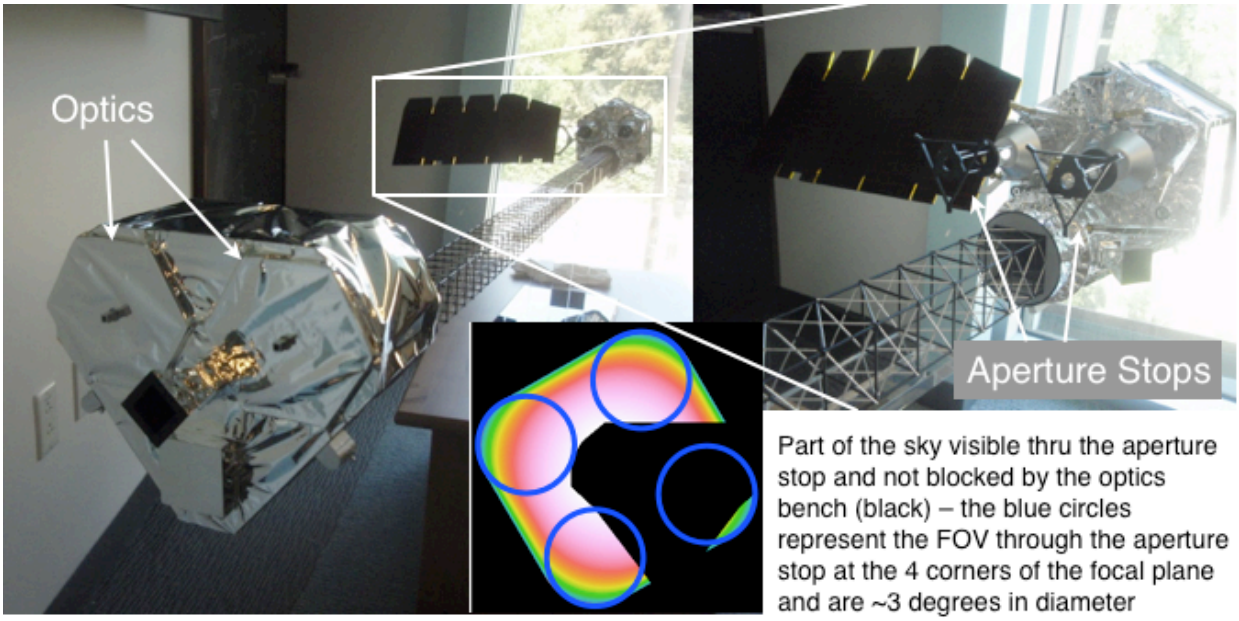
How-to Guide for Modeling the Background in the Sky Frame to Produce Spectra and Images with nuskybgd

(disclaimer: these tools are provided for your convenience only – the user assumes sole responsibility – but if you find them useful, please acknowledge in publications!)

Purpose of nuskybgd and the nature of the background components it models

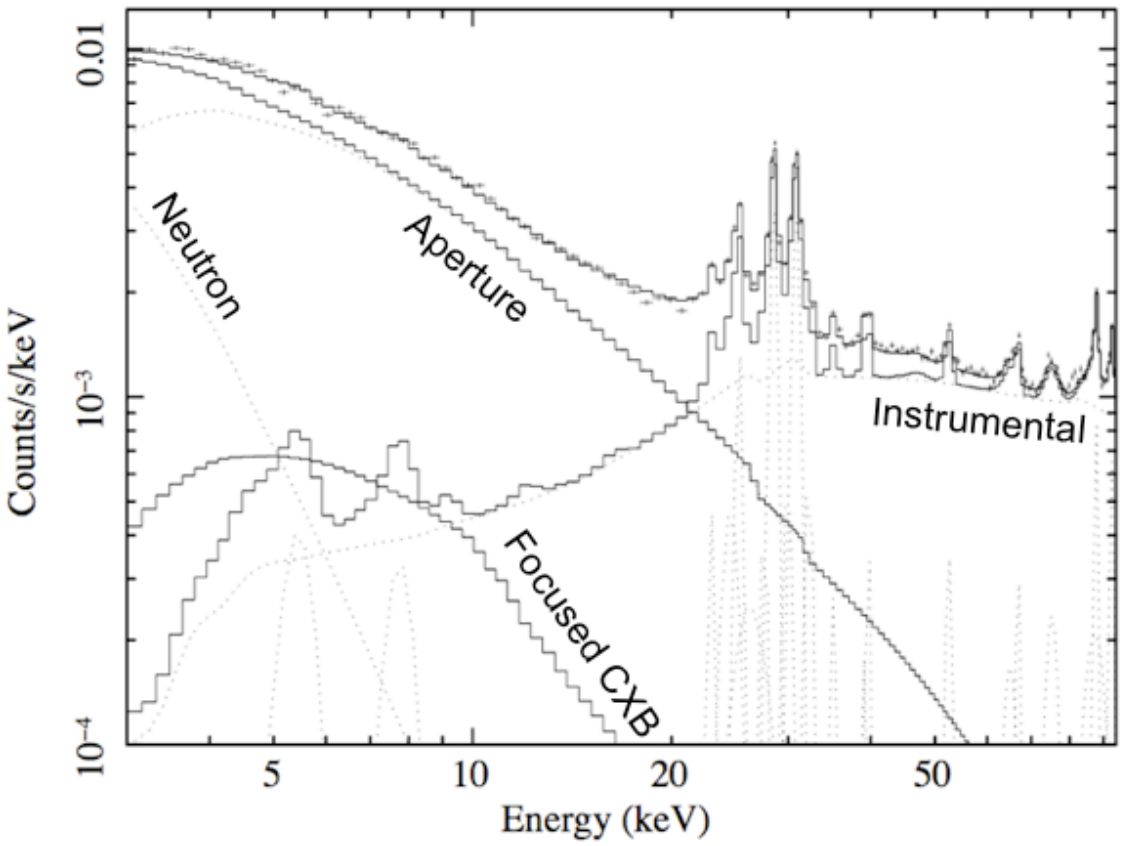
The *NuSTAR* observatory has some unique features that give rise to various, independent background (often abbreviated 'bgd' hereafter) components that vary spatially across the field of view (FOV), invalidating simple bgd estimation techniques. The purpose of `nuskybgd` is to take what we know about the behavior of the spatial variation of the bgd in the detector plane and translate that to the sky frame. In this framework, observers are free to choose and extract bgd regions in a more standard way, allowing the software to keep track of the idiosyncratic details of the bgd. Once the bgd is characterized (or a set of default parameters assumed, if appropriate), you can produce properly scaled bgd spectra for any arbitrary region and/or bgd images for arbitrary energy bands.

Characterizing the bgd in `nuskybgd` requires source-free regions (including from the wings of the PSF of bright sources); if no such regions exist, then `nulyses` is a better option for obtaining bgd spectra. If you understand the nature of the contamination, it is possible to include that when modeling the bgd spectra so the bgd characterization will not be biased.



NuSTAR consists of two separate telescopes (two sets of optics focusing onto two focal planes, creatively referred to as A and B). Each focal plane is covered by a 2x2 array of CZT detectors (also occasionally called 'chips') that are electronically pixelated into a 32x32 array. In principle, each pixel has a unique bgd response, but in practice all the pixels on a single detector – excepting edge pixels – behave similarly. Due to differences in thickness and other properties of the detectors, the instrumental bgd for each detector is somewhat unique. The optics and detectors are separated on two ends of an unenclosed boom, which are constantly moving with respect to each other, so a given detector pixel samples several times more sky than without this wobble. Because the light path is open to space, stray light from the cosmic X-ray background (CXB) is able to skirt between the optics bench and the aperture stops in front of the two focal planes; the geometry of this window produces highly non-uniform bgd gradients across the detectors. The low altitude and inclination orbit of NuSTAR minimizes SAA activation and proton flares, but soft environmental neutrons (most likely) contribute to the bgd at the lowest energies.

The bgd spectrum can generally be decomposed into four components of fixed spectral shape (excluding instrumental line strengths) with individual normalizations that are position dependent. For certain observations near the Galactic plane in the direction of the Galactic center, an additional non-uniform component is also needed. The spectral components, fit to a combined spectrum of many observations over the entire FOV of focal plane A, are shown below.



Below ~20 keV, the background is dominated by stray light from unblocked sky leaking through the aperture stop; when the origin of this emission is the CXB, this component is referred to as the 'aperture bgd.' Near the Galactic center, individual bright sources and diffuse Galactic Ridge emission (GRXE) can also contribute. A GRXE component will be incorporated in an imminent update of `nuskybgd`, but as of now is not included. By their nature, the bgd they (CXB, GRXE, or bright sources) produce is spatially non-uniform. Below ~5 keV, there is a strong, very soft additional component that is most likely due to neutrons striking the detectors. This component is poorly characterized spectrally but appears to be spatially uniform across all detectors. The other low energy contributor to the bgd is from the CXB 'focused' by the optics (fCXB). This includes both 2-bounce and 1-bounce photons[1] from the many still unresolved sources within the FOV AND 1-bounce photons from sources outside the FOV. Its shape is roughly flat across the detector plane despite vignetting due to scattered light from sources outside the FOV.

Above ~20 keV, the instrumental or internal background dominates. It is made up of many lines activated by interactions between the spacecraft/detectors and the radiation environment in orbit, as well as several fluorescence lines. Many of the weaker lines make up a roughly power law-like continuum, which is modeled as a broken power law with a break at 30 keV. Over 20 individual Gaussian lines are needed to match the observed spectrum, some of which represent single lines and some of which represent line complexes. The energies of these 'model lines' in `nuskybgd` is entirely empirical and is not based on a physical model for their origin. The overall strength (and the relative strengths of the individual lines) of the internal bgd is detector dependent, but does not appear to spatially vary within a single detector.

Using early blank fields (most notably from the North Ecliptic Pole) and all the extragalactic survey fields complete to date, an empirical spatial-spectral model of all background components was derived. `nuskybgd` implements that characterization of the background for any given observation and generates spectra or images (in sky coordinates) for the user. User-discretion is *strongly* advised (if you are happy to blindly trust the output, I have a boat I would be happy sell you).

[1] 'Bounce' in this context describes whether a photon has undergone the two designed reflections (2-bounce) or whether it is scattered light, having only reflected off of only one of the mirrors (1-bounce). This latter type populates the wings of the PSF.

Preliminaries: Installation and Setup

nuskybgd routines are all written in IDL and occasionally spawn HEASoft processes (namely XSPEC – I don't think any others are called at the present time). No effort has been made to ensure that the code works with all the various versions of IDL and HEASoft, and while it shouldn't do anything fancy enough for this to matter, certain kinds of bugs (especially in IDL) can cause problems under one distribution but not in others.

First, download the code from the public svn repository:

```
cd /path/
svn co https://www.srl.caltech.edu/svn/nustar_public nustar_public
```

This will add the svn tree to whatever directory you're in. The code will be updated regularly, and without warning, so be sure all the routines are from the same distribution. Because the svn is public (meaning you are free to modify the code and update the repository), please copy the code from here to somewhere in your IDL path or typical IDL working directory:

```
cp /path/nustar_public/trunk/background/nuskybgd/* /path/to/workingdir/
```

Whenever you update the repository, via

```
cd /path/
svn update
```

you will again need to copy ALL code (and auxil/ contents, see below) to that location. If you find a small typo-like bug or similar AND you know what you're doing, you are free to update the code in the repository, but please reserve this for small changes or you may ruin everything for everyone. Bugs should be reported to me via email.

To complete the setup, you need the CALDB environment variable set correctly (should already be), nuabs installed (`lmod nuabs` inside XSPEC should load it, see `nuskybgd.pro` for details), and a new environment variable set thusly (for csh variants):

```
setenv NUSKYBGD_AUXIL /path/to/nuskybgd/auxil/directory/
```

A sensible location for this is your `$CALDB_AUXIL` directory. Wherever you decide to put it, copy the contents of the svn auxil/ to it:

```
cp /path/nustar_public/trunk/background/nuskybgd/auxil/* $NUSKYBGD_AUXIL/
```

1) Make reference spectra for bgd components

Because nuabs parameters may change, reference count rate spectra need to be generated by the user. If the `$NUSKYBGD_AUXIL/nuabs.dat` file changes, either by you (`nuabs.dat` is a simple text file) or an update, you will need to rerun this command, otherwise this is a one time deal. The output pha files are stored in `$NUSKYBGD_AUXIL`.

```
prompt> cd /path/to/workingdir/
prompt> idl
IDL> nuskybgd_imrefspec
```

If you experience problems at this stage, you did not follow the installation instructions carefully enough. BEWARE: In a previous version of this guide I said the absorption would be included in the RMF and nuabs would no longer be needed. *I lied.* The absorption has been included in the ARFs, which `nuskybgd` does not use, so as of now the nuabs XSPEC model is still required.

2) Making instrument maps

To be annoying, `nuskybgd` makes certain assumptions about your directory structure that are admittedly inflexible (or stupid), but such is life. Wherever you choose to start IDL (e.g., `/path/to/workingdir/`), you need to set a variable indicating the relative location of the data directory (an absolute location is fine too) and the ObsID, which is assumed to be a directory just below the data directory and which contains the observation `auxil`, `hk`, and `event_uf` directories. Your data was distributed with this directory structure, so that's fine. After running `nupipeline`, you will also have an `event_cl` under the ObsID directory containing the cleaned event files. *This directory is hardcoded into `nuskybgd`.* If you want to run `nupipeline` several times with different settings, `nuskybgd` can only work with the files in a directory explicitly named `event_cl`, and certain files are assumed to be in this directory or a subdirectory of it, so renaming this directory is problematic. The workaround is not that onerous, however, so stop your complaining.

As an example, let's assumed you've unpacked the tar file for Mrk 231 in `/path/to/workingdir/`, which is where you've also started IDL, so the path to the data is `/path/to/workingdir/60002025_Mrk231/60002025002/`. It is useful to define these variables:

```
IDL> dir='60002025_Mrk231'                          (relative case)
 or
IDL> dir='/path/to/workingdir/60002025_Mrk231'      (absolute case)
IDL> obsid='60002025002'
```

Instrument maps specific to your observation are needed to account for bad pixels that have been excluded from processing, especially if you specified a user bad pixel list during the call to `nupipeline`. The first time the following routine is run, images are created that map RAW detector pixels (the true 32x32 pixel array) to the more

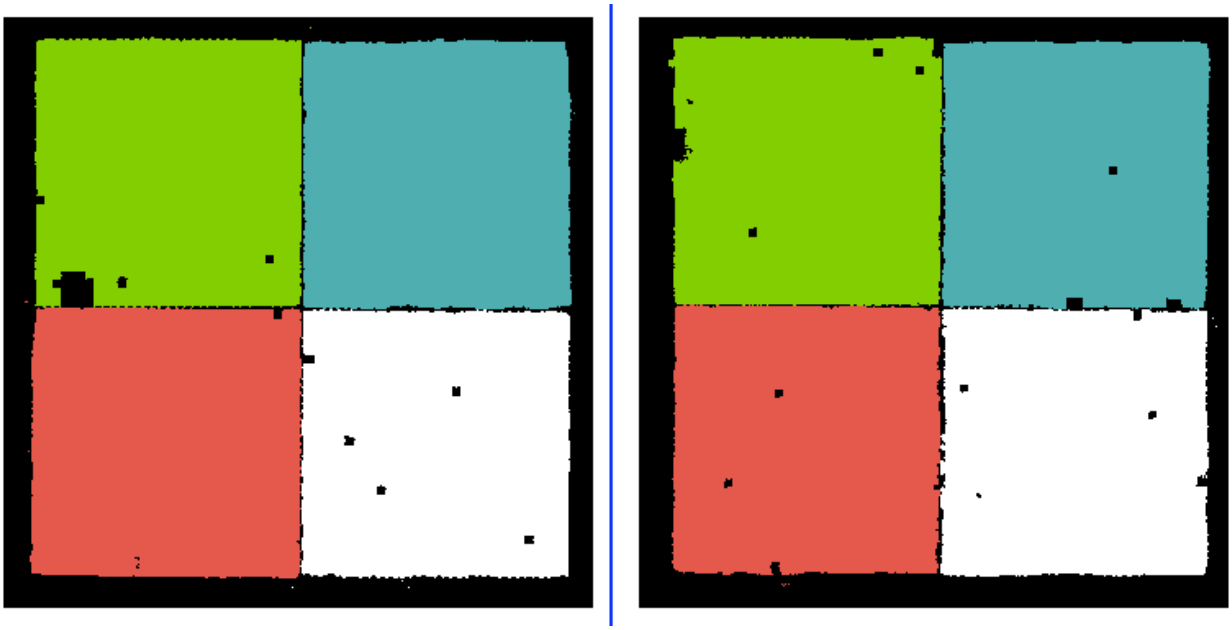
oversampled DET1 pixels based on grade-specific PDFs (weighted by the grade frequency of a typical observation) in the CALDB. (During `nupipeline`, events are assigned DET1 values probabilistically based on their grades.) These pixel maps are stored in `$NUSKYBGD_AUXIL`.

```
IDL> nuskybgd_instrmap,dir,obsid,'A','bgd'
IDL> nuskybgd_instrmap,dir,obsid,'B','bgd'
```

These commands create the new files inside a new directory:

```
prompt> ls /path/to/workingdir/60002025_Mrk231/60002025002/event_cl/bgd/
newinstrmapA.fits
newinstrmapB.fits
```

The new directory is where all observation-specific reference files are stored and will be used as a parameter in calls to other routines. Call it `snuffleupagus` for all I care, but it is assumed to be called `bgd` in this document. The maps look something like this:



Focal plane A is on the left, B on the right, and bad pixels have been excised. Detector names 0, 1, 2, and 3 are (counterclockwise from the top right) shown in blue, green, red, and white, respectively.

3) Choosing and defining bgd regions

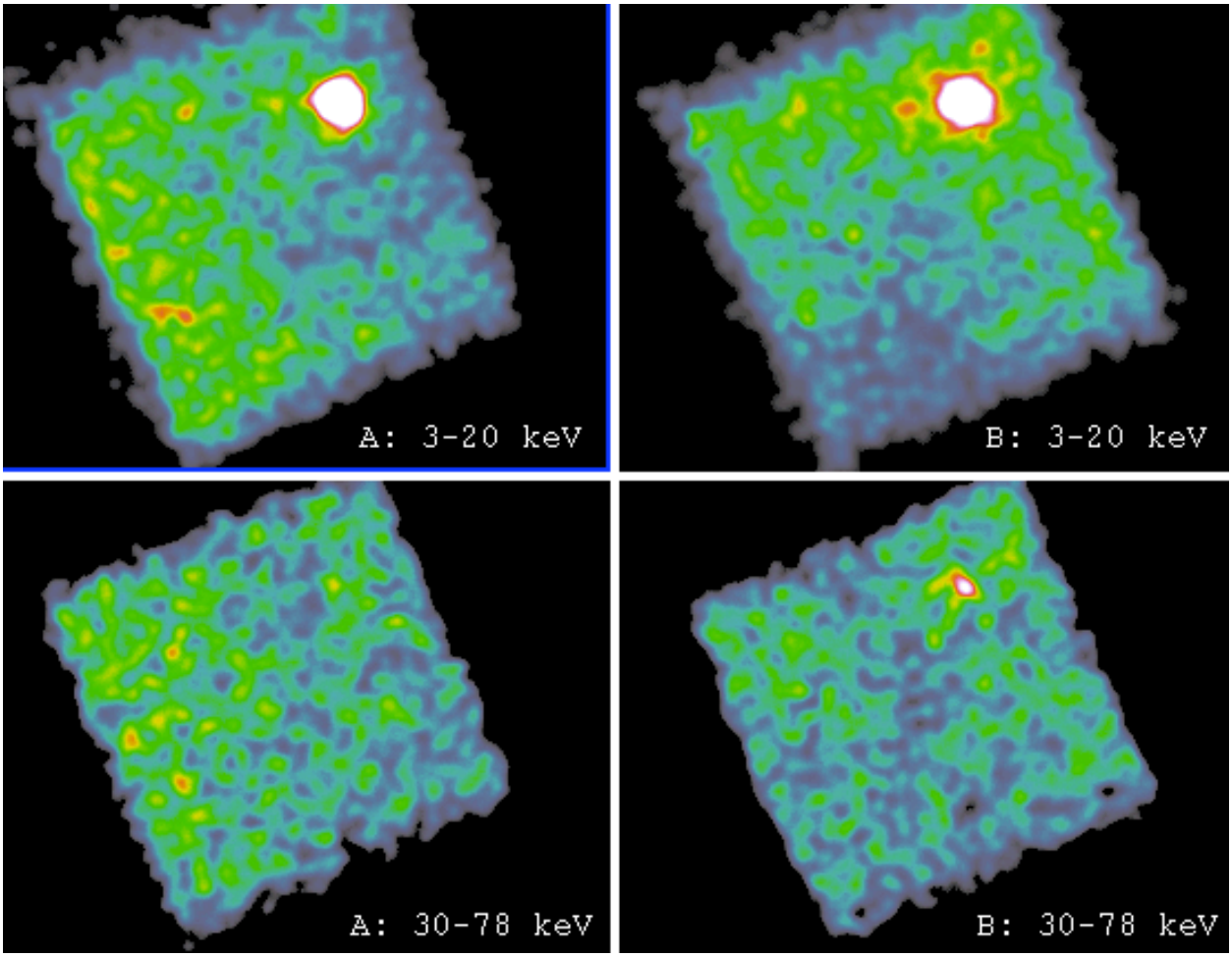
In principle, we know what contributes to the background and the spectral shapes of all those components, but for a given observation we don't know what the normalizations of all those components will be. Unfortunately, because the aperture bgd and the instrumental bgd vary spatially across the FOV, you cannot simply extract a background

from elsewhere and scale it for your source region or the FOV as a whole. The spatial variation is known, however, so it's just a matter of appropriately scaling each normalization, determined from bgd regions, for any region of interest.

First, you need images from which to choose background regions. For convenience, a simple python script is distributed that can create images for the A and B telescopes in arbitrary energy bands from the cleaned event files:

```
prompt> mkimgs.py 60002025_Mrk231 60002025002 3,20 20,78
```

This command creates two sets of images in the 3-20 keV and 30-78 keV bands.

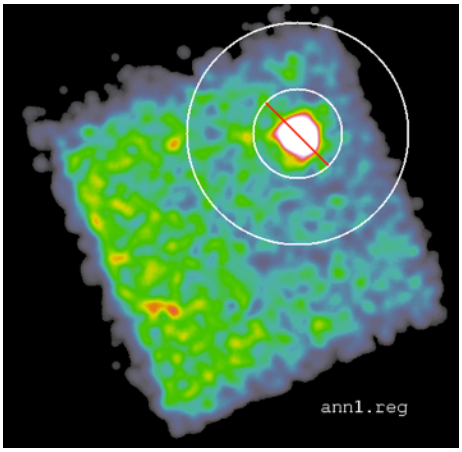


These images have been stretched and heavily smoothed to bring out the background. In the 3-20 keV band, the background is dominated by the aperture component, which smoothly varies across the FOV (but in different senses for A and B). There is still some contribution of the aperture bgd above 30 keV, but here the background is mostly internal and each detector (chip) has a unique overall normalization. These differences are harder to see because of small number statistics, but in the 30-78 keV image of

telescope A, notice that the normalization for detector 3 (top left) is noticeably higher than for the rest of the focal plane.

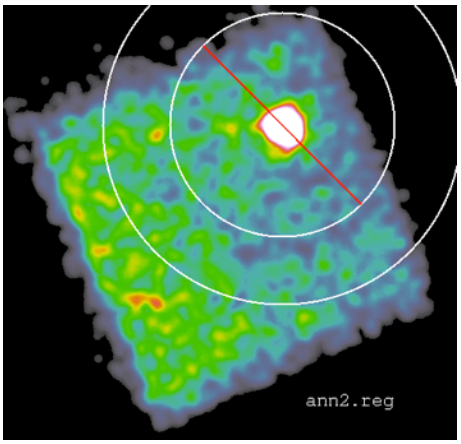
These images (or similar ones) are useful for identifying uncontaminated regions in which to characterize the background. Because of degeneracy between low energy bgd components, the goal is to simultaneously fit multiple bgd regions, tying normalizations between them based on the relative brightnesses expected from the known variation across the focal plane. How many regions you use and exactly where you place them is something worth experimenting with since it depends on the field and its exposure time. Regions that separate bright and faint areas of the aperture bgd gradient and/or individual detectors are good choices. However, dumb (I mean simple) regions work pretty well too, which is the case in the example described below.

Regions must be defined in the sky projection (hence the name of the code), but the region files themselves can have either fk5 or image units (or physical, but physical units are assumed to be equivalent to image units – the associated keywords for the physical coordinate system are not used). Also, this definition can only be specified once in a given file, and must be isolated on its own line or `nuskybgd` will not recognize it. Default DS9 region files should work. For this dataset, three annular bgd regions are chosen, identical for both A and B.



ann1[A/B].reg:

```
fk5
circle(194.06497,56.874489,300")
-circle(194.06497,56.874489,120")
```



ann2[A/B].reg:

```
# Region file format: DS9 version 4.1
# Filename: imA3to20keV.fits
global color=green dashlist=8 3 width=1
font="helvetica 10 normal" select=1 highlite=
1 dash=0 fixed=0 edit=1 move=1 delete=1
include=1 source=1
fk5
circle(194.06497,56.874489,480") # color=white
width=2 font="helvetica 10 normal roman"
-circle(194.06497,56.874489,300") # color=white
width=2 font="helvetica 10 normal roman"
```

The first region shows the basic format of a region file, and the second case shows what a default DS9 region file might look like, which is also acceptable. At present, the only shapes that `nuskybgd` understands are circles, ellipses, boxes, and polygons. Multiple include and exclude regions are allowed, but multiple include regions are ORed together (events extracted from either include region), not ANDed (events must be inside the overlapping part of the include regions) together. In this example, the third region (ann3[A/B].reg) covers the remaining part of the FOV.

4) Fitting the bgd spectra

This is the tricky part, but first you need spectra to fit. For this step, no ARFs are required, so extracting spectra with individual calls to `nuproducts` is fairly quick (since it's `numkarf` that takes most of the time). You can use the simple script provided or your own execution of `nuproducts` (there's nothing special about this script):

```
prompt> getspecnoarf.py 60002025_Mrk231 60002025002 ann1A bgdspec A
prompt> getspecnoarf.py 60002025_Mrk231 60002025002 ann1B bgdspec B
prompt> getspecnoarf.py 60002025_Mrk231 60002025002 ann2A bgdspec A  etc.
```

In the above, each call extracts one spectrum from the region file (ann1A.reg, ann1B.reg, etc.) into the directory `60002025_Mrk231/60002025002/event_cl/bgdspec/`. The script also groups spectra (to 30 counts per bin), which is important because the chi-square statistic is used during fitting.

> Aside: Making Reference Projected Images
>
> If the header keyword `PA_PNT` is sufficiently accurate and uses the updated definition of the PA, this step is done automatically by later routines and can be ignored for now. If the images called det0im.fits, bgdap0.fits, etc., in your `event_cl/bgd/` directory incorrectly match the projected positions of the detectors in sky coords, you can remake them:
>
> ```
> IDL> projinitbgds,dir,obsid,header,'A','bgd',pa=pa,/clobber
> ```
>
> The header must be from an image (any image) with an fk5 WCS defined (obtained with fits_read), and you only need to specify `pa` if you don't want to use the `PA_PNT` keyword value. If this keyword is defined in the old way, which can be the case for earlier datasets not yet reprocessed, you can set this keyword:
>
> ```
> IDL> projinitbgds,dir,obsid,header,'A','bgd',pa=pa,/clobber,/oldpa
> ```
>
> This is the first thing to try if you get funny results and the reference images in the `bgd/` directory are wildly out of whack or XSPEC crashes.

Using XSPEC, the normalizations of all the background components are fit taking into account known spatial variations in order to break degeneracies at lower energies. Depending on the needs of the user, this step can be quick and simple or horrifying (unless you're an XSPEC12 ninja, in which case it's just less quick). By default, the code pauses inside XSPEC to allow you to evaluate and muck around with the fit. Be sure you understand what it's doing – even excellent-looking fits can poorly represent the true low energy component normalizations, which when extrapolated to another region or the entire image will result in utter carnage. Following our example:

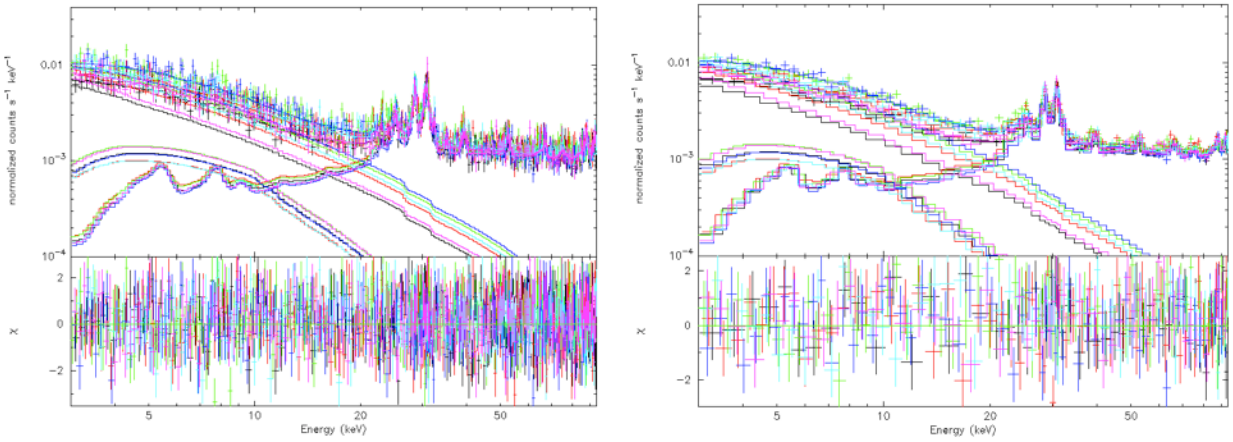
```
IDL> fits_read,dir+'/'+obsid+'/event_cl/imA3to20keV.fits',blah,header
IDL> bgdcore=['ann1A','ann2A','ann3A','ann1B','ann2B','ann3B']
IDL> bgdreg=bgdcore+'.reg'
IDL> bgdspec=bgdcore+'_sr_g30.pha'
IDL> nuskybgd_fitab,dir,obsid,bgdreg,'bgdspec',bgdspec,'AB','bgd',header
```

The fits image header – which must define fk5 WCS coords – only needs to be specified the first time the routine is called (it is used to create projected reference images, see box on previous page). The variables `bgdreg` and `bgdspec` define arrays with the spectral filenames and associated region filenames; the spectra are assumed to be in the `event_cl/bgdspec/` directory (based on the `'bgdspec'` parameter in the call) and again the region files are assumed to be in `event_cl/`. The parameter `'bgd'` indicates the directory where all the observation-specific reference files are stored (must be the same as in the `nuskybgd_instrmap` call). The remaining parameter `'AB'` tells the routine that spectra from both telescopes A and B are being provided – in this case, it is imperative that the regions are roughly the same for both telescopes and listed in the same order, with the A regions listed first. The spectra for A and B are fit independently except for the fCXB normalization, which is tied between the two since each sees the same piece of sky. By specifying `'A'` or `'B'` instead of `'AB'`, the spectra for each telescope can be fit completely independently (and of course only list the A or B spectra/regions in the call). For shorter observations, the fCXB norm can take unphysical values, so it adding this extra constraint can be helpful even though it makes it harder to tell what is going on.

When the routine pauses inside XSPEC, you will see something like:

```
----------
Check fit and adjust if necessary
  then run @60002025_Mrk231/60002025002/event_cl/bgdspec/bgdparams.xcm to
complete.
----------
```

This xcm script writes the best-fit values to temporary files that are then read in, scaled, and written to 'parameter' files that are used as input to the later routines to produce bgd spectra or images. The 'parameter' files are the goal of this step and are equivalent to the parameter files `bgdfitparams[A/B].dat` that proscribe the 'nominal' background. For a rough bgd spectrum or image, these files can be used instead and this step can be entirely skipped, which would be acceptable for *certain* applications.



The left panel is the resulting fit with the raw binning; the right panel is the same model with 10 channels/bin (for display purposes, using `setplot rebin 10 10`). The instrumental component dominates above 20 keV (you can see its low energy part drop with decreasing energy), and the aperture component dominates from 5-20 keV. The neutron component is included with the aperture model and causes it to turn up below 5 keV. The fCXB component lies below the aperture lines; as a rule of thumb, it should intersect the two bumps that are part of the low energy internal component. This is an example of a good fit The fCXB normalization is higher than average, which is probably explained by the existence of a few faint serendipitous sources in the chosen bgd regions. One may want to identify such faint sources and mask them out in the region files, depending on the end goal. In a sense, however, the fCXB component is more like a contaminant – you can't assume the flux is the same for two regions arcminutes apart since independent source populations are producing it. The purpose here is to separate its contribution from the aperture component, whose emission *is* correlated across the FOV. BUT BEWARE: whatever value is fit here is saved to the 'parameter' files and applied, by default, to background images and spectra – more on this later.

There are 3 named models, each with their own set of responses (ARFs). An optional fourth component representing Galactic Ridge X-ray emission (GRXE) can also be included by setting a `grxe` keyword, but this model is currently experimental.

```
Model apbgd:nuabs<1>(powerlaw<2>*highecut<3> + powerlaw<4>) Source No.: 2
Active/On
      For Data Group(s): 1 2 3 4 5 6

Model fcxb:nuabs<1>(powerlaw<2>*highecut<3>) Source No.: 4   Active/On
      For Data Group(s): 1 2 3 4 5 6

Model intbgd:nuabs<1>(gaussian<2> + gaussian<3> + gaussian<4> + gaussian<5> +
gaussian<6> + gaussian<7> + gaussian<8> + gaussian<9> + gaussian<10> +
gaussian<11> + gaussian<12> + gaussian<13> + gaussian<14> + gaussian<15> +
gaussian<16> + gaussian<17> + gaussian<18> + gaussian<19> + gaussian<20> +
gaussian<21> + gaussian<22> + bknpower<23>) Source No.: 3   Active/On
      For Data Group(s): 1 2 3 4 5 6
```

Any parameter fixed or tied should be left alone; fixing or constraining free parameters is OK, but better results here may not lead to better results later if the fit is unphysical. If your background regions are contaminated by an additional known source, you can add a model to account for its contribution. For example, the PSF wings of bright targets can contaminate the entire FOV. In this case, simply fit the un-bgd-subtracted source spectrum (over energies where bgd photons make up less than a few percent of the flux) and add that model here with a lower normalization. The normalization could be directly determined in this case by calculating the fraction of flux in each bgd region given the PSF shape, but allowing the norm.s to float *may* also be sufficient. The point here is to model-out non-bgd emission, so perfection is not necessary. It is up to the user to decide what is "good enough" to get accurate results. The model is created for the Source 1 responses, and only the RMFs are loaded so you will need to load appropriate ARFs along with the model:

```
XSPEC12> arf 1:1 whatever.arf
XSPEC12> arf 1:2 whatever.arf   etc. for all spectra
XSPEC12> model 1:arbitrarymodelname nuabs*(phabs*powerlaw) & =apbgd:1 &
=apbgd:2 & =apbgd:3 & =apbgd:4 & 0.1 -1 & 2.1 -1 & 0.01 & =apbgd:11 & =apbgd:
12 & =apbgd:13 & =apbgd:14 & 0.1 -1 & 2.1 -1 & 0.003 & etc. for all spectra
```

To set the `nuabs` parameters correctly for each spectrum (since they may vary), in the above they are tied to the corresponding values in the `apbgd` model, which has 10 parameters so you need to go by multiples of 10 for each spectrum. You can then mess with these parameters until you're happy with the fit:

```
XSPEC12> thaw arbitrarymodelname:6
XSPEC12> newpar arbitrarymodelname:13=arbitrarymodelname:6
XSPEC12> freeze arbitrarymodelname:7
XSPEC12> fit
```

This new model is not saved; its only purpose is to prevent non-bgd emission from biasing the characterization of the background components described here. When the background fit looks good, write the parameter file:

```
XSPEC12> @60002025_Mrk231/60002025002/event_cl/bgdspec/bgdparams.xcm
 XSPEC: quit
IDL>
```

The only output is a text file of normalizations for all the components, scaled either to the entire FOV or entire detectors (units are as defined in XSPEC). However, the aperture component (and GRXE, since its origin is the same) is given as a ratio to the expected CXB flux from HEAO-1 (Boldt 1987). More recent measurements have found a 10% higher normalization, and any given pointing will suffer from cosmic variance (the brightness distribution of sources varies from one part of sky to another), which is roughly a 10% effect. In the `auxil/` directory, the `bgdfitparams[A/B].dat` files give typical values from blank fields, which can be used in lieu of fitting for quick and dirty

results. In our example, the file (which is placed in the same directory as the bgd spectra, `bgdspec/`) looks like:

```
  9.2256E-01   # Factor relative to reference norm of PL for aperture comp
  1.4215E-04  1.1049E-04  1.6327E-04   # Norm of PL in CXB xspec model for
entire FOV
  7.6509E+00   # Norm of PL in xspec model of soft component for entire FOV
# Norms of instrumental line components for 4 detectors: 0 1 2 3
  1.0788E-04  7.3975E-05  1.3300E-04  2.5941E-04
  5.8417E-04  5.2813E-04  5.2970E-04  7.0081E-04
  4.6604E-04  5.0475E-04  7.8781E-04  8.2644E-04
  7.6088E-04  6.4555E-04  5.7991E-04  6.3204E-04
  1.6548E-03  9.8083E-04  3.0476E-03  3.5385E-03
  4.5155E-04  4.2611E-04  6.0626E-04  4.5472E-04
  3.2275E-03  3.3988E-03  2.7814E-03  2.7431E-03
  7.2994E-04  6.6400E-04  1.2627E-03  1.2220E-03
  3.9857E-03  3.9904E-03  3.5511E-03  3.2824E-03
  1.1858E-03  1.0226E-03  1.0771E-03  1.2142E-03
  2.7866E-04  2.6327E-04  3.4848E-04  1.1808E-04
  6.6350E-04  6.7771E-04  7.5541E-04  8.9118E-04
  4.1959E-04  1.9223E-04  4.3827E-04  4.6657E-04
  3.9915E-04  1.7255E-04  4.4299E-04  1.6354E-04
  7.7073E-04  5.6417E-04  7.5421E-04  9.4935E-04
  1.1787E-03  1.4886E-03  1.3115E-03  2.1145E-03
  2.9672E-04  3.6215E-04  2.2079E-04  3.5676E-04
  1.0054E-03  2.6007E-03  7.7596E-04  1.9011E-03
  1.7424E-03  4.5707E-03  2.1841E-03  1.9373E-03
  1.1721E-03  9.3249E-04  6.8395E-04  8.1570E-04
  1.4687E-03  1.5703E-03  9.9346E-04  1.0317E-03
# Norms of instrumental continua for 4 detectors: 0 1 2 3
  9.5624E-06  9.2225E-06  1.0000E-05  1.1125E-05
```

These values are read in by `nuskybgd_spec` and `nuskybgd_image` to produce background spectra and images for the observation. You are free to change these values by hand if you like – the line energies are given in the first column of the `ratios[A/B].dat` files in the `auxil/` directory.

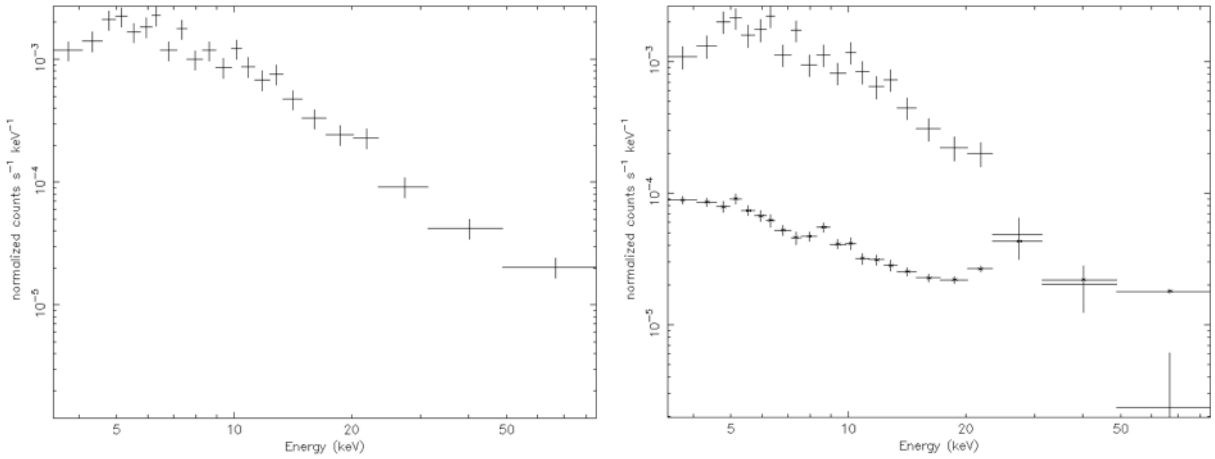
5) Making background spectra for a source region

In principle, if the fitting was successful and our assumptions about the background hold, we know what the background is doing anywhere in the detector plane. At present, it is unknown to what extent this is really true, but preliminary tests suggest that the systematic error on the bgd in a given energy band is only a few percent, at maybe 4% (90% confidence level). This number is conservative in some bands and too small (e.g., the strong lines around 30 keV) in others. NOTE: the aperture component's intrinsic ~10% uncertainty is not its systematic error since we directly measure it. However, the focused CXB's variance is ~100% or more and is NOT measured at the location of sources.

In our example, we would like a background spectrum for the region `src.reg` for the target of the observation. Assuming the spectra were extracted into the directory `event_cl/src/`, the call is simply:

```
IDL> nuskybgd_spec,dir,obsid,'src.reg','src','bgdspec','A','bgd'
```

where `src` and `bgdspec` indicate the directories where the source spectrum and bgdfitparam file is located, respectively. The last entry, `bgd`, gives the directory where the bgd reference files are located (previously created). The background spectrum prefixes 'bgd' to the source spectrum name, which is assumed to be named (in this case) `srcA_sr_g30.pha`. The keyword `specname` can input another name if you don't follow my convention. Background spectra for the 3 model components are also produced separately (prefixing bgdap, bgdfcxb, bgdinstr to the spectrum name). These spectra are simulated from the best-fit model as tailored for the source region using `fakeit` in XSPEC. By default, the background is simulated at 100x the exposure to ensure plenty of counts/channel, but you are free to adjust this via the `expfactor` keyword; see the source code for more options. The result:



On the left side the source plus bgd is shown; on the right the background has been subtracted (and is shown below it). The parameter file nuskybgd_spec uses can be specified directly (in the event you have several versions from multiple bgd fits) using the `paramfile` keyword (although note: for whatever idiotic reason you must give the full path or relative path from where you started IDL, which in this case would be `paramfile=dir+'/'+obsid+'/event_cl/bgp_try2A.dat'` perhaps).

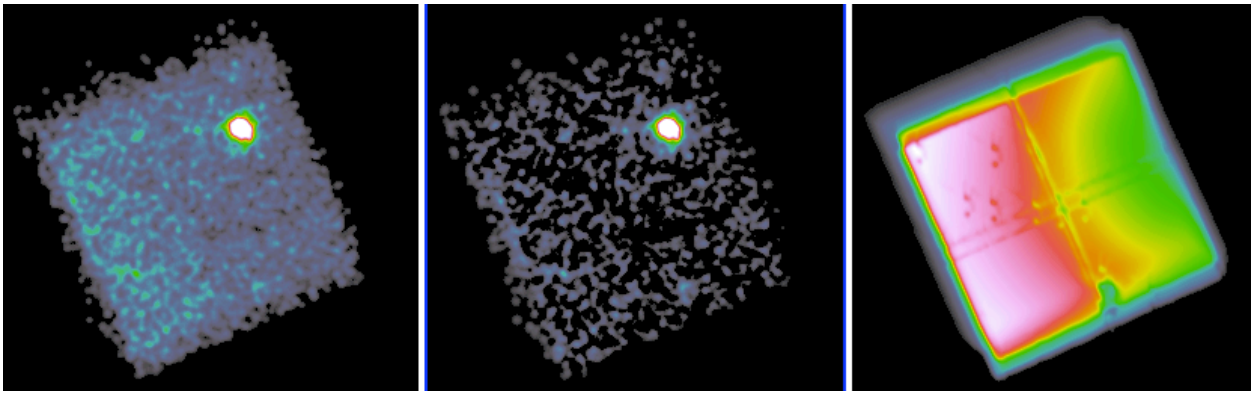
6) Making background (and background-subtracted) images

The call for this is very much the same as with `nuskybgd_spec`, although of course the order of input parameters is mixed up just to mess with you. You need a pre-existing image as input, which can be made via `mkimgs.py` or whatever – it's primarily used as a

lazy way to attach a header to the background image, but it also allows a background subtracted image to be part of the output. Make it so:

```
IDL> nuskybgd_image,dir,obsid,'imA3to20keV.fits',3,20,'A','bgd','bgdspec'
```

where `bgd` and `bgdspec` are directories with the reference bgd files and bgdfitparam file, respectively. The bgd and bgd-subtracted images are called `bgdimA3to20keV.fits` and `imA3to20keVbsub.fits`, respectively, and are written in `event_cl/`.



The original, bgd-subtracted, and background images (3-20 keV) are shown left to right, smoothed by a 7-pixel Gaussian to demonstrate the accuracy of the subtraction. The background is given in exact units of counts/pixel based on the model parameters. By default, the value of the focused CXB normalization is taken from the first value in the second line in `bgdfitparamsA.dat`. Since this value only applies to the first bgd region listed, it may not be wise to apply that to the entire image. In general, subtracting the fCXB component will result in empty regions with negative values since the CXB is 'clumpy' (i.e., made up of discrete sources). You can modify or eliminate the level of this or other component normalizations with keywords (see the source code). To derive the instrumental bgd component, a spectrum needs to be generated (but only once for a given bgdfitparam file). If you have several images to make and are extremely impatient, you can set the keyword `noremakeinstr`. You will need to remake this spectrum, however, if the bgdfitparam file changes.

7) Caveat Emptor

The modeling done by `nuskybgd` is generally accurate, but is currently quite experimental and is rather dependent on the user understanding what they are doing. Experimenting with placement and number of bgd regions is highly encouraged, since statistical fluctuations can easily bias fits to low signal-to-noise spectra. Questions are welcome, but don't make me reference this document.