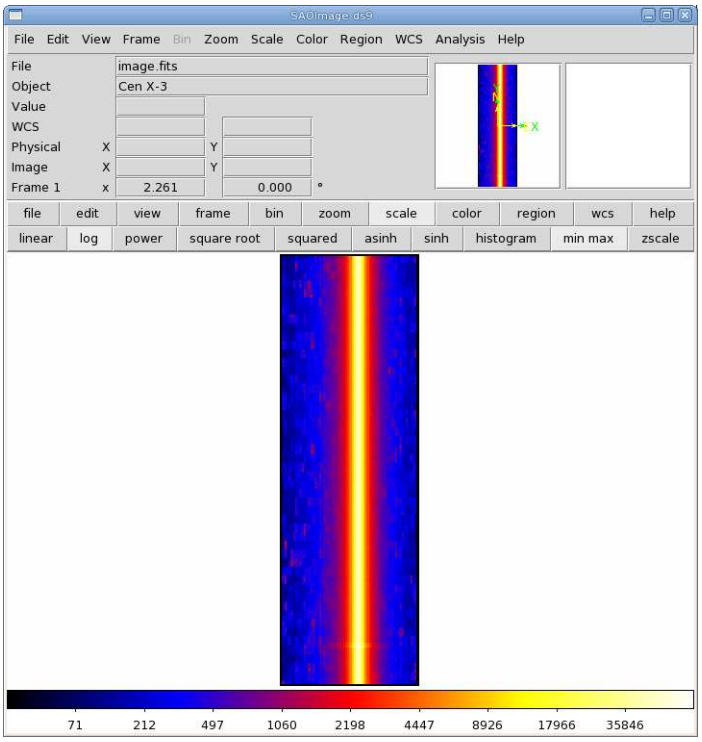


Figure 9.1: An image of PN data taken in Timing mode, displayed in *ds9*.

used when serious spectral analysis is to be done on PN data.

To filter the data using *xmmselect*,

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window:

   `(PATTERN <= 4)&&(PI in [200:15000])&&#XMMEA_EP`

2) Click on the "Filtered Table" box at the lower left of the *xmmselect* GUI.

3) Change the *evselect* `filteredset` parameter, the output file name, to something useful, e.g., `pn_filt.fits`

4) Click "Run".

9.4 Create and Display a Light Curve

Sometimes, it is necessary to use filters on time in addition to those mentioned above. This is because of soft proton background flaring, which can have count rates of 100 counts/sec or higher across the entire bandpass. To determine if our observation is affected by background flaring, we can make a light curve with *xmmselect*. For the time binning, we will set it to something reasonable (usually between 10 and 100 s).

Load the filtered event file, and then

1) Check the round box to the left of the "Time" entry.

2) Click on the "OGIP Rate Curve" button near the bottom of the page. This brings up the *evselect* GUI (see Figure 7.2).

3) Click on the "Lightcurve" tab and change the "timebinsize" to a reasonable amount, e.g., 50. In the "rateset" textbox, enter the name of the output file, `pn_ltcrv.fits`.

4) Click on the "Run" button at the lower left corner of the *evselect* GUI.

The output file `pn_ltcrv.fits` can be viewed by using *fv*:

```
fv pn_ltcrv.fits &
```

In the *fv* pop-up window, the RATE extension will be available in the second row (index 1, as numbering begins with 0). Select "PLOT" from this row, and select the column name and axis on which to plot it. The light curve is shown in Fig. 9.2. No flares are evident, so we will continue to the next section. However, if a dataset does contain flares, they should be removed in the same way as shown for EPIC Imaging mode data in §7.6.

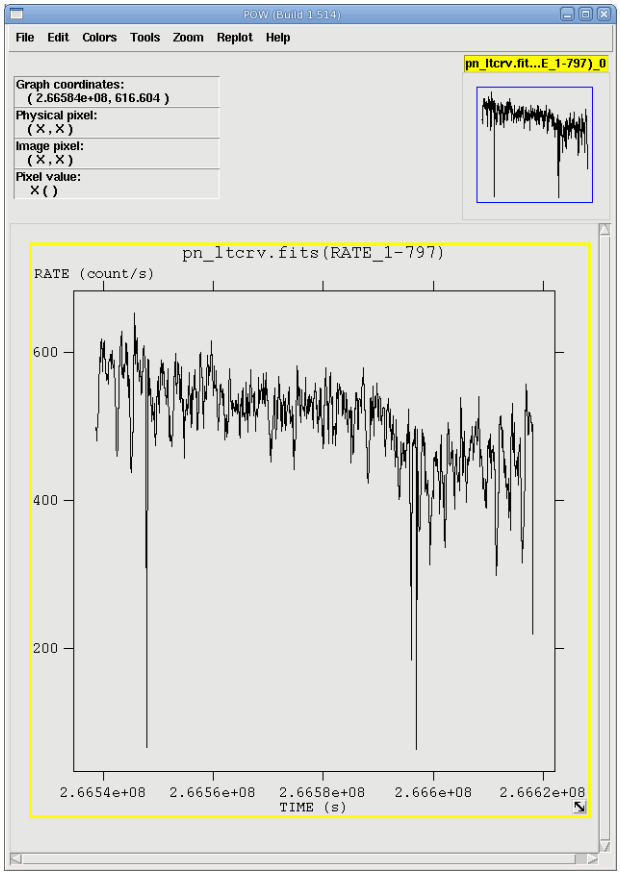


Figure 9.2: The light curve of our example PN Timing mode data, displayed in *fv*.

9.5 Extract the Source and Background Spectra

The first step in extracting a spectrum from PN Timing data is to make an image of the event file over the energy range we are interested in; for this example, we'll say 0.5-15 keV. And since this is the PN, we need to remember to set `(FLAG==0)` to get a high-quality spectrum. Thus, the filtering expression would be set to `(FLAG==0) && (PI in [500:15000])`. So, with `pn_filt.fits` loaded in *xmmselect*,

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window: `(FLAG == 0)&&(PI in [500:15000])`

2) Make an image by checking the square boxes to the left of the "RAWX" and "RAWY" entries to indicate which is on the X and Y axis. Click the "Image" button near the bottom of the page. This brings up the evselect GUI.

3) In the "Image" tab, enter the name of the output file in the imageset box; we will use `pn_image.fits`. Toggle Binning to binSize, and confirm that ximagebinsize and yimagebinsize are set to 1.

4) Click "Run". The image will be displayed automatically in a *ds9* window.

As can be seen in Figure 9.3 (top), the source is centered on RAWX=37. We will extract this and the 10 pixels on either side of it.

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window: `(FLAG==0) && (PI in [500:15000]) && (RAWX in [27:47])`.

2) Click the round button next the `PI` column on the *xmmselect* GUI.

3) Click on "OGIP Spectrum".

4) In the "General" tab, check `keepfilteroutput` and `withfilteredset`. In the `filteredset` box, enter the name of the event file output, in this case, `pn_filt_source_WithBore.fits`.

5) In the "Spectrum" tab, set the file name and binning parameters for the spectrum. Confirm that `withspectrumset` is checked. Set `spectrumset` to the desired output name, `source_pi_WithBore.fits`. Confirm that `withspecranges` is checked. Set `specchannelmin` to 0 and `specchannelmax` to 20479.

6) Click "Run".

For the background, the extraction area should be as far from the source as possible. However, sources with $>$ 200 ct/s (like our example!) are so bright that they dominate the entire CCD area, and there is no source-free region from which to extract a background. (It goes without saying that this is highly energy-dependent.) In such a case, it may be best not to subtract a background. Users are referred to Ng et al. (2010, A&A, 522, 96) for an in-depth discussion. While this observation is too bright to have a good background extraction region, the process is shown below nonetheless for the sake of demonstration:

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window: `(FLAG==0) && (PI in [500:15000]) && (RAWX in [3:5])`.

2) Click the round button next the `PI` column on the *xmmselect* GUI.

3) Click on "OGIP Spectrum".

4) In the "General" tab, check `keepfilteroutput` and `withfilteredset`. In the `filteredset` box, enter the name of the event file output, in this case, `pn_filt_bkg.fits`.

5) In the "Spectrum" tab, set the file name and binning parameters for the spectrum. Confirm that `withspectrumset` is checked. Set `spectrumset` to the desired output name, in this case, `bkg_pi.fits`. Confirm that `withspecranges` is checked. Set `specchannelmin` to 0 and `specchannelmax` to 20479.

6) Click "Run".

9.6 Check for Pile Up

Depending on how bright the source is and what modes the EPIC detectors are in, event pile up may be a problem. Pile up occurs when a source is so bright that incoming X-rays strike two neighboring pixels or the same pixel in the CCD more than once in a read-out cycle. In such cases the energies of the two events are in effect added together to form one event. If this happens sufficiently often, 1) the spectrum will appear to be harder than it actually is, and 2) the count rate will be underestimated, since multiple events will be undercounted. Briefly, we deal with it in PN Timing data essentially the same way as in Imaging data, that is, by using only single pixel events, and/or removing the regions with very high count rates, checking the amount of pile up, and repeating until it is no longer a problem. We recommend to always check for it.

Note that this procedure requires as input the event files created when the spectrum was made, not the usual time-filtered event file.
To check for pile up,

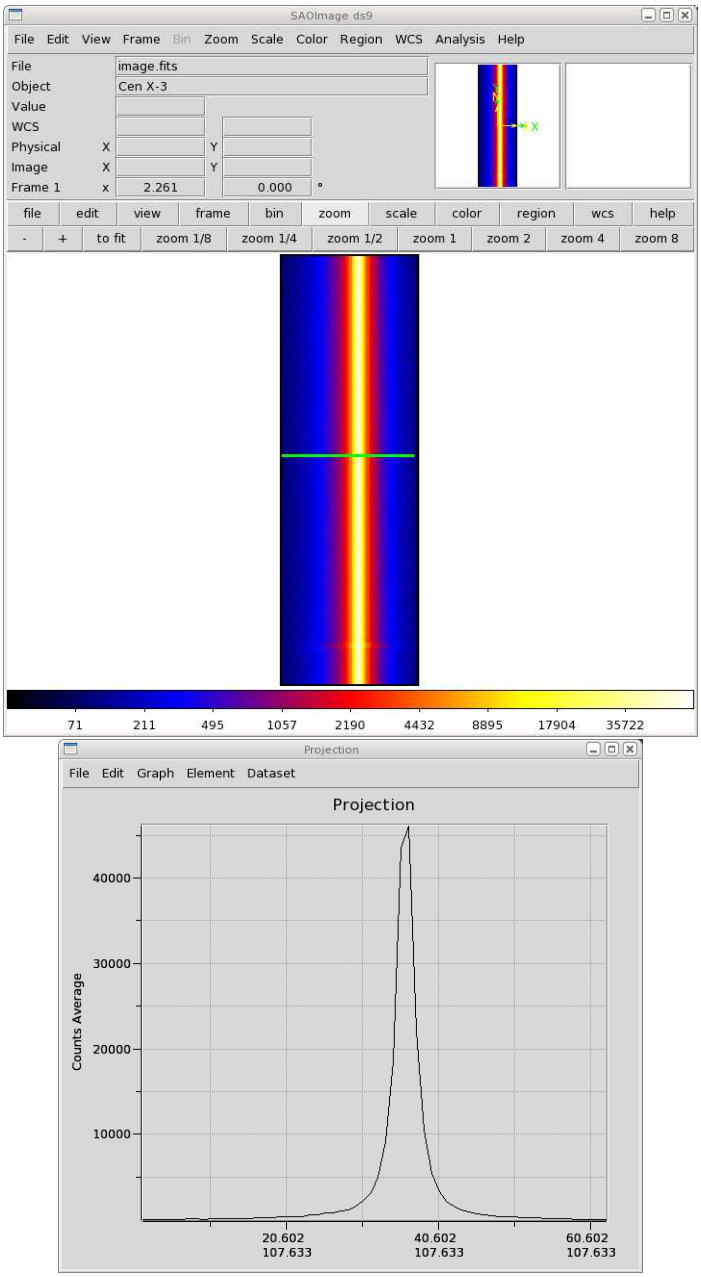


Figure 9.3: TOP: The Cen X-3 Timing Mode image, from 0.5-15 keV. The green line is a projection cut. BOTTOM: The average counts in the cut across the CCD.

1) Invoke *epatplot* in the SAS GUI.

2) In the "0" tab, in the `set` text area, enter the name of the event file that was made when the spectrum was extracted, `pn_filt_source.fits`. If you want to change the output file name to something other than the default, click the `useplotfile` box and enter the name in the `plotfile` text area. For this example, we will use `pn_epat.ps`.

3) In the "1" tab, set `withbackgroundset` to `yes`. In the `backgroundset` text area, enter the name of the background event file output when the background spectrum was extracted, `pn_filt_bkg.fits`.

4) Click "Run".

The output of *epatplot* is a postscript file, `pn_epat.ps`, which may be viewed with viewers such as *gv*, containing

two graphs describing the distribution of counts as a function of PI channel; see Figure 9.4.

A few words about interpretting the plots are in order. The top is the distribution of counts versus PI channel for each pattern class (single, double, triple, quadruple), and the bottom is the expected pattern distribution (**smooth lines**) plotted over the observed distribution (**histogram**). The lower plot shows the model distributions for single and double events and the observed distributions. It also gives the ratio of observed-to-modeled events with 1-σ uncertainties for single and double pattern events over a given energy range. (The default is 0.5-2.0 keV; this can be changed with the `pileupnumberenergyrange` parameter.) If the data is not piled up, there will be good agreement between the modeled and observed single and double event pattern distributions. Also, the observed-to-modeled fractions for both singles and doubles in the 0.5-2.0 keV range will be unity, within errors. In contrast, if the data is piled up, there will be clear divergence between the modeled and observed pattern distributions, and the observed-to-modeled fraction for singles will be less than 1.0, and for doubles, it will be greater than 1.0.

Finally, when examining the plots, it should noted that the observed-to-modeled fractions can be inaccurate. Therefore, the agreement between the modeled and observed single and double event pattern distributions should be the main factor in determining if an observation is affected by pile up or not.

Examining the plots, we see that there is a large difference between the modeled and observed single and double pattern events, and that the observed-to-model fraction for doubles is over 1.0, indicating that the observation is piled up.

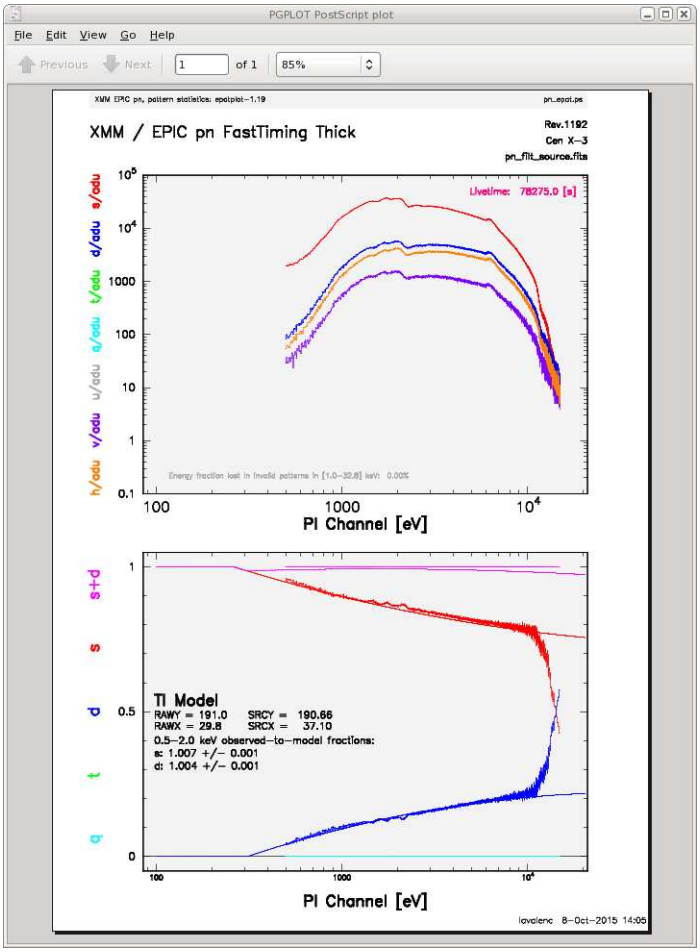


Figure 9.4: The output of *epatplot*.

9.7 My Data is Piled Up! Now What?

There are a couple ways to deal with pile up. First, you can use event file filtering procedures to include only single pixel events (`PATTERN==0`), as these events are less sensitive to pile up than other patterns.

You can also excise areas of high count rates, i.e., the boresight column and several columns to either side of it. (This is analogous to removing the inner-most regions of a source in Imaging data.) The spectrum can then be re-extracted and you can continue your analysis on the excised event file. As with Imaging data, it is recommended that you take an iterative approach: remove an inner region, extract a spectrum, check with *epatplot*, and repeat, each time removing a slightly larger region, until the model and observed pattern distributions agree.

To extract only the columns to either side of the boresight,

1) Enter the filtering criteria in the "Selection Expression" area at the top of the *xmmselect* window: `(FLAG==0) && (PI in [500:15000]) && (RAWX in [27:47]) &&! (RAWX in [29:45])`
2) Click the round button next the PI column on the *xmmselect* GUI.
3) Click on "OGIP Spectrum".
4) In the "General" tab, check `keepfilteroutput` and `withfilteredset`. In the `filteredset` box, enter the name of the event file output. We will use `pn_filt_source_NoBore.fits`.
5) In the "Spectrum" tab, set the file name and binning parameters for the spectrum. Confirm that `withspectrumset` is checked. Set `spectrumset` to the desired output name. We will use `source_pi_NoBore.fits`. Confirm that `withspecranges` is checked. Set `specchannelmin` to 0 and specchannelmax to 20479.
6) Click "Run".

Be aware that if you do this and are using SAS v. 13.x or older, you will need to use a non-standard way to make the ancillary files (ARFs) for your spectrum! This is discussed further in a later section.

9.8 Determine the Spectrum Extraction Areas

Now that we are confident that our spectrum is not piled up, we can continue by finding the source and background region areas. (This process is identical to that used for Imaging data.) This is done with the task *backscale*, which takes into account any bad pixels or chip gaps, and writes the result into the BACKSCAL keyword of the spectrum table. To find the source and background extraction areas, call *backscale* and then

1) In the "Main" tab, enter the name of the spectrum, `source_pi_NoBore.fits`.
2) In the "Effects" tab, confirm that `withbadpixcorr` is checked, and enter the name of the event file, `pn_filt.fits`, in `badpixlocation`.
3) Click "Run".

If you extracted a background spectrum, follow the same steps to find its extraction area, changing the input spectrum file to `bkg_pi.fits`.

9.9 Create the Photon Redistribution Matrix (RMF) and Ancillary File (ARF)

If you are using SAS v. 14 or higher, making the RMF and ARF for PN data in TIMING mode is exactly the same as in IMAGING mode, *even if you had to excise piled up areas.* This is a change from earlier SAS versions; if you are working with an older SAS, you will need to use the special recipe below to generate the ARF (the method to make a RMF file is the same as shown here.)
To make the RMF, call *rmfgen* from the GUI, and then

1) In the "main" tab, set *rmfset* to the output file name. We will use `source_rmf_NoBore.fits`. Set the `spectrumset` parameter to the spectrum, `source_pi_NoBore.fits`.

2) Click "Run".

To make the ARF, call *arfgen* from the GUI, and then

1) In the "main" tab, set `arfset` to the output file name. We will use `source_arf_NoBore.fits`. Set the `spectrumset` parameter to the spectrum, `source_pi_NoBore.fits`.

2) In the "detector map" tab, use the pull-down menu to set the map type to `psf.`

3) In the "calibration" tab, set `withrmfset` to `yes` and rmfset to `source_rmf_NoBore.fits`.

3) In the "effects" tab, verify that `withbadpixcorr` is checked, and set `badpixlocation` to the event file, `pn_filt.fits`.

4) Click "Run".

If you excised regions to make a spectrum and are using SAS v. 13.x or older, you will need to make an ARF for the full extraction area, another one for the piled up area, and then subtract the two to find the ARF for the non-piled regions. To get those, we will need the spectra of the full extraction area and the excised area. We already have it for the full extraction area, so for the excised area, use the same procedure as in §9.5, but change the "Selection Expression" to `(FLAG==0) && (PI in [500:15000]) && (RAWX in [29:45])`, set the `filteredset` parameter to `pn_filt_source_Excised.fits`, and `spectrumset` to `source_pi_Excised.fits`.

Now we can use the spectra to make the ARFs. Invoke *arfgen* from the SAS GUI, and then

1) In the "main" tab, set `arfset` to the output file name. We will use `source_arf_WithBore.fits`. Set the `spectrumset` parameter to the spectrum, `source_pi_WithBore.fits`.

2) In the "detector map" tab, use the pull-down menu to set the map type to `psf.`

3) Click "Run".

Use the same procedure to make the ARF for the excised region, changing the `arfset` parameter to `source_arf_Excised.fits` and `spectrumset` parameter to `source_pi_Excised`.

We can now subtract them. This is easiest to do on the command line:

```
addarf "source_arf_WithBore.fits source_arf_Excised.fits" "1.0 -1.0" source_arf_NoBore.fits
```

At this point, the spectrum is ready to be analyzed, so skip ahead to prepare the spectrum for fitting (§13).

The timing data can also be examined in Xronos; this is discussed in §15.

Chapter 10

An RGS Data Processing and Analysis Primer (Command Line)

Before beginning this chapter please consult the "watchout" page at the SOC:

```
http://xmm.esac.esa.int/sas/current/watchout/
```

This web site discusses current and past SAS bugs and analysis issues.

Many files are associated with an RGS dataset, and it is easy to be overwhelmed. The `INDEX.HTM` file, and links therein, are viewable with a web browser and will help you navigate the dataset. The different types of files are discussed in more detail in Chapter 1.

As ever, it is strongly recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the reprocessing and analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If you have just received your data from the SOC, it has been processed with the most recent version of SAS, and you should not need to repipeline it (though no harm is done if you do); you need only to gunzip the files and prepare the data for processing (see §5). However, it is very likely that you will want to filter your data; in this case, you will need to reprocess it in order to determine the appropriate filters. Therefore, we recommend that you rerun the pipeline regardless of the age of your dataset.

But if you decide that reprocessing is unnecessary, you need only to gunzip the files and rename event files for easier handling. For example, for the RGS1 event list,

```
cp PPS/PiiiiiijjkkR1lEVENLInmmm.FTZ PROC/r1_evt.fits
```

where

`iiiiiijjkk` – observation number
`l` – scheduled (`S`) or unscheduled (`U`) obseravtion
`n` – spectral order number
`mmm` – source number

As noted in Tables 3.2 and 3.3 you can view images of your data. While the zipped FITS files may need to be unzipped before display in *ds9* (depending on the version of *ds9*), they can be displayed when zipped using *fv*. As usual, there are some HTML products to help you inspect the data. These have file names of the form:

```
PPiiiiiijjkkAAAAAA000_0.HTM
```

where

`iiiiiijjkk` – Observation number
`jj` – observation ID - target number in proposal
`kk` – observation ID - observation number for target
`AAAAAA` – Group ID (see Table 3.2)

You will find a variety of RGS-specific files in *XMM-Newton* data sets. Generally there are two of each because there are two RGS instruments. Table 3.3 lists typical file names, their purpose, the file format, and a list of tools that will enable the user to inspect their data. Remember that the `INDEX.HTM` file will help you navigate.

Various analysis procedures are demonstrated using the Mkn 421 dataset, ObsID 0153950701. The following procedures are applicable to all XMM-Newton datasets, so it is not required that you use this particular dataset; any observation should be sufficient.

10.1 Rerun the Pipeline on the Command Line

We assume that the data was prepared and environment variables were set according to §5. In the window where SAS was initialized, in your "processing directory" PROC, run the task *rgsproc*:

```
rgsproc orders='1 2' bkgcorrect=no withmlambdacolumn=yes spectrumbinning=lambda
```

where

`orders` – dispersion orders to extract
`bkgcorrect` – subtract background from source spectra?
`withmlambdacolumn` – include a wavelength column in the event file product
`spectrumbinning` – accumulate the spectrum either in wavelength or beta space

Note the last keyword, `spectrumbinning`. If you want to merge data from the same orders in RGS1 and RGS2, keep it at the default value `lambda`. If you want to merge data from the same instrument, with different orders, set it to `beta`. Merging spectra is discussed in §10.6.

This takes several minutes, and outputs 12 files per RGS, plus 3 general use FITS files. At this point, renaming files to something easy to type is a good idea.

```
ln -s *R1*EVENLI*FIT r1_evt1.fits
ln -s *R2*EVENLI*FIT r2_evt1.fits
```

10.1.1 Potentially useful tips for using the pipeline

The pipeline task, *rgsproc*, is very flexible and can address potential pitfalls for RGS users. In §10.1, we used a simple set of parameters with the task; if this is sufficient for your data (and it should be for most), feel free to skip to later sections, where data filters are discussed. In the following subsections, we will look at the cases of a nearby bright optical source, a nearby bright X-ray source, and a user-defined source.

10.1.2 A Nearby Bright Optical Source

With certain pointing angles, zeroth-order optical light may be reflected off the telescope optics and cast onto the RGS CCD detectors. If this falls on an extraction region, the current energy calibration will require a wavelength-dependent zero-offset. Stray light can be detected on RGS DIAGNOSTIC images taken before, during and after the observation. This test, and the offset correction, are not performed on the data before delivery. To check for stray light and apply the appropriate offsets, type

```
rgsproc orders='1 2' bkgcorrect=no calcoffsets=yes withoffsethistogram=no
```

where the parameters are as described in §10.1 and

`calcoffsets` – calculate PHA offsets from diagnostic images
`withoffsethistogram` – produce a histogram of uncalibrated excess for the user

10.1.3 A Nearby Bright X-ray Source

In the example above, it is assumed that the field around the source contains sky only. Provided a bright background source is well-separated from the target in the cross-dispersion direction, a mask can be created that excludes it from the background region. Here the source has been identified in the EPIC images and its coordinates have been taken from the EPIC source list which is included among the pipeline products. The bright neighboring object is found to be the third source listed in the sources file. The first source is the target:

```
rgsproc orders='1 2' bkgcorrect=no withepicset=yes \
   epicset=Piiiiiijjkkaabl llEMSRLInmmm.FTZ exclsrcsexpr='INDEX==1&&INDEX==3'
```

where the parameters are as described in §10.1 and

`withepicset` – calculate extraction regions for the sources contained in an EPIC source list
`epicset` – name of the EPIC source list, such as generated by *emldetect* or *eboxdetect* procedures
`exclsrcsexpr` – expression to identify which source(s) should be excluded from the background extraction region

10.1.4 User-defined Source Coordinates

If the true coordinates of an object are not included in the EPIC source list or the science proposal, the user can define the coordinates of a new source by typing:

```
rgsproc orders='1 2' bkgcorrect=no withsrc=yes srclabel=Mkn421 srcstyle=radec\
   srcra=166.113808 srcdec=+38.208833
```

where the parameters are as described in §10.1 and

`withsrc` – make the source be user-defined
`srclabel` – source name
`srcstyle` – coordinate system in which the source position is defined
`srcra` – the source's right ascension in decimal degrees
`srcdec` – the source's declination in decimal degrees

Since the event files are current, we can proceed with some simple analysis demonstrations, which will allow us to generate filters. Rememer that all tasks should be called from the window where SAS was initiated, and that tasks place output files in whatever directory you are in when they are called.

10.2 Create and Display an Image

Two commonly-made plots are those showing `PI` vs. `BETA_CORR` (also known as "banana plots") and `XDSP_CORR` vs. `BETA_CORR`.
To create images on the command line, type

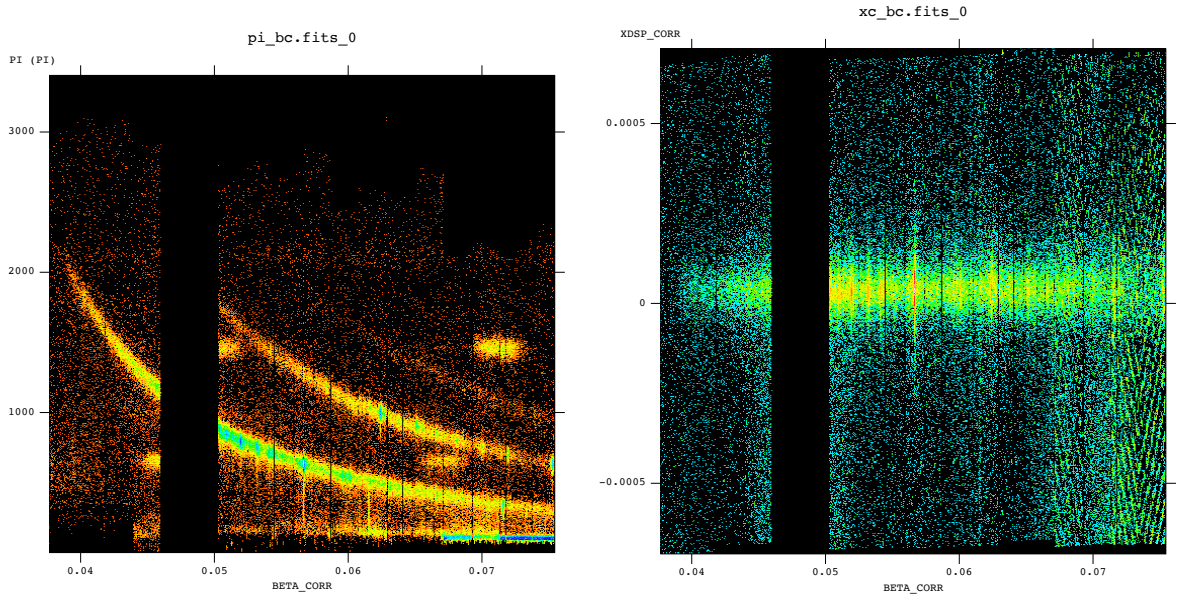
```
evselect table=r1_evt1.fits:EVENTS withimageset=yes \
   imageset=pi_bc.fits xcolumn=BETA_CORR ycolumn=PI \
   imagebinning=imageSize ximagesize=600 yimagesize=600
```

where

`table` – input event table
`withimageset` – make an image
`imageset` – name of output image
`xcolumn` – event column for X axis
`ycolumn` – event column for Y axis
`imagebinning` – form of binning, force entire image into a given size or bin by a specified number of pixels
`ximagesize` – output image pixels in X
`yimagesize` – output image pixels in Y

Plots comparing BETA_CORR to XDSP_CORR may be made in a similar way. The output files can be viewed by using a standard FITS display, such as *ds9* (see Figure 10.1).

Figure 10.1: Plots of PI vs. BETA_CORR (left) and XDSP vs. BETA_CORR (right).



10.3 Create and Display a Light Curve

The background is assessed through examination of the light curve. We will extract a region, CCD9, that is most susceptible to proton events and generally records the least source events due to its location close to the optical axis. Also, to avoid confusing solar flares for source variability, a region filter that removes the source from the final event list should be used. The region filters are kept in the source file product `*SRCLI_*.FIT`.

Please be aware that with SAS 13, the `*SRCLI_*.FIT` file's column information changed. *rgsproc* now outputs an M_LAMBDA column instead of BETA_CORR, and M_LAMBDA should be used to generate the light curve. (The `*SRCLI_*.FIT` file that came with the PPS products still contains a BETA_CORR column if you prefer to use that instead.)

To create a light curve, type

```
evselect table=r1_evt1.fits withrateset=yes rateset=r1_ltcrv.fits \
   maketimecolumn=yes timebinsize=100 makeratecolumn=yes\
   expression= \
   '(CCDNR==9)&&(REGION(P0153950701R1S001SRCLI_0000.FIT:RGS1_BACKGROUND,M_LAMBDA,XDSP_CORR))'
```

where

`table` – input event table
`withrateset` – make a light curve
`rateset` – name of output light curve file
`maketimecolumn` – control to create a time column
`timebinsize` – time binning (seconds)
`makeratecolumn` – control to create a count rate column, otherwise a count column will be created
`expression` – filtering criteria

The output file `r1_ltcrv.fits` can be viewed using *dsplot*:

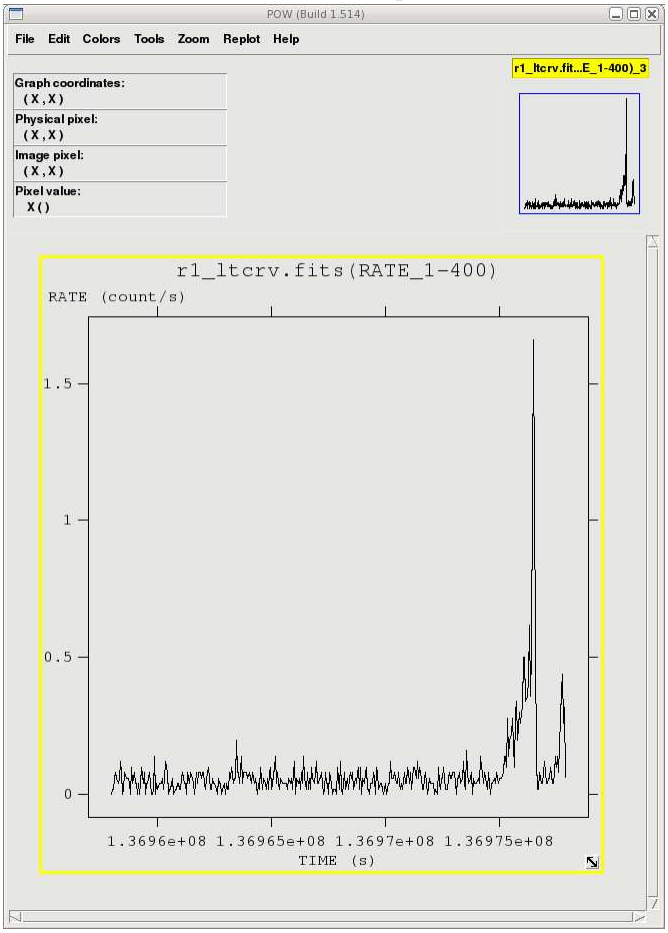
```
dsplot table=r1_ltcrv.fits x=TIME y=RATE &
```

where

`table` – input event table
`x` – column for plotting on the X axis
`y` – column for plotting on the Y axis

The light curve is shown in Figure 10.2.

Figure 10.2: The event rate from the RGS1 CCD9 chip. The time units are elapsed mission time.



10.4 Generating the Good Time Interval (GTI) File

Examination of the lightcurve shows that there is a loud section at the end of the observation, after 1.36975e8 seconds, where the count rate is well above the quiet count rate of $\sim$ 0.05-0.2 count/second. To remove it, we need to make an additional Good Time Interval (GTI) file and apply it by rerunning *rgsproc*.

There are two tasks that make a GTI file: *gtibuild* and *tabgtigen*. Either will produce the needed file, so which one to use is a matter of the user's preference. Both are demonstrated below.

10.4.1 Filter on Time with *gtibuild*

The first method, using *gtibuild*, requires a text file as input. In the first two columns, refer to the start and end times (in seconds) that you are interested in, and in the third column, indicate with either a + or - sign whether

that region should be kept or removed. Each good (or bad) time interval should get its own line. Comments can also be entered, if they are preceeded by a "#". In the example case, we would write in our ASCII file (named `gti.txt`):

```
1.36958e8 1.36975e8 + # The quiet part only, please.
```

and proceed to the SAS task *gtibuild*:

```
gtibuild file=gti.txt table=gti.fits
```

where

`file` – intput text file
`table` – output gti table

10.4.2 Filter on Time with *tabgtigen*

Alternatively, we could filter on time using *tabgtigen* using the filtering expression from the times noted previously:

```
tabgtigen table=r1_ltcrv.fits gtiset=gti.fits timecolumn=TIME \
expression='(TIME in [1.36958e8:1.36975e8])'
```

where

`table` - input data set (the light curve)
`gtiset` - output GTI file name
`timecolumn` - name of column that contains the time stamps
`expression` - filtering expression

10.4.3 Filter on Rate with *tabgtigen*

Finally, we could filter on rate using tabgtigen. The typical quiet count rate for this observation is about 0.05 ct/s, so we would have:

```
tabgtigen table=r1_ltcrv.fits gtiset=gti.fits expression='(RATE<=0.2)'
```

where

`table` – the lightcurve file
`gtiset` – output gti table
`expression` – the filtering criteria. Since the nominal count rate is 0.05 about count/sec, we have set the upper limit to 0.2 count/sec.

10.4.4 Apply the new GTI

Now that we have GTI file, we can apply it to the event file by running *rgsproc* again. *rgsproc* is a complex task, running several steps, with five different entry and exit points. It is not necessary to rerun all the steps in the procedure, only the ones involving filtering.

To apply the GTI to the event file, type

```
rgsproc orders='1 2' auxgtitables=gti.fits bkgcorrect=no \
   withmlambdacolumn=yes entrystage=3:filter finalstage=5:fluxing
```

where

`orders` – spectral orders to be processed
`auxgtitables` – gti file in FITS format
`bkgcorrect` – subtract background from source spectra?
`withmlambdacolumn` – include a wavelength column in the event file product
`entrystage` – stage at which to begin processing
`finalstage` – stage at which to end processing

Since we made a soft link to the event file in §10.1, there is no need to rename it.

10.5 Creating the Response Matrices (RMFs)

Response matrices (RMFs) are now provided as part of the pipeline product package, but you might want create your own. The task *rgsproc* generates a response matrix automatically, but as noted in §10.1.4, the source coordinates are under the observer's control. The source coordinates have a profound influence on the accuracy of the wavelength scale as recorded in the RMF that is produced automatically by *rgsproc*, and each RGS instrument and each order will have its own RMF.

Making the RMF is easily done with the package *rgsrmfgen*. Please note that, unlike with EPIC data, it is not necessary to make ancillary response files (ARFs).
To make the RMFs, type

```
rgsrmfgen spectrumset=P0153950701R1S001SRSPEC1001.FIT rmfset=r1_o1_rmf.fits \
   evlist=r1_evt1.fits emin=0.4 emax=2.5 rows=4000
```

where

`spectrumset` – spectrum file
`evlist` – event file
`emin` – lower energy limit of the response file
`emax` – upper energy limit of the response file
`rows` – number of energy bins; this should be greater than 3000
`rmfset` – output FITS file

RMFs for the RGS1 2nd order, and for the RGS2 1st and 2nd orders, are made in a similar way.
At this point, the spectra can be analyzed or combined with other spectra.

10.6 Combining Spectra

Spectra from the same order in RGS1 and RGS2 can be safely combined to create a spectrum with higher signal-to-noise if they were reprocessed using *rgsproc* with `spectrumbinning=lambda`, as we did in §10.1 (this also happens to be the default). (Spectra of different orders, from one particular instrument, can also be merged if the were reprocessed using *rgsproc* with `spectrumbinning=beta`.) The task *rgscombine* also merges response files and background spectra. When merging response files, be sure that they have the same number of bins. For this example, we assume that RMFs were made for order 1 in both RGS1 and RGS2 with *rgsproc*.

To merge the first order RGS1 and RGS2 spectra, type

```
rgscombine pha='P0153950701R1S001SRSPEC1001.FIT P0153950701R2S002SRSPEC1001.FIT'
   rmf='P0153950701R1S001RSPMAT1001.FIT P0153950701R2S002RSPMAT1001.FIT'
   bkg='P0153950701R1S001BGSPEC1001.FIT P0153950701R2S002BGSPEC1001.FIT'
   filepha='r12_o1_srspec.fits' filermf='r12_o1_rmf.fits'
   filebkg='r12_o1_bgspec.fits' rmfgrid=5000
```

where

`pha` – list of spectrum files
`rmf` – list of response matrices
`bkg` – list of bakcground spectrum files

`filepha` – output merged spectrum
`filermf` – output merged response matrix
`filebkg` – output merged badkground spectrum
`rmfgrid` – number of energy bins; should be the same as the input RMFs

The spectra are ready for analysis, so skip ahead to prepare the spectrum for fitting (§14).

Chapter 11

An RGS Data Processing and Analysis Primer, GUI

Before beginning this chapter please consult the "watchout" page at the SOC:

```
http://xmm.esac.esa.int/sas/current/watchout/
```

This web site discusses current and past SAS bugs and analysis issues.

Many files are associated with an RGS dataset, and it is easy to be overwhelmed. The `INDEX.HTM` file, and links therein, are viewable with a web browser and will help you navigate the dataset. The different types of files are discussed in more detail in Chapter 1.

As ever, it is strongly recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this primer, it is assumed that the Pipleline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, the reprocessing and analysis is taking place in the PROC directory, and the CCF data are in the directory CCF.

If you have just received your data from the SOC, it has been processed with the most recent version of SAS, and you should not need to repipeline it (though no harm is done if you do); you need only to gunzip the files and prepare the data for processing (see §5. However, it is very likely that you will want to filter your data; in this case, you will need to reprocess it in order to determine the appropriate filters. Therefore, we recommend that you rerun the pipeline regardless of the age of your dataset.

But if you decide that reprocessing is unnecessary, you need only to gunzip the files and rename event files for easier handling. For example, for the RGS1 event list,

```
cp PPS/Piiiiiijjkk R1lEVENLInmmm.FTZ PROC/r1_evt.fits
```

where

`iiiiiijjkk` – observation number
`l` – scheduled (`S`) or unscheduled (`U`) obseravtion
`n` – spectral order number
`mmm` – source number

As noted in Tables 3.2 and 3.3 you can view images of your data. While the zipped FITS files may need to be unzipped before display in *ds9* (depending on the version of *ds9*), they can be displayed when zipped using *fv*. As usual, there are some HTML products to help you inspect the data. These have file names of the form:

```
PPiiiiiijjkkAAAAAA000_0.HTM
```

where

`iiiiiijjkk` – Observation number
`jj` – observation ID - target number in proposal
`kk` – observation ID - observation number for target
`AAAAAA` – Group ID (see Table 3.2)

You will find a variety of RGS-specific files in *XMM-Newton* data sets. Generally there are two of each because there are two RGS instruments. Table 3.3 lists typical file names, their purpose, the file format, and a list of tools that will enable the user to inspect their data. Remember that the `INDEX.HTM` file will help you navigate.

Various analysis procedures are demonstrated using the Mkn 421 dataset, ObsID 0153950701, which definitely needs to be repipelined. The following procedures are applicable to all XMM-Newton datasets, so it is not required that you use this particular dataset; any observation should be sufficient.

11.1 Rerun the Pipeline on the Command Line

We assume that the data was prepared and environment variables were set according to §5, the GUI has been invoked (see §5.3), and we are in our working directory, "PROC".

A window will appear with two panels. The upper one will contain task names, what group they belong to (such as utility, epic, timing, calibration, etc.) and a short description of each. The lower one will contain information about environment variables, and as tasks are invoked, feedback from the tasks.

From the upper window, select *rgsproc*. As with all SAS tasks in the GUI, double-clicking the task will bring up pop-up windows that will allow you to change the parameters. For *rgsproc*,

1) In the "global" tab, verify that "orders" is set to `1 2` and "spectrumbinning" is set to `lambda`. In the "angles" tab, verify that "withmlambdacolumn" is set to `yes`. In the "spectra" tab, under the "rgsspectrum" sub-tab, verify that "bkgcorrect" is set to `no`.
2) Click "Run".

Note the last keyword, `spectrumbinning`. If you want to merge data from the same orders in RGS1 and RGS2, keep it at the default value `lambda`. If you want to merge data from the same instrument, with different orders, set it to `beta`. Merging spectra is discussed in §11.7.

This takes several minutes, and outputs 12 files per RGS, plus 3 general use FITS files. At this point, renaming files to something easy to type is a good idea:

```
ln -s *R1*EVENLI*FIT r1_evt1.fits
ln -s *R2*EVENLI*FIT r2_evt1.fits
```

11.1.1 Potentially useful tips for using the pipeline

The pipeline task, *rgsproc*, is very flexible and can address potential pitfalls for RGS users. In §11.1, we used a simple set of parameters with the task; if this is sufficient for your data (and it should be for most), feel free to skip to later sections, where data filters are discussed. In the following subsections, we will look at the cases of a nearby bright optical source, a nearby bright X-ray source, and a user-defined source.

11.1.2 A Nearby Bright Optical Source

With certain pointing angles, zeroth-order optical light may be reflected off the telescope optics and cast onto the RGS CCD detectors. If this falls on an extraction region, the current energy calibration will require a wavelength-dependent zero-offset. Stray light can be detected on RGS DIAGNOSTIC images taken before, during and after the observation. This test, and the offset correction, are not performed on the data before delivery.

To check for stray light and apply the appropriate offsets,

1) Call *rgsproc*.
2) In the "global" tab, verify that "orders" is set to `1 2` and "spectrumbinning" is set to `lambda`. In the "angles" tab, verify that "withmlambdacolumn" is set to `yes`. In the "spectra" tab, under the "rgsspectrum" sub-tab, verify that "bkgcorrect" is set to `no`.
3) In the "events" tab, under the "rgsoffsetcalc" sub-tab, set "calcoffsets" to `yes` and "withoffsethistogram" to `no`.
4) Click "Run".

11.1.3 A Nearby Bright X-ray Source

In the example above, it is assumed that the field around the source contains sky only. Provided a bright background source is well-separated from the target in the cross-dispersion direction, a mask can be created that excludes it from the background region. Here the source has been identified in the EPIC images and its coordinates have been taken from the EPIC source list which is included among the pipeline products. The bright neighboring object is found to be the third source listed in the sources file. The first source is the target:

1) Call *rgsproc*.
2) In the "global" tab, verify that "orders" is set to `1 2` and "spectrumbinning" is set to `lambda`. In the "angles" tab, verify that "withmlambdacolumn" is set to `yes`. In the "spectra" tab, under the "rgsspectrum" sub-tab, verify that "bkgcorrect" is set to `no`.
3) In the "events" tab, under the "rgssources" sub-tab, set "withepicset" to `yes` and "epicset" to the name of the EPIC source list as generated by *emldetect* or *eboxdetect*. In the "spectra" tab, under the "rgsregions" sub-tab, set "exclsrcsexpr" to `INDEX==1&&INDEX==3`.
4) Click "Run".

11.1.4 User-defined Source Coordinates

If the true coordinates of an object are not included in the EPIC source list or the science proposal, the user can define the coordinates of a new source.

1) Call *rgsproc*.
2) In the "global" tab, verify that "orders" is set to `1 2` and "spectrumbinning" is set to `lambda`. In the "angles" tab, verify that "withmlambdacolumn" is set to `yes`. In the "spectra" tab, under the "rgsspectrum" sub-tab, verify that "bkgcorrect" is set to `no`.
3) In the "events" tab, under the "rgssources" sub-tab, set "withsrc" to `yes`. Be sure that the box beneath "withsrc" is toggled to `radec`. Next to "srcra", enter the RA in decimal degrees, `166.113808` and next to "srcdec", enter the declination (again in decimal degrees), `+38.208833`. Next to "srclabel", enter `Mkn421`.
4) Click "Run".

Since the event files are current, we can proceed with some simple analysis demonstrations, which will allow us to generate filters. Remember that all tasks should be called from the window where SAS was initiated, and that tasks place output files in whatever directory you are in when they are called.

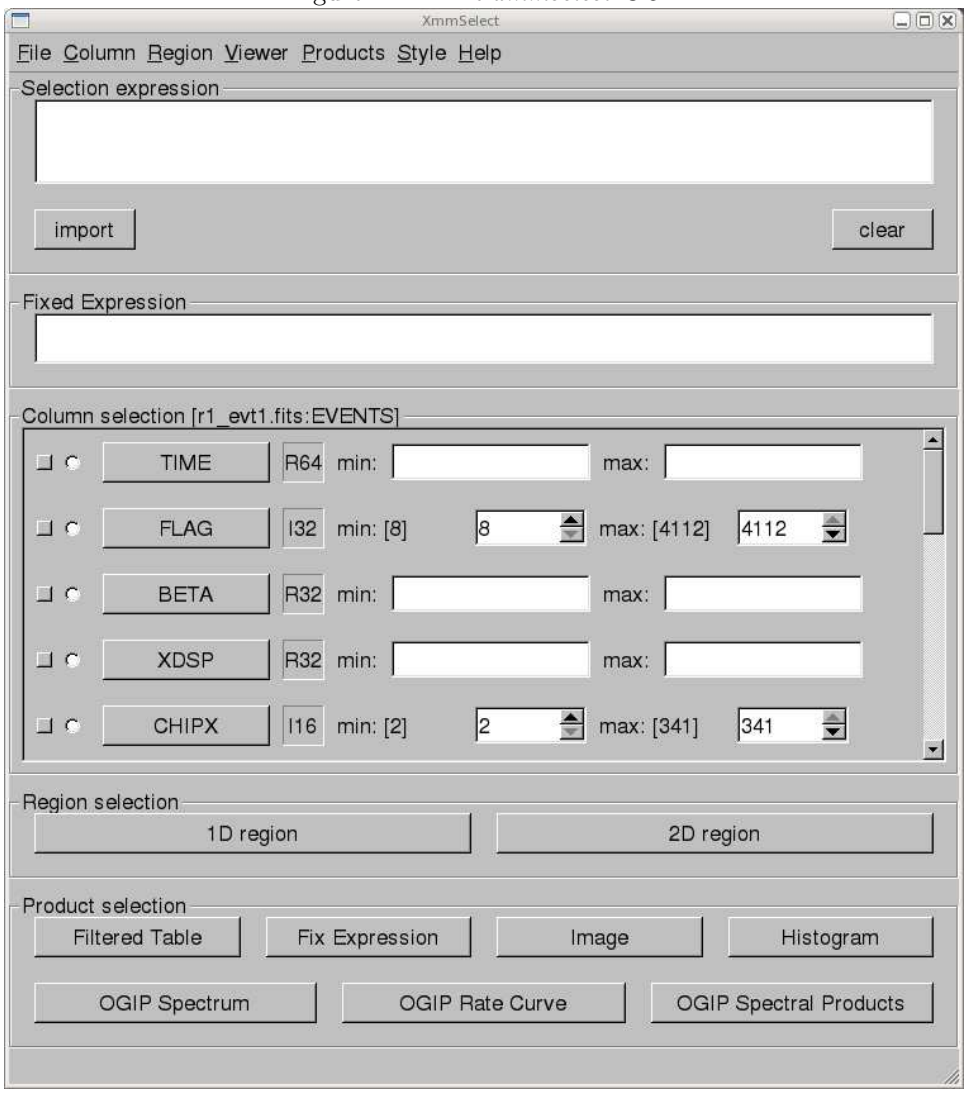
11.2 An Introduction to *xmmselect*

The task *xmmselect* is used for many procedures in the GUI. Like all tasks, it can easily be invoked by starting to type the name and pressing enter when it is highlighted.

When *xmmselect* is invoked a dialog box will first appear requesting a file name. You can either use the browser button or just type the file name in the entry area, "r1_evts1.fits:EVENTS" in this case. To use the browser, select the file folder icon; this will bring up a second window for the file selection. Choose "r1_evts1.fits", then the "EVENTS" extension in the right-hand column, and click "OK". The directory window will then disappear and you can click "Run" on the selection window.

When the file name has been submitted the *xmmselect* GUI (see Figure 11.1) will appear. (Later, when loading an event file that has been filtered, the filters that have been applied to that point will appear in the selection expression area.)

Figure 11.1: The *xmmselect* GUI.



11.3 Create and Display an Image

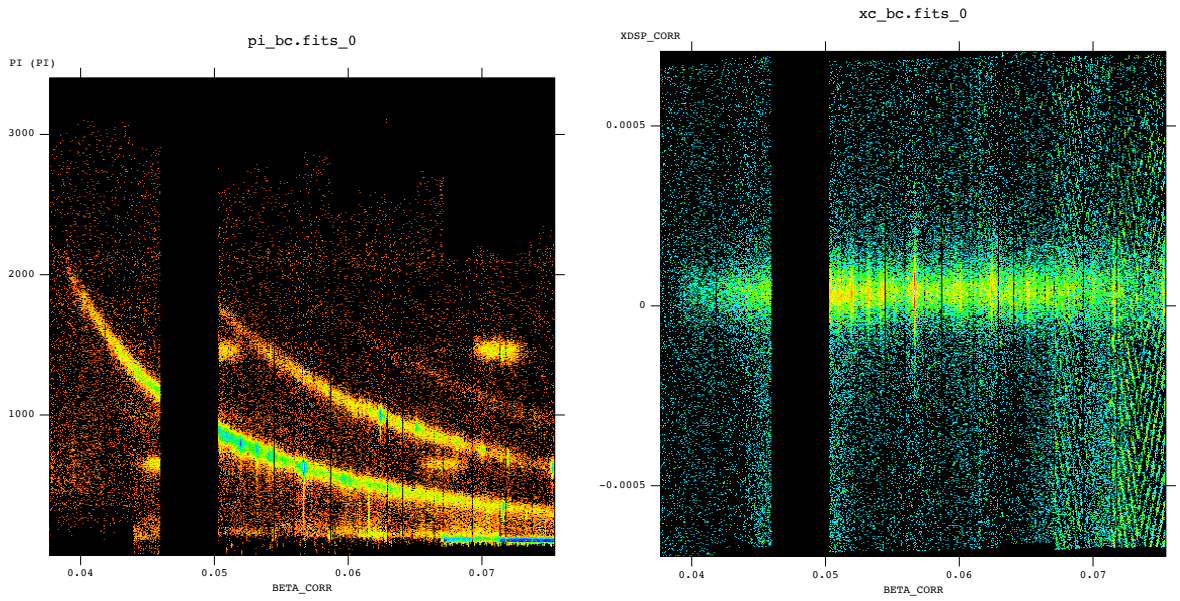
Two commonly-made plots are those showing `PI` vs. `BETA_CORR` (also known as "banana plots") and `XDSP_CORR` vs. `BETA_CORR`. To make a banana plot, invoke *xmmselect* in the SAS GUI and load the event file `r1_evt1.fits`,

as discussed above. Then,

1) Check the square boxes to the left of the `BETA_CORR` and `PI` entries to set the X and Y values in the image.
2) Click on the "Image" button near the bottom of the page. This brings up the `evselect` GUI.
3) Click on the "Image" tab in the `evselect` GUI. In the `imageset` box, enter the name of the output file. We will use `pi_bc.fits`.
4) Click "Run".

Different binnings and other selections can be invoked by accessing the "Image" tab at the top of the GUI. The default settings are reasonable, however, for a basic image. The resultant image is automatically displayed using *ds9*. Similarly, plots can be made comparing `BETA_CORR` to `XDSP_CORR`. These two example plots can be seen in Figure 11.2.

Figure 11.2: Plots of PI vs. BETA_CORR (left) and XDSP vs. BETA_CORR (right).



11.4 Create and Display a Light Curve

The background is assessed through examination of the light curve. We will extract a region, CCD9, that is most susceptible to proton events and generally records the least source events due to its location close to the optical axis. Also, to avoid confusing solar flares for source variability, a region filter that removes the source from the final event list should be used. The region filters are kept in the source file product `P*SRCLI_*.FIT`. (For our example data, this would be `P0134520301R1S001SRCLI_0000.FIT`).

More experienced users should be aware that with SAS 13, the `*SRCLI*` file's **column information changed**. *rgsproc* now outputs an M_LAMBDA column instead of BETA_CORR, and M_LAMBDA should be used to generate the light curve. (The `*SRCLI*` file that came with the PPS products still contains a BETA_CORR column if you prefer to use that instead.)
To create light curves, call *xmmselect* from the GUI. Then,

1) Enter the filtering criteria in the "Selection expression" box at the top of the *xmmselect* GUI:
`(CCDNR==9)&&(REGION(P0153950701R1S001SRCLI_0000.FIT:RGS1_BACKGROUND,M_LAMBDA,XDSP_CORR))`
2) Check the round box to the left of the `time` entry.

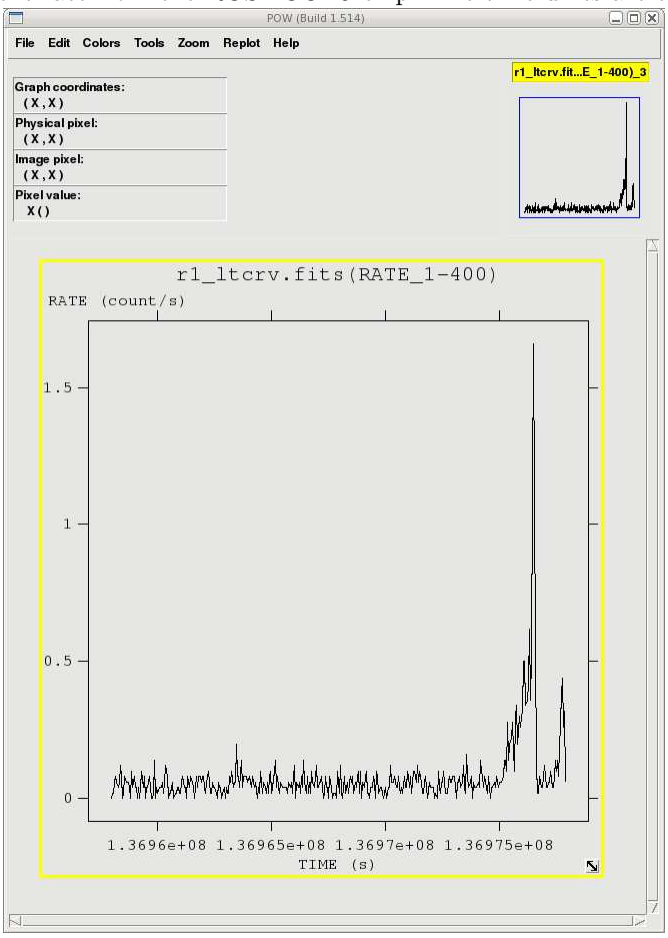
3) Click on the "OGIP Rate Curve" button near the bottom of the page. This brings up the *evselect* GUI.

4) Click on the "Lightcurve" tab and confirm that the `withrateset` box is checked. Change the `timebinsize` to a reasonable amount, e.g. 10 or 100 s, and change the default output file name in the `rateset` box to something appropriate, in this case, `r1_ltcrv.fits`.

5) Click on the "Run" button at the lower left corner of the *evselect* GUI.

The resultant light curve is displayed automatically using Grace and is shown in Figure 11.3.

Figure 11.3: The event rate from the RGS1 CCD9 chip. The time units are elapsed mission time.



11.5 Generating the Good Time Interval (GTI) File

Examination of the lightcurve shows that there is a loud section at the end of the observation, after 1.36975e8 seconds, where the count rate is well above the quiet count rate of $\sim$ 0.05-0.2 count/second. To remove it, we need to make an additional Good Time Interval (GTI) file and apply it by rerunning *rgsproc*.

There are two tasks that make a GTI file: *gtibuild* and *tabgtigen*. Either will produce the needed file, so which one to use is a matter of the user's preference. Both are demonstrated below.

11.5.1 Filter on Time with *gtibuild*

The first method, using *gtibuild*, requires a text file as input. In the first two columns, refer to the start and end times (in seconds) that you are interested in, and in the third column, indicate with either a + or - sign whether

that region should be kept or removed. Each good (or bad) time interval should get its own line. Comments can also be entered, if they are preceeded by a "#". In the example case, we would write in our ASCII file (named `gti.txt`):

```
1.36958e8 1.36975e8 + # The quiet part only, please.
```

and proceed to invoke the task *gtibuild* from the SAS GUI. Then,

1) For the `file` parameter, enter the name of the text file, `gti.txt`. For the `table` parameter, enter the output file name. We will use `gti.fits`.
2) Click "Run".

11.5.2 Filter on Time with *tabgtigen*

Alternatively, we could filter on time using *tabgtigen* using the filtering expression from the times noted previously. To do this, invoke *tabgtigen* from the SAS GUI, and then

1) In the `table` box, enter the name of the lightcurve file, `r1_ltcrv.fits`. In the `gtiset` box, enter the name of the output file; we will use `gti.fits`. In the `expression` box, enter the filtering expression: `(TIME in [1.36958e8:1.36975e8])`
2) Click "Run".

11.5.3 Filter on Rate with *tabgtigen*

Finally, we could filter on rate using *tabgtigen*. The quiet count rate for this observation is about 0.1-0.2 ct/s, so we will use that in the filtering expression. Invoke *tabgtigen* from the SAS GUI, and then

1) In the `table` box, enter the name of the lightcurve file, `r1_ltcrv.fits`. In the `gtiset` box, enter the name of the output file; we will use `gti.fits`. In the `expression` box, enter the filtering expression. We will set the upper limit to 0.2 ct/s: `RATE < 0.2`
2) Click "Run".

11.5.4 Applying the GTI

Now that we have GTI file, we can apply it to the event file by running *rgsproc* again. *rgsproc* is a complex task, running several steps, with five different entry and exit points. It is not necessary to rerun all the steps in the procudure, only the ones involving filtering. Call *rgsproc* from the SAS GUI, then

1) In the "global" tab, set `entrystage` to `3:filter` and `finalstage` to `5:fluxing`.
2) In the "filter" tab, set `auxgtitables` to the new gti file, `gti.fits`.
3) Click "Run".

Since we made soft links to the event files, we don't need to rename them again.

11.6 Creating the Response Matrices (RMFs)

Response matrices (RMFs) are now provided as part of the pipeline product package, but you might want create your own. The task *rgsproc* generates a response matrix automatically, but as noted in §11.1.4, the source coordinates are under the observer's control. The source coordinates have a profound influence on the accuracy of the wavelength scale as recorded in the RMF that is produced automatically by *rgsproc*, and each RGS instrument and each order will have its own RMF.

Making the RMF is easily done with the package *rgsrmfgen*. Please note that, unlike with EPIC data, it is not necessary to make ancillary response files (ARFS).
To make the RMFs, call *rgsrmfgen* from the GUI, then

1) In the `spectrumset` box, enter the name of the spectrum file; for RGS1, order 1, this would be `P0153950701R1S001SRSPEC1001.FIT`. In the `evlist` box, enter the name of the event list, `r1_evt2.fits`. Set `emin` to `0.4` and `emax` to `2.5`. Set `rmfset` to the output file name. We will use `r1_o1_rmf.fits`.

2) Click "Run".

RMFs for the RGS1 2nd order, and for the RGS2 1st and 2nd orders, are made in a similar way.
At this point, the spectra can be analyzed or combined with other spectra.

11.7 Combining Spectra

Spectra from the same order in RGS1 and RGS2 can be safely combined to create a spectrum with higher signal-to-noise if they were reprocessed using *rgsproc* with `spectrumbinning=lambda`, as we did in §11.1 (this also happens to be the default). (Spectra of different orders, from one particular instrument, can also be merged if the were reprocessed using *rgsproc* with `spectrumbinning=beta`.) The task *rgscombine* also merges response files and background spectra. When merging response files, be sure that they have the same number of bins. For this example, we assume that RMFs were made for order 1 in both RGS1 and RGS2 with *rgsproc*.

To merge RGS1 and RGS2 spectra, call *rgscombine* from the SAS GUI, and then

1) Call the *rgscombine* task.

2) In the `pha` box, enter the spectrum files to combine, seperated by a space:
`P0153950701R1S001SRSPEC1001.FIT P0153950701R2S002SRSPEC1001.FIT`. In the `bkg` box, enter the corresponding background files:
`P0153950701R1S001BGSPEC1001.FIT P0153950701R2S002BGSPEC1001.FIT` In the `rmf` box, enter the response files:
`P0153950701R1S001RSPMAT1001.FIT P0153950701R2S002RSPMAT1001.FIT`

3) In the `filepha` box, enter the output merged spectrum, `r12_o1_srspec.fits`. In the `filermf` box, enter the output merged response files, `r12_o1_rmf.fits`. In the `filebkg` box, enter the output merged background spectrum, `r12_o1_bgspec.fits`.

4) Confirm that the `rmfgrid` parameter is set to the same value as that used to make the RMFs, `4000`.

5) Click "Run".

The spectra are ready for analysis, so we can now prepare the spectrum for fitting (§14).

Chapter 12

An OM Data Processing and Analysis Primer

As with EPIC and RGS datasets, many files are associated with an OM dataset. The `INDEX.HTM` file, and links therein, are viewable with a web browser and will help you navigate the dataset. The different types of files are discussed in Chapter 3.2; however, since the OM is somewhat different from the other instruments on-board XMM-Newton, we will discuss them in more detail in later sections.

The OM can operate in Imaging, Fast, and Grism mode. Each of these modes has dedicated commands to reprocess the data: *omichain*, *omfchain*, and *omgchain*. These are metatasks that each call several procedures that are used to prepare the data for processing, make and apply flatfield images, and detect sources. The tasks *omichain* and *omfchain* also calculate the instrumental magnitudes of sources, find the position of the sources (in equatorial coordinates), and produce a sky image; *omgchain* produces a spectrum. If you run these chains, it is helpful to inspect the `sas_log` file to get a detailed list of the performed tasks. These chains rely on filters specified by the user; if no arguments are given, they run on all the files present in the ODF directory. **Due to the long file names and the large number of input parameters, users are urged to simply use the chains and not run the chains' individual tasks one at a time.**

Most OM data are obtained in Imaging mode. If they were obtained in the Fast mode, there will be an additional event list file corresponding to the Fast window (`*FAE.FIT`). Reprocessing of data taken in Fast mode using the command line and SAS GUI is discussed in §12.3. Reprocessing OM Grism data is discussed in §12.4.

As always, it is **strongly** recommended that you keep all reprocessed data in its own directory! SAS places output files in whichever directory it is in when a task is called. Throughout this Guide, it is assumed that the Pipeline Processed data are in the PPS directory, the ODF data (with upper case file names, and uncompressed) are in the directory ODF, and the analysis is taking place in the PROC directory. It is also assumed that the data were prepared for processing (see §5). For example data, we will use observations of the Lockman Hole (Obs ID 0123700101; the same as for the EPIC walk-through) for Image mode, Mkn 421 (Obs ID 0411081601) for Fast mode, and BPM 16274 (Obs ID 0125320801) for Grism mode, though any dataset with the appropriate mode will suffice.

12.1 OM Artifacts and General Information

Before proceeding with the pipeline, it is appropriate to discuss the artifacts that often affect OM images. These can affect the accuracy of a measurement by, for example, increasing the background level. Some of these can be seen in Fig. 12.1.

- Stray light – background celestial light is reflected by the OM detector housing onto the center on the OM field of view, producing a circular area of high background. This can also produce looping structures and long streaks.
- Modulo 8 noise – In the raw images, a modulo 8 pattern arises from imperfections in the event centroiding algorithm in the OM electronics. This is removed during image processing.
- Smoke rings – light from bright sources is reflected from the entrance window back on the detector, producing faint rings located radially away from the center of the field of view.

- Out-of-time events – sources with count rates of several tens of counts/sec show a strip of events along the readout direction, corresponding to photons that arrived while the detector was being read out.

Further, artifacts also can contaminate grism data. Due to this mode's complexity, users are urged to be very careful when working with grism data, and should refer to the SOC's website on this topic.

Users should also keep in mind some differences between OM data and X-ray data. Unlike EPIC and RGS, there are no good time intervals (GTIs) in OM data; an entire exposure is either kept or rejected. Also, OM exposures only provide direct energy information when in grism mode, and the flat field response of the detector is assumed to be unity.

For detailed descriptions of PP data nomenclature, file contents, and which tasks can be used to view them, see Tables 3.2 and 3.3.

If you simply want a quick look at your data, sky images and source lists are in `*SIMAGE*.FTZ` and `*SWSRLI*.FTZ`, respectively. Further, there are low resolution sky images for each filter; they follow the nomenclature:

`PjjjjjjkkkkOMX000RSIMAGbb000.QQQ`

`jjjjjj` – Proposal number
`kkkk` – Observation ID
`b` – Filter keyword: B, V, U, M (UVM2), L (UVW1) and S (UVW2)
`QQQ` – File type (e.g., PNG, FTZ)

So for example, `P0123700101OMX000RSIMAGV000.FTZ` is the final low resolution sky image in the V filter. To see what files have been summed to make the final image, search for the keyword `XPROC0` in the FITS header. For our example image, this would be

```
XPROC0  = 'ommosaic imagesets=''product/P0123700101OMS004SIMAGE1000.FIT produc&'
CONTINUE  't/P0123700101OMS415SIMAGE1000.FIT product/P0123700101OMS416SIMAGE10&'
CONTINUE  '00.FIT product/P0123700101OMS417SIMAGE1000.FIT product/P0123700101O&'
CONTINUE  'MS418SIMAGE1000.FIT'' mosaicedset=product/P0123700101OMX000RSIMAGV0&'
CONTINUE  '00.FIT exposuremap=no exposure=1000 # (ommosaic-1.11.7) [xmmsas_200&'
CONTINUE  '61026_1802-6.6.0]'
```

The source list file (`*SWSRLI*.FTZ`) also contains useful information for the user; the column names are listed in Table 12.1.

Table 12.1: Some of the important columns in the source list file.

Column name	Contents
SRCNUM	Source number
RA	RA of the detected source
DEC	Dec of the detected source
POSERR	Positional uncertainty
RATE	extracted count rate
RATE_ERR	error estimate on the count rate
SIGNIFICANCE	Significance of the detection (in σ)
MAG	Brightness of the source in magnitude
MAGERR	uncertainty on the magnitude

12.2 Imaging Mode

12.2.1 Rerunning the Pipeline

Please note that calling any of the repipelining tasks will initiate processing on all OM data of that particular mode; currently, only *omichain* will accept parameters to limit processing to a specific filter or exposure.

To rerun the pipeline on all exposures and filters from the command line, type

```
omichain
```

Alternatively, to run the pipeline from the SAS GUI,

1) Call *omichain*.
2) In the task pop-up window, click "Run".

This produces numerous files, including images and regions for each exposure and each filter. If we are interested in the sources detected in the mosaicked V band image, we could run *omichain* with the appropriate flags from the command line typing:

```
omichain filters=V processmosaicedimages=yes omdetectnsigma=2.0
   omdetectminsignificance=3.0
```

where

`filters` – list of filters to be processed
`processmosaicedimages` – process the mosaicked sky images?
`omdetectnsigma` – number of σ above background required for a pixel to be considered part of a source
`omdetectminsignificance` – minimum significance of a source to be included in the source list file

Alternatively, we can do it with the SAS GUI:

1) Call *omichain*.
2) In the pop-up window, in the "0" tab, next to "filters", enter `V`; next to "omdetectnsigma", enter `2.0`; next to "omdetectminsignificance", enter `3.0`. In the "1" tab, next to "processmosaicedimages", enter `yes`.
3) Click "Run".

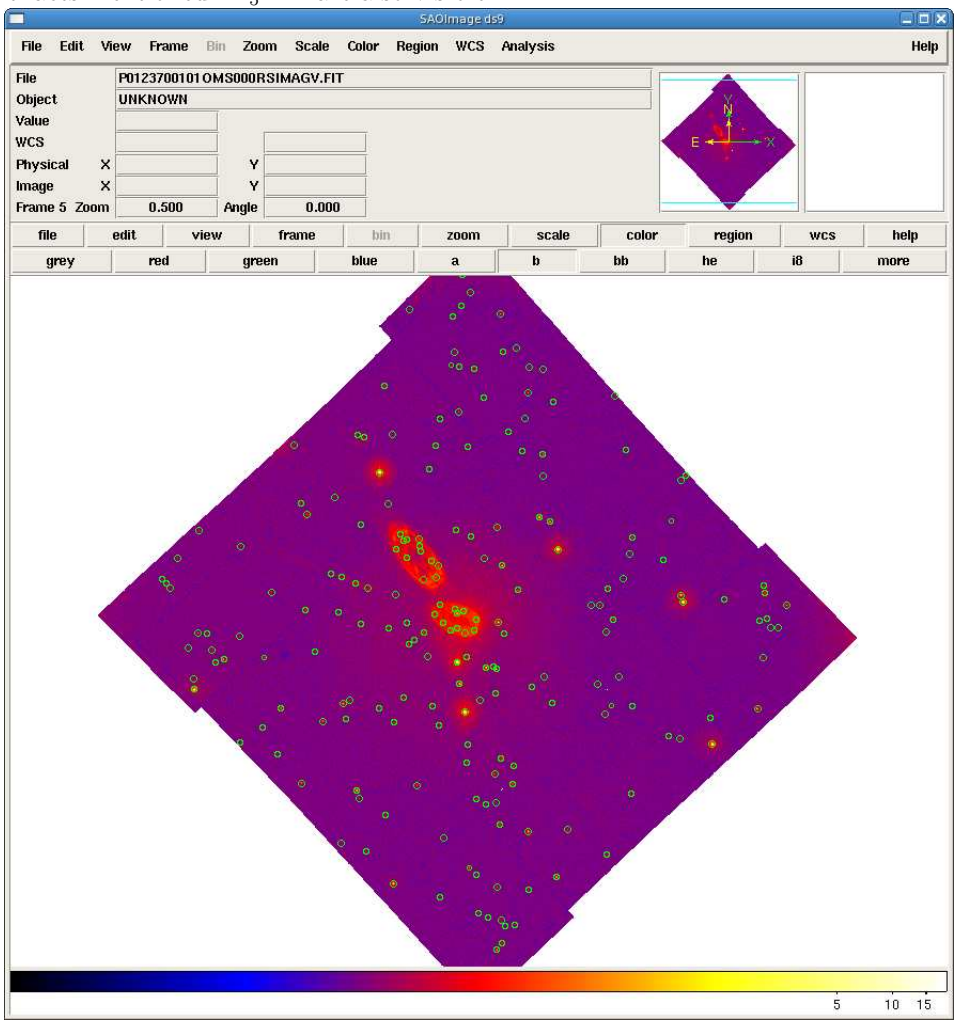
The output files can be used immediately for analysis, though users are strongly urged to examine the output for consistancy first (see §12.2.2). The chains apply all necessary corrections, so no further processing or filtering needs to be done. Please note that the chains do not produce output files with exactly the same names as those in the PPS directory (they also produce some files which are not included in the PPS directory at all.)

12.2.2 Verifying the Output

While the output from the chains is ready for analysis, OM does have some peculiarities, as discussed in §12.1. While these usually have only an aesthetic effect, they can also affect source brightness measurements, since they increase the background. **In light of this, users are strongly encouraged to verify the consistency of the data prior to analysis.** There are a few ways to do this. Users can examine the combined source list with *fv*, which will let them see if interesting sources have been detected in all the filters where they are visible. Users can also overlay the image source list on to the sky image with *implot* by downloading the relevant files to the user's local machine. This can also be done with *ds9* or *gaia* by using *slconv* to change source lists into region files and downloading the relevant files to your local machine. The task *slconv* allows users to set the regions radii in arcseconds to a constant value or scale them to header keywords, such as RATE. By default, *ds9* region files have suffixes of `.reg`; *gaia* region files have suffixes of `.gaia`. In the example below, we make a region file from the source list for the mosaicked, V-band sky image.

To make a *ds9* region file from a source list from the command line, type:

Figure 12.1: A mosaicked, V-band sky image and corresponding region file viewed with *ds9*. Spurious detections are clearly present and can be removed either by hand or by rerunning omichain with the appropriate flags. Some of the artifacts mentioned in §12.1 are also visible.




```
slconv srclisttab=P0123700101OMS000RSISWSV.FIT radiusexpression=5
   outfileprefix=Vband_mosaic outputstyle=ds9
```

where

`srclisttab` – source list file name
`radiusexpression` – constant or expression (possibly involving keywords) used to determine the radii of the plotted circles
`outputstyle` – output format; either ds9 or gaia
`outfileprefix` – prefix of output file name

The mosaicked, V-band sky image, `P0123700101OMS000RSIMAGV.FIT`, with the region file from *slconv* overlayed, is shown in Fig. 12.1. There clearly are spurious detections; these can be removed by hand, or by rerunning *omichain* with a different background-level threshold or source significance.

12.3 Fast Mode

12.3.1 Rerunning the Pipeline

The repipelining task for OM data taken in fast mode is *omfchain*. It produces images of the detected sources, extracts events related to the sources and the background, and extracts the corresponding light curves. At present, unlike *omichain*, *omfchain* does not allow for keywords to specify filters or exposures; calling this task will process all fast mode data.

To run the pipeline on fast mode data on the command line, type

```
omfchain
```

Alternatively, *omfchain* can be run from the GUI:

1) Call *omfchain*.

2) In the task pop-up window, click "Run".

There are two types of output files: those that start with `f` are intermediate images or time series files; those that start with `p` are products.
To demonstrate some of these output files, we have rerun the pipeline on the example dataset. The processed image in sky-coordinates from one exposure, `P0411081601OMS006SIMAGE1000.FIT`, is shown in Fig. 12.2 (left). The background-subtracted light curve produced automatically by the task, `F0411081601OMS006TIMESR1000.PS`, is shown in Fig. 12.2 (right).

The background light curve in Fig. 12.2 (right) is constant because *omfchain* runs with the parameter *bkgfromimage=yes* by default, so that the background light curve is found by using the imaging-mode data, instead of the fast-mode window. This is preferable for even only moderately bright sources (count rate > 0.6 ct/s), as the fast-mode window is small and any background measurement that uses it will likely be contaminated with source photons. This is less of a concern if your source is faint, in which case the background can by found from data in the fast-mode window by typing:

```
omfchain bkgfromimage=no
```

12.3.2 Verifying the Output

A good first step in checking the output is to examine the light curve plot for both the source and background, making sure they are reasonable: no isolated, unusually high (or low) values, and no frequent drop-outs. Users should also check the image with *fv* or *ds9* in the Fast mode window to see if the source is near an edge. If it is, it's a good idea to examine the light curves from diffent exposures to verify that they are consistent from exposure to exposure (while keeping in mind any intrinsic source variability). If the image is blurred or unusual in any way, users should check the tracking history file to verify the tracking was reliable.

12.4 Grism Analysis

12.4.1 Rerunning the Pipeline

The repipelining task for OM data taken in grism mode is *omgchain*. It produces images of the detected sources and background, extracts source spectra and region files, and makes source lists and postscript and PDF plots. At present, unlike *omichain*, *omgchain* does not allow for keywords to specify filters or exposures; calling this task will process all grism mode data.

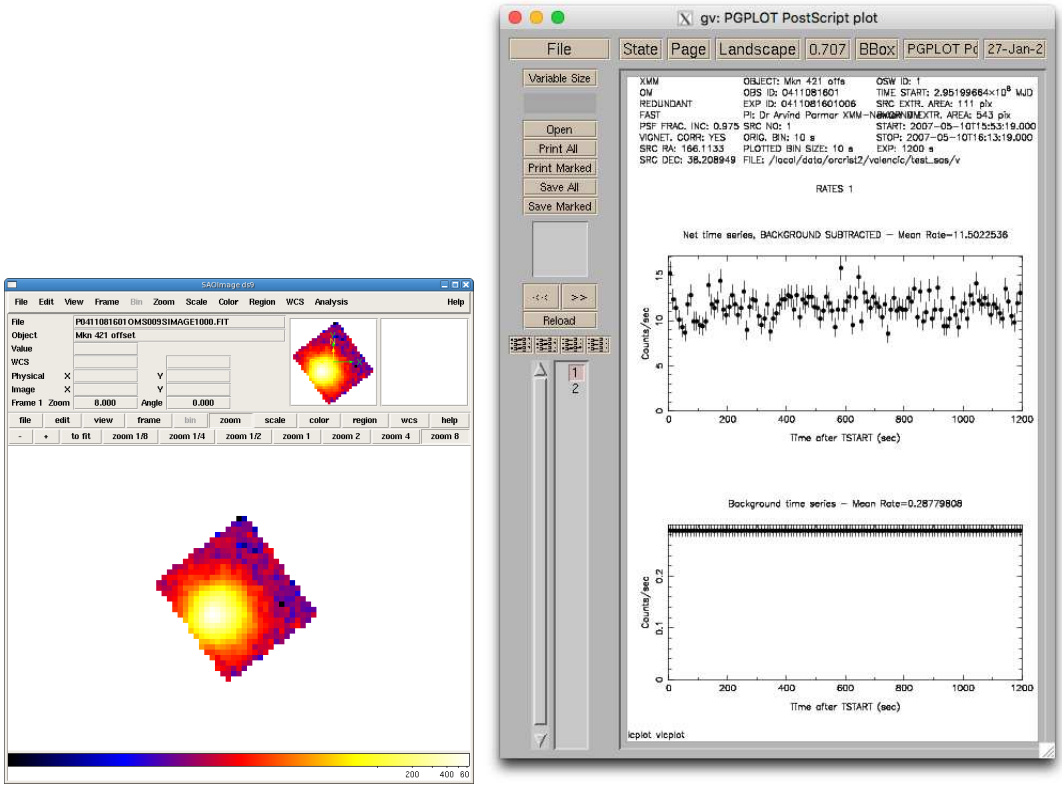
To run the pipeline on fast mode data on the command line, type

```
omgchain
```

Alternatively, *omgchain* can be run from the GUI:

1) Call *omgchain*.

Figure 12.2: Left: The processed Fast mode sky image. Right: the light curve produced automatically by *omfchain*.



2) In the task pop-up window, click "Run".

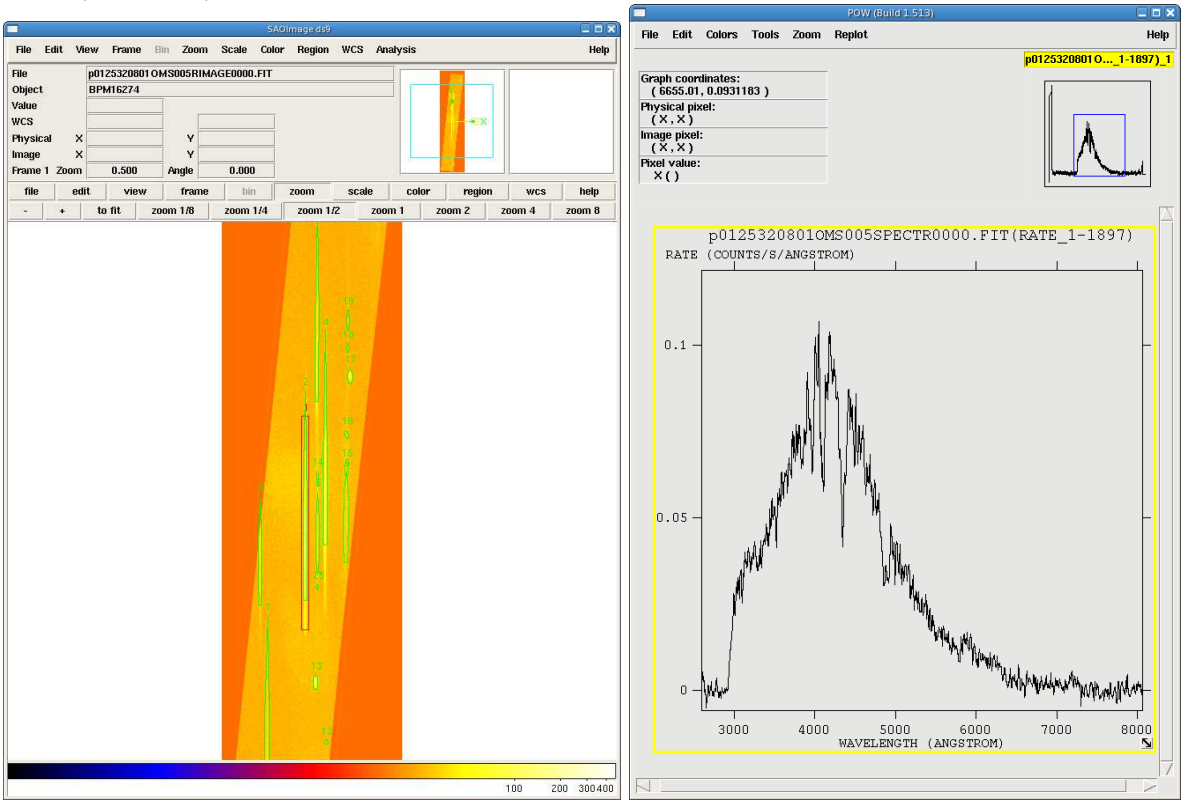
There are two types of output files: those that start with `g` are intermediate or auxiliary files and source lists; those that start with `p` are products.

To demonstrate some of these output files, we have rerun the pipeline on the example dataset. The processed image, rotated to align with the columns of the image (`p0125320801OMS005RIMAGE0000.FIT`), is shown in Fig. 12.3 (left). Two region files are overlayed: `p0125320801OMS005REGION0001.ASC`, which corresponds to the sources detected in this rotated image (green), and `p0125320801OMS005SPCREG0001.ASC`, which corresponds to the sources in the spectra list file (red) and indicates the locations of the zero and first orders. The task *omgchain* automatically extracted the spectrum of the red region (`p0125320801OMS005SPECTR0000.FIT`); this is shown in Fig. 12.3 (right).

12.4.2 Verifying the Output

The correct correlation of zero and first orders is crucial for grism analysis. Users should inspect the rotated image with *fv* or *ds9* and verify the identification of the orders by overlaying the `*SPCREG*` region file, as shown in Fig. 12.3 (left); the `*SPECLI*` file also contains this informtaion. If users are interested in all source detections, the region file can also be overlayed and the full source list examined. Users should also examine the spectra plots automatically produced by *omgchain*, for both the source and background, making sure they are reasonable. For improved source detection, the parameter `nsigma` can be changed.

Figure 12.3: Left: The repipelined, rotated image with regions overlayed. Right: the spectrum extracted from the source (red region).



Chapter 13

Fitting an EPIC Spectrum in XSPEC

In this chapter, we will do some simple spectral fitting, using as an example the source and background that was extracted from the Lockman Hole observation (Obs ID 0123700101) in §6 (or §7 if you used the GUI), and its RMF and ARF files. All tasks will be called from the command line.

Prior to fitting, users are encouraged to read this Guide's discussion on statistics in §14.1. More information can be found in Humphreys et al. 2009, ApJ, 693, 822.

For this example, we are interested in just a quick fit to see what we are dealing with, so we will rebin the data. The procedure *grppha* from FTOOL provides an excellent mechanism to do that. The following commands not only group the source spectrum for Xspec but also associate the appropriate background and response files for the source. This is done simply by calling the task and editting parameters as is appropriate:

```
> ftools
> grppha
   Please enter PHA filename[] mos1_pi.fits      ! input spectrum file name
   Please enter output filename[] mos1_grp30.fits  ! output grouped spectrum
   GRPPHA[] chkey BACKFILE bkg_pi.fits          ! include the background spectrum
   GRPPHA[] chkey RESPFILE mos1_rmf.fits         ! include the RMF
   GRPPHA[] chkey ANCRFILE mos1_arf.fits         ! include the ARF
   GRPPHA[] group min 30                         ! group the data by 30 counts/bin
   GRPPHA[] exit
```

Next, use Xspec to fit the spectrum, editting the parameters as needed. We'll use two lightly absorbed power laws as a first try.

```
   XSPEC> data mos1_grp.fits      ! input data
   XSPEC> ignore 0.0-0.2,6.6-**   ! set a range appropriate for the data, in keV
   XSPEC> model pow               ! set spectral model to a power law
   1:powerlaw:PhoIndex> 2.0       ! set the power law photon index to 2.0
   2:powerlaw:norm>               ! default model normalization
   XSPEC> renorm                  ! renormalize the model spectrum
   XSPEC> fit                     ! fit the model to the data
   XSPEC> setplot device /xw      ! set the plot device
   XSPEC> setplot energy          ! plot energy along the X axis
   XSPEC> plot ldata ratio        ! plot two panels with the log of the data and
                                  !     the data/model ratio values along the Y axes
   XSPEC> exit                    ! exit Xspec
   Do you really want to exit? (y) y
```

The spectrum is shown in Figure 13.1.

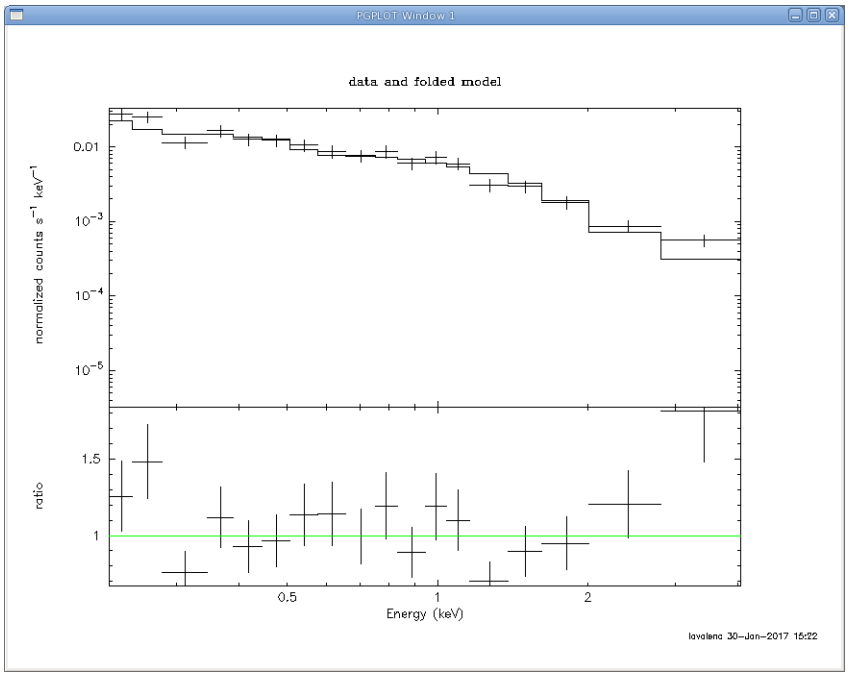


Figure 13.1: An EPIC MOS spectrum fitted with Xspec.

Chapter 14

Fitting an RGS Spectrum in XSPEC

Before we get to fitting the spectrum, a few words are in order about different approaches to take. We will use for our example the RGS spectrum of Mkn 421 (ObsID 0153950701) that we reprocessed in §10 (or §11 if we used the GUI).

14.1 Approaches to Spectral Fitting and the Cash Statistic (cstat)

For data sets of high signal-to-noise and low background, where counting statistics are within the Gaussian regime, the data products above are suitable for analysis using the default fitting scheme in Xspec, χ^2-minimization. However, for low count rates, in the Poisson regime, χ^2-minimization is no longer suitable. With low count rates in individual channels, the error per channel can dominate over the count rate. Since channels are weighted by the inverse-square of the errors during χ^2 model fitting, channels with the lowest count rates are given overly-large weights in the Poisson regime. Spectral continua are consequently often fit incorrectly, with the model lying underneath the true continuum level. This will be a common problem with most RGS sources. Even if count rates are large, much of the flux from these sources can be contained within emission lines, rather than the continuum. Consequently, even obtaining correct equivalent widths for such sources is non-trivial.

The traditional way to increase the signal-to-noise of a data set is to rebin or group the channels, since, if channels are grouped in sufficiently large numbers, the combined signal-to-noise of the groups will jump into the Gaussian regime. However, this results in the loss of information. For example, sharp features like an absorption edge or emission line can be completely washed out. Further, in the Poisson regime, the background spectrum cannot simply be subtracted, as is commonly done in the Gaussian regime, since this could result in negative counts. Therefore, rebinning should be reserved for fast, preliminary analysis of spectra without sharp features, or for making plots for publication. When working on the final analysis for a low-count data set, the (unbinned) background and source spectra should be fitted simultaneously using the Cash statistic. (If fitting with XSPEC, be sure you are running v11.1.0 or later. This is because RGS spectrum files have prompted a slight modification to the OGIP standard, since the RGS spatial extraction mask has a spatial-width which is a varying function of wavelength. Thus, it has become necessary to characterize the BACKSCL and AREASCL parameters as vectors (i.e., one number for each wavelength channel), rather than scalar keywords as they are for data from the EPIC cameras and past missions. These quantities map the size of the source extraction region to the size of the background extraction region and are essential for accurate fits. Only Xspec v11.1.0, or later versions, are capable of reading these vectors, so be certain that you have an up-to-date installation at your site.)

Finally, a caveat of using the Cash statistic in Xspec is that the scheme requires a "total" and "background" spectrum to be loaded into Xspec. This is in order to calculate parameter errors correctly. Consequently, be sure not to use the "net" spectra that were created as part of product packages by SAS v5.2 or earlier. To change which statistic you use in Xspec before fitting the data, type:

```
XSPEC>  statistic cstat
```

A more in-depth discussion on statistics in the Poissonian regime can be found in Humphreys et al. 2009, ApJ, 693, 822. For our purposes, a quick, preliminary fit is sufficient, so we will rebin and use χ^2 statistics.

14.2 Rebinning the Spectrum

There are two ways to rebin a spectrum: the FTOOL grppha, or the RGS pipeline. *grppha* can group channels using an algorithm which bins up consecutive channels until a count rate threshold is reached. This method conserves the resolution in emission lines above the threshold while improving statistics in the continuum. However, while channel errors are propagated through the binning process correctly, the errors column in the original spectrum product is not strictly accurate. The problem arises because there is no good way to treat the errors within channels containing no counts. To allow statistical fitting, these channels are arbitrarily given an error value of unity, which is subsequently propagated through the binning. Consequently, the errors are overestimated in the resulting spectra.

To rebin the spectrum and set the RESPFILE keyword in the header to our response file, type

```
> grppha
```

and edit the parameters as needed:

```
>Please enter PHA filename[] P0153950701R1S001SRSPEC1001.FIT
>Please enter output filename[] P0153950701R1S001SRSPEC1001.bin30.FIT
>GRPPHA[] chkey RESPFILE P0153950701R1S001RSPMAT1001.FIT
>GRPPHA[] group min 30
>GRPPHA[] exit
```

The other approach, which involves calling the RGS pipeline after it is complete, bins the data during spectral extraction. The following rebins the pipeline spectrum by a factor 3:

```
rgsproc rebin=3 rmfbins=4000 entrystage=4:spectra finalstage=5:fluxing
```

where

`rebin` - wavelength rebinning factor
`rmfbins` - number of bins in the response file; this should be greater than 3000
`entrystage` - entry stage to the pipeline
`finalstage` - exit stage for the pipeline

One disadvantage of this approach is that you can only choose integer binning of the original channel size. To change the sampling of the events, the pipeline must be run from the second stage ("angles") or earlier:

```
rgsproc nbetabins=1133 rmfbins=4000 entrystage=2:angles finalstage=5:fluxing
```

where the parameters are as defined previously, and

`nbetabins` - number of bins in the dispersion direction; the default is 3400

The disadvantage of using *rgsproc* is that the binning is linear across the dispersion direction. Velocity resolution is lost in the lines, so the accuracy of redshift determinations will be degraded, transition edges will be smoothed, and neighboring lines will become blended.

14.3 Fitting a Model

To fit the spectrum, invoke Xspec on the command line:

```
xspec
```

Enter the data, background, and response file at the prompts, and edit the fitting parameters as needed. Please note that in this example, we are using the output from *rgsproc*, not the PPS data that came with the data set, so the names are slightly different. If we were using the PPS data, the input spectrum would have the format `*SBSPEC*`. Since we did not include background correction when we ran *rgsproc*, we can correct for it now.

```
XSPEC> data P0153950701R1S001SRSPEC1001.bin30.FIT ! input data
XSPEC> back P0153950701R1S001BGSPEC1001.bin30.FIT ! input background
XSPEC> ignore **-0.4                              ! set sensible limits
XSPEC> model wabs*pow                             ! set spectral model to absorbed powerlaw
1:wabs:nH> 0.01                                   ! enter reasonable initial values
2:powerlaw:PhoIndex> 2.0
3:powerlaw:norm> 1.0
XSPEC> renorm
XSPEC> fit
XSPEC> cpd /xw
XSPEC> setplot wave
XSPEC> setplot command window all
XSPEC> setplot command log x off
XSPEC> plot data chi
XSPEC> exit
```

Figure 1 shows the fit to the spectrum.

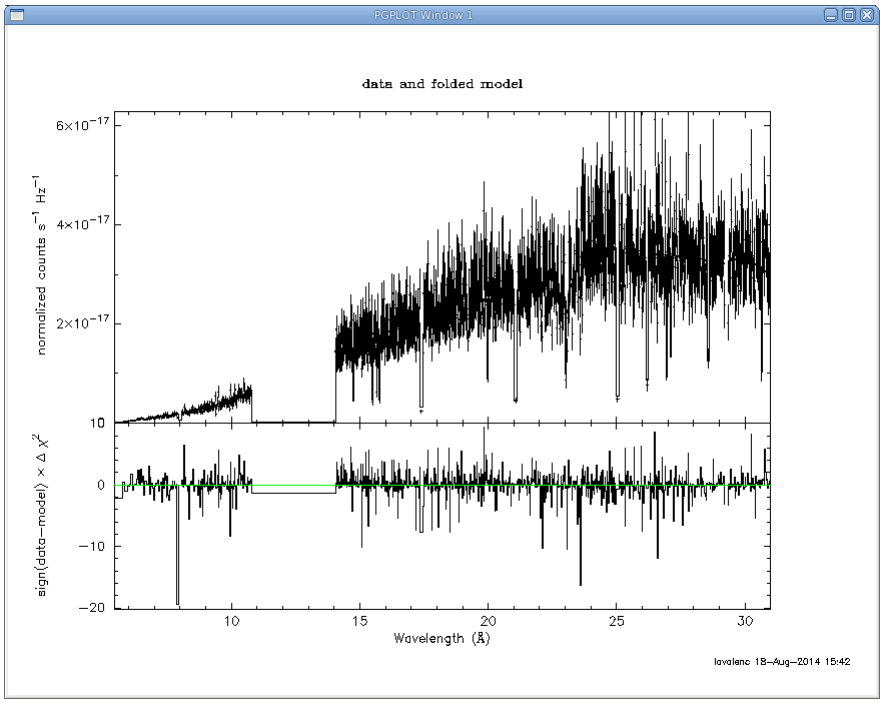


Figure 14.1: The 1st order RGS1 spectrum fitted with an absorbed power law model. The gap between 10-15 Å is due to the absence of CCD7.

Chapter 15

Fitting EPIC Timing Data with Xronos

We will do some analysis of PN Timing data of the high mass X-ray binary GX 301-2 (Obs ID 0555200301) with Xronos, which is a powerful timing analysis software package available through the HEASARC. More about Xronos can be found here:

`http://heasarc.gsfc.nasa.gov/xanadu/xronos/xronos.html`

We will assume that the data has been reprocessed with SAS, and a light curve with time bin size of 1 second was made with SAS and named `pn_ltcrv`. Xronos does not have a GUI, so all tasks will be called from the command line. In the examples below, all the tasks prompt the user for further information. Input that we've been prompted for is shown with the indented lines.

Let's start by pulling up a new window and invoking ftools:

```
ftools
```

15.1 Making a Power Spectrum

The task *powspec* calculates the power spectrum density for a given time series, plots the result, and writes the results to a FITS file. We can use the recommended defaults as a starting point.

```
powspec
   pn_ltcrv.fits
   -
   25
   4096
   1
   0
   [enter]
   [enter]
   [enter]
```

The output is shown in Figure 15.1. We can zoom in on a region of interest by resetting the axis minimum and maximum. To quit the plotting software, just type 'exit'.

```
r x 0 3e-3
exit
```

We can also do this with geometric rebinning:

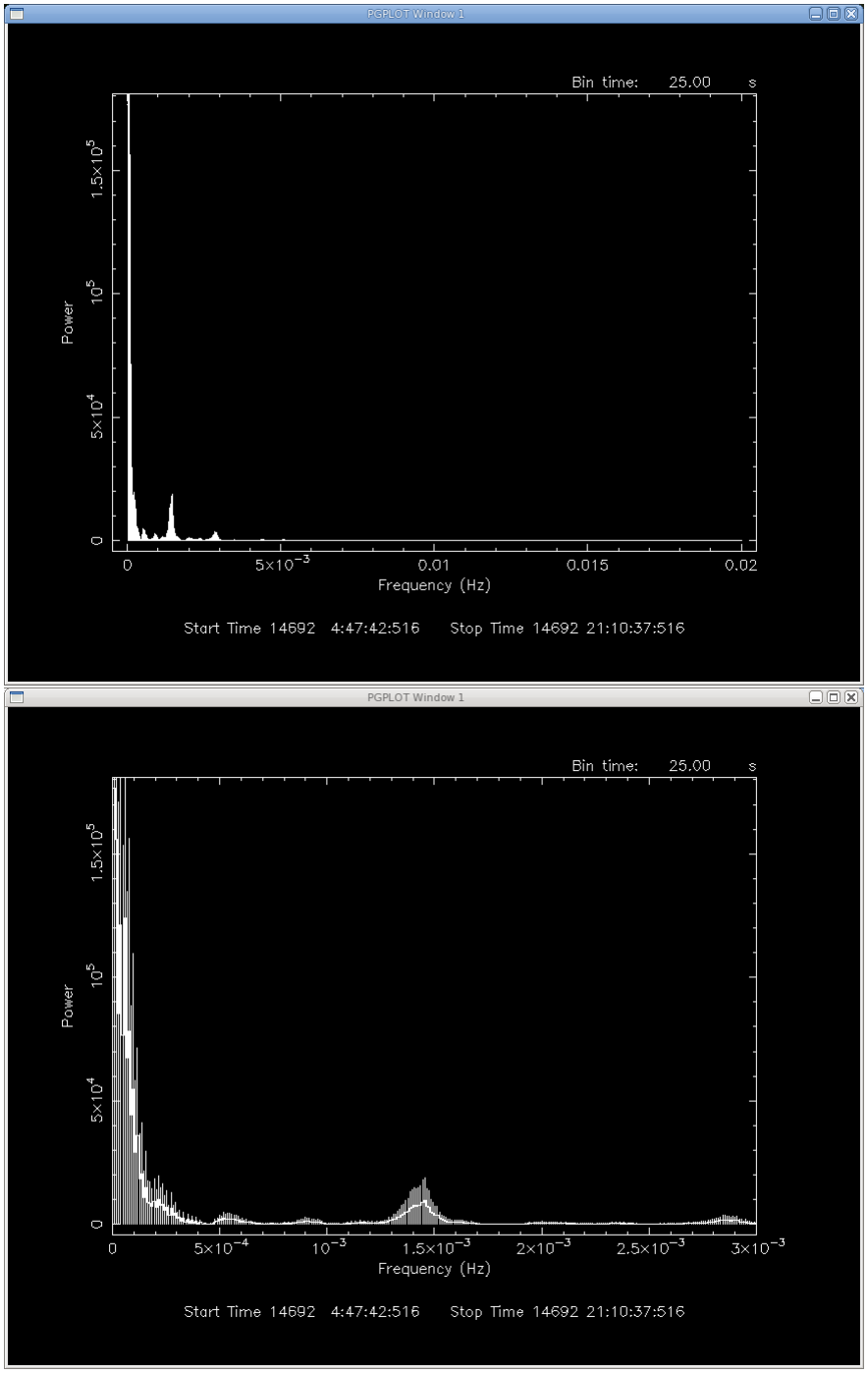


Figure 15.1: Top: The power spectrum of GX 301-2. Bottom: Same, zoomed in.

```
powspec
   pn_ltcrv.fits
   -
   25
   4096
   1
   -1.01
   [enter]
```

```
   [enter]
   [enter]
   exit
```

The output is shown in Figure 15.2.

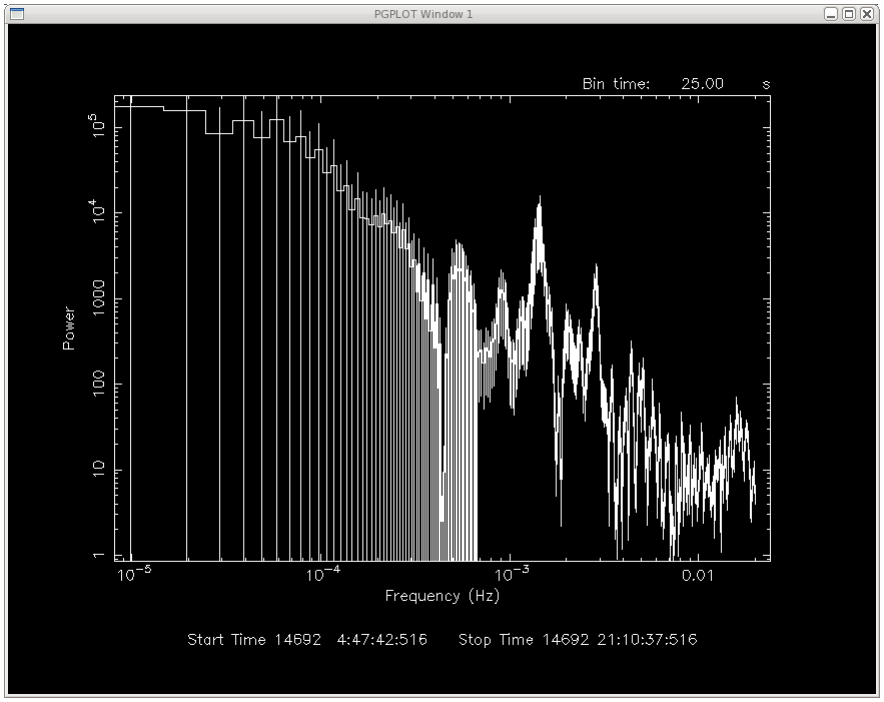


Figure 15.2: The power spectrum of GX 301-2 with geometric rebinning.

15.2 Finding the Period of a Source

We can also search for the source's period with the task *efsearch*. The period was determined by Furst et al. (2011, A&A, 535A, 9) to be 685 s, so we will use that as a starting point.

```
efsearch
   pn_ltcrv.fits
   -
   INDEF
   685
   8
   689
   4
   128
   [enter]
   [enter]
   [enter]
   exit
```

The output is shown in Figure 15.2. *efsearch* automatically indicates the best period and resolution at the top of the plot, with the bin size on the upper right side.

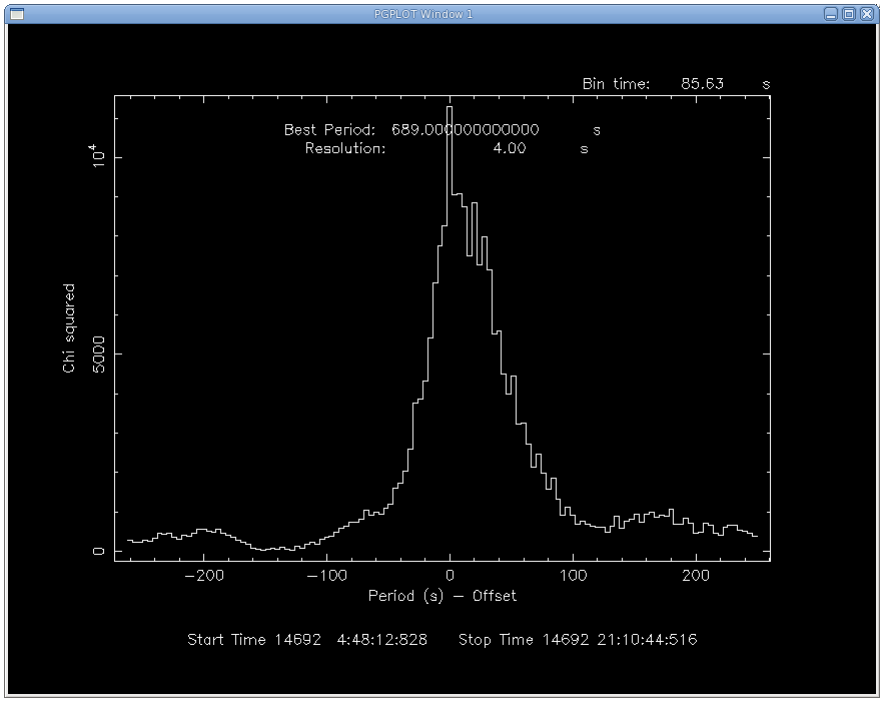


Figure 15.3: The output of *efsearch*. By default, the X axis represents the difference between the periods attempted and the best period found.

15.3 Making a Folded Light Curve

The task *efold* will produce a folded light curve.

```
efold
   1
   pn_ltcrv.fits
   -
   INDEF
   689
   10
   857
   1
   [enter]
   [enter]
   [enter]
```

We can add a line to the plot to make it easier to see.

```
   line step
   p
```

And color, if we are feeling fancy. In the statement below, "3" refers to the color choice and "2" refers to the data on the Y axis. If we wanted to turn off the error bars, we could also enter the command `error off`.

```
   color 3 on 2
   p
   exit
```

The plot is shown in Figure 15.4. By default, the folded period is shown in the upper right part of the plot.

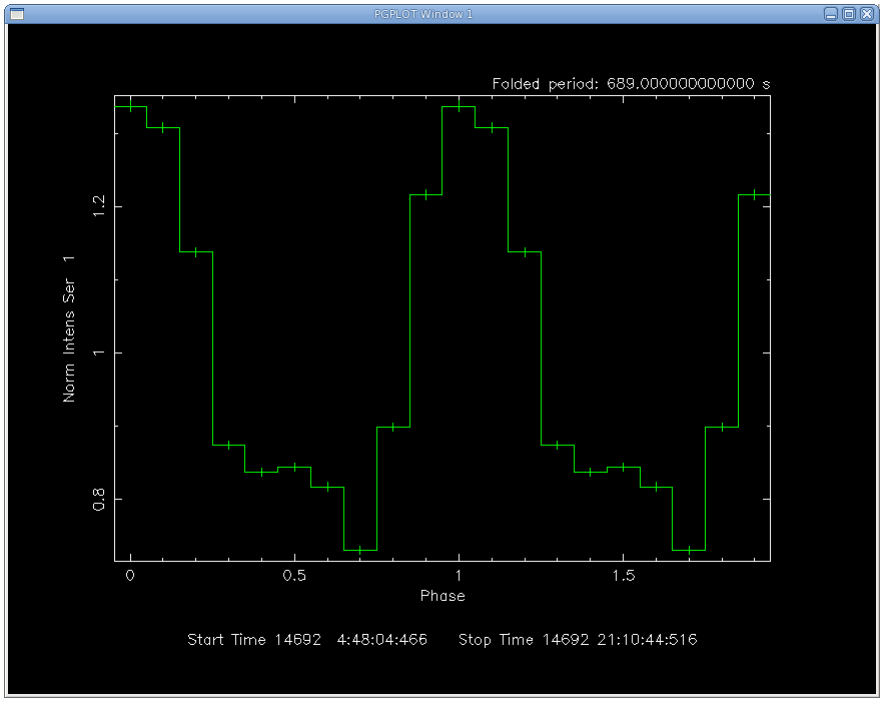


Figure 15.4: The folded period of GX 301-2.