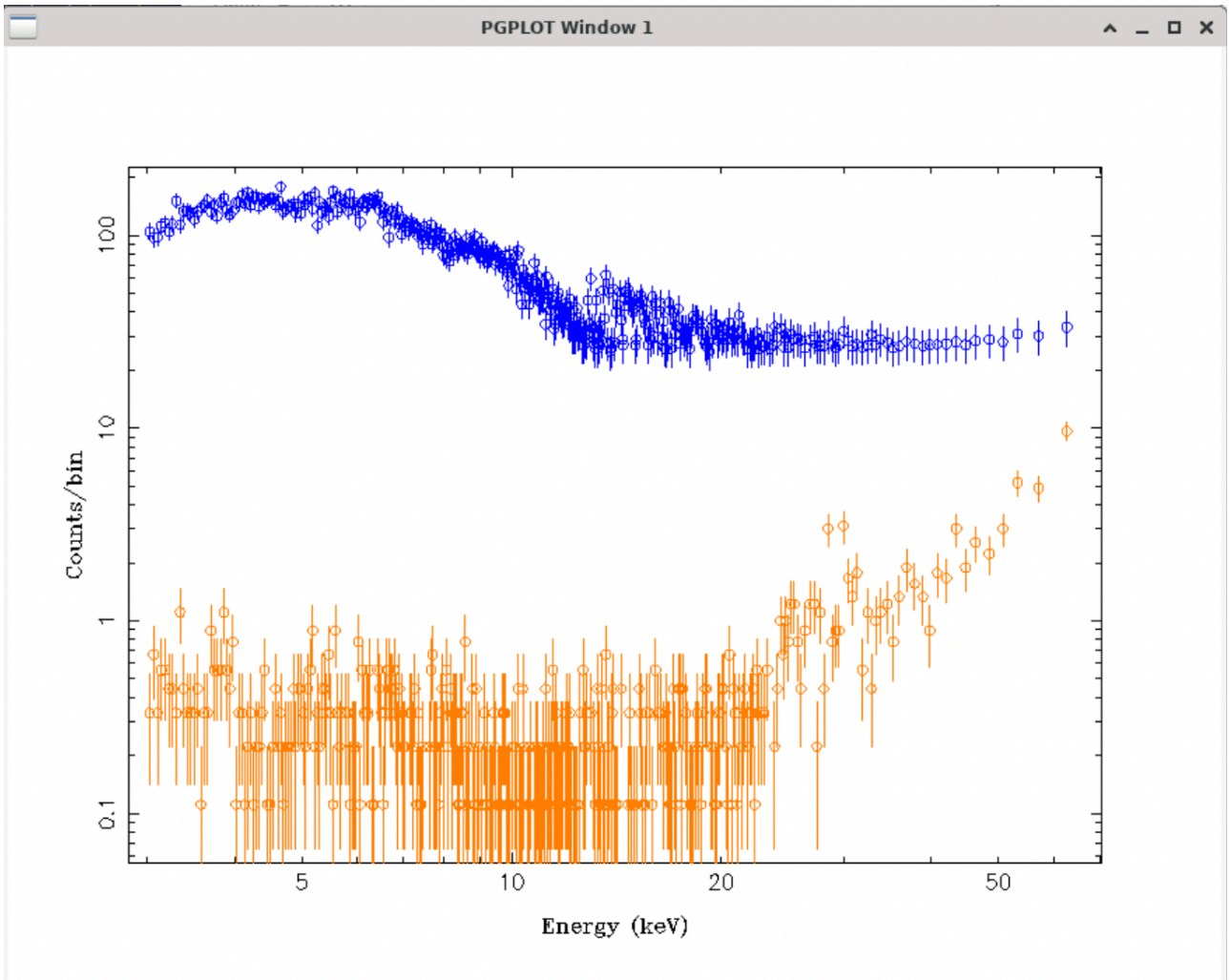
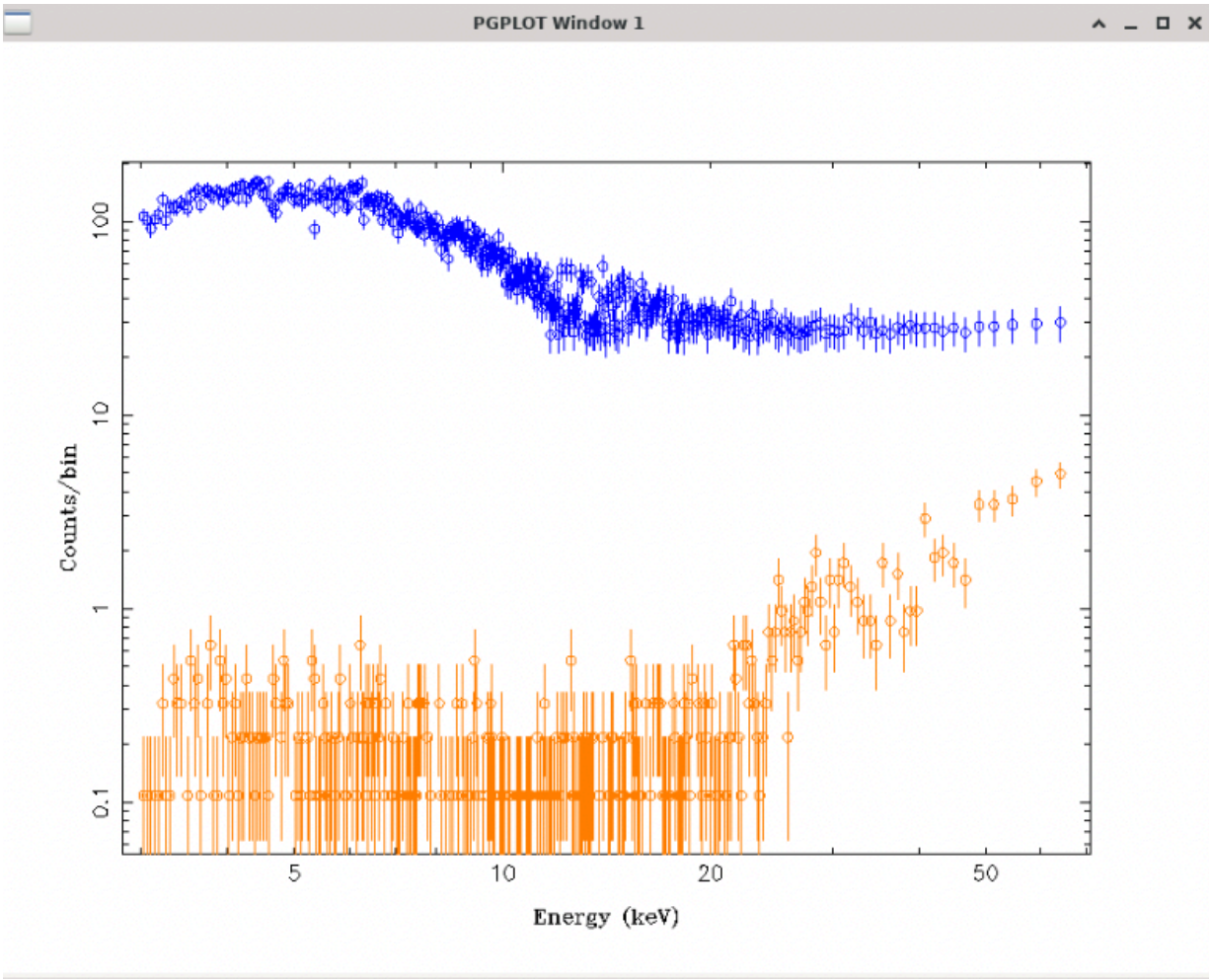
Pt.2:

All below show with following groupings:
group(index_number; min_sn=5, bounds=3, unit= “kev”);
notice_values(index_number,3,100; unit= “kev”);
xlog;
ylog;
plot_counts({index_number,index_number}; bkg={0,-1}, dcol={4,8});

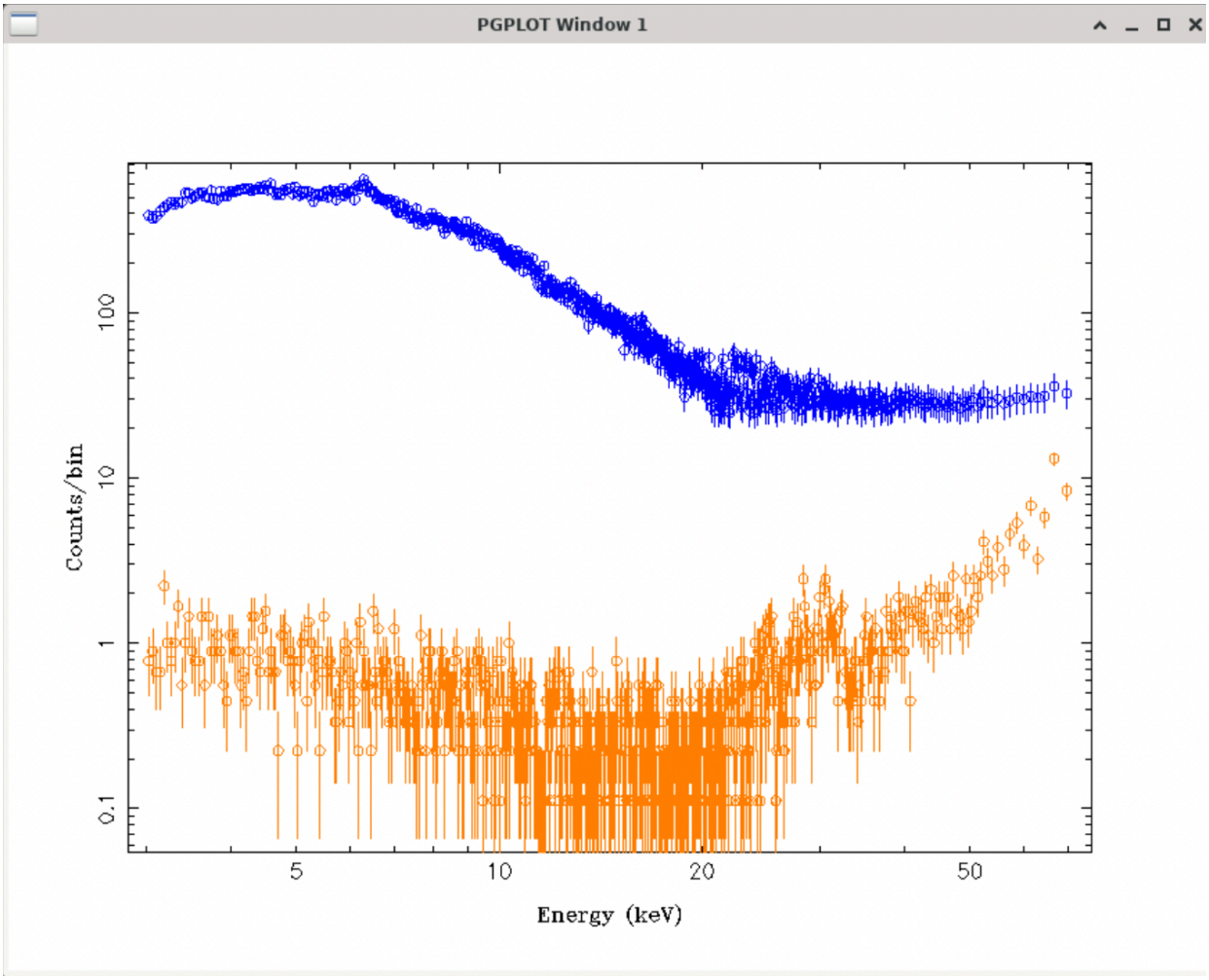
nu60160371002_A_sr.pha, source and background



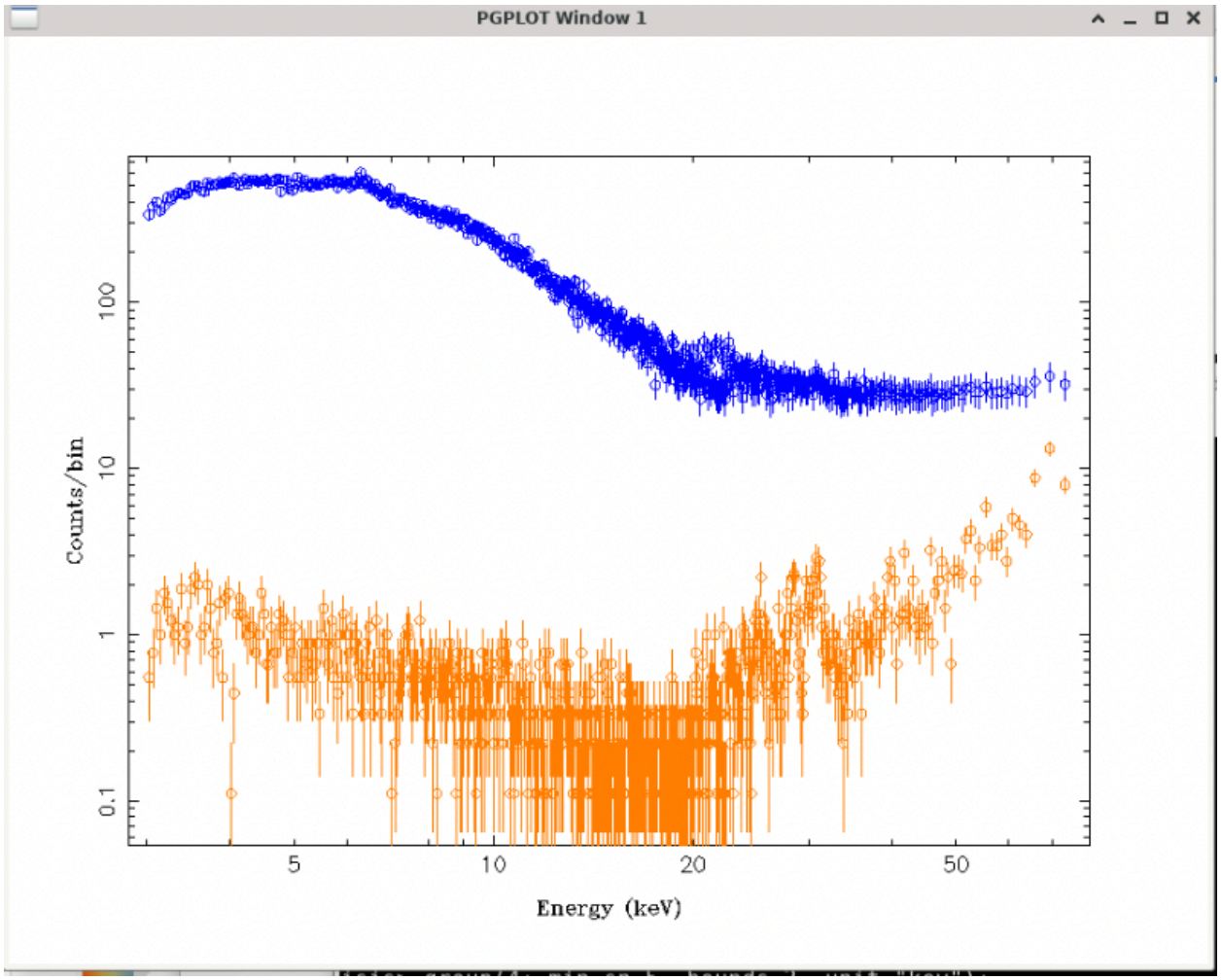
nu60160371002_B_sr.pha, source and background


PGPLOT Window 1
Counts/bin
100
10
1
0.1
5
10
20
50
Energy (keV)

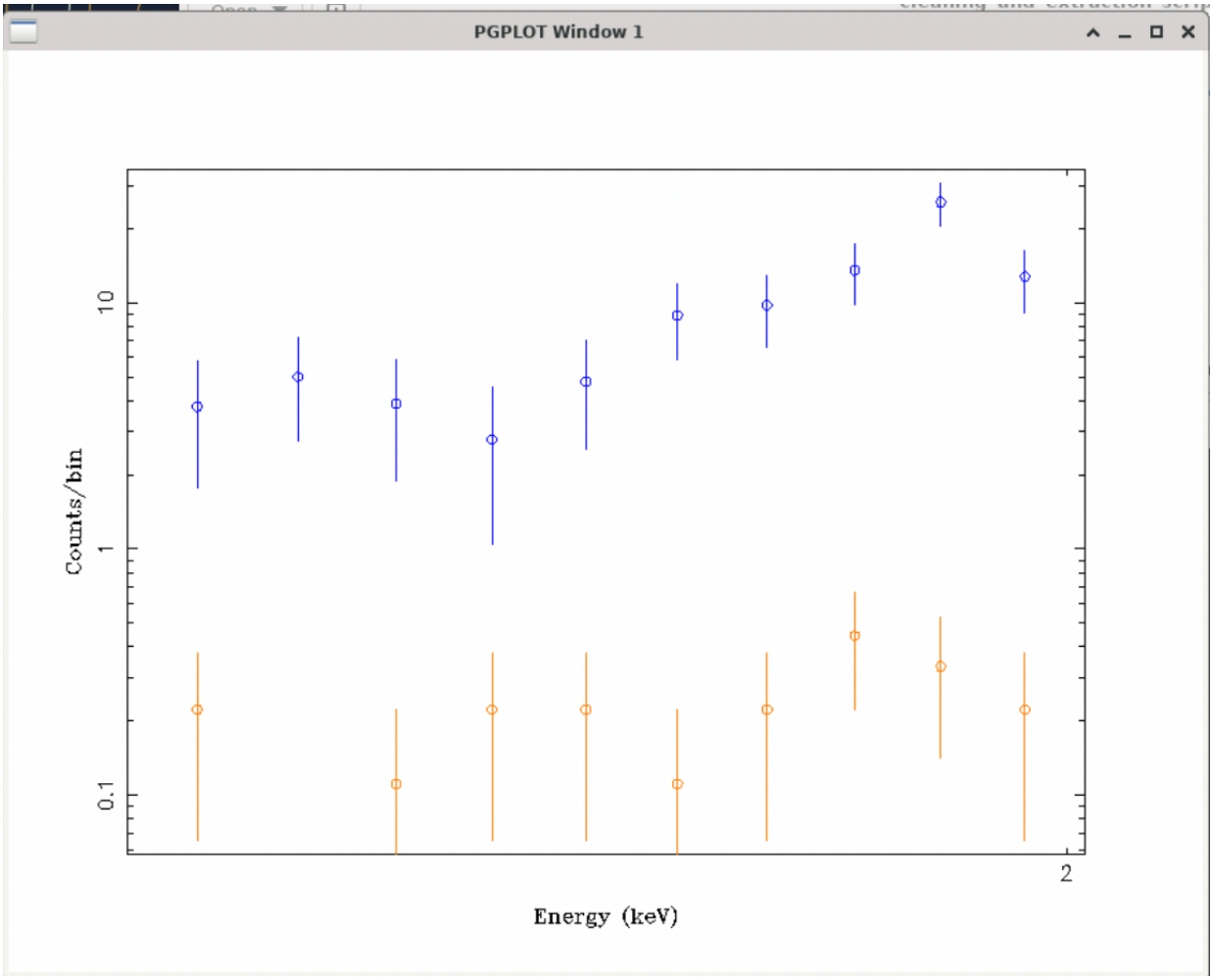
nu90501623002_A_sr.pha, source and background



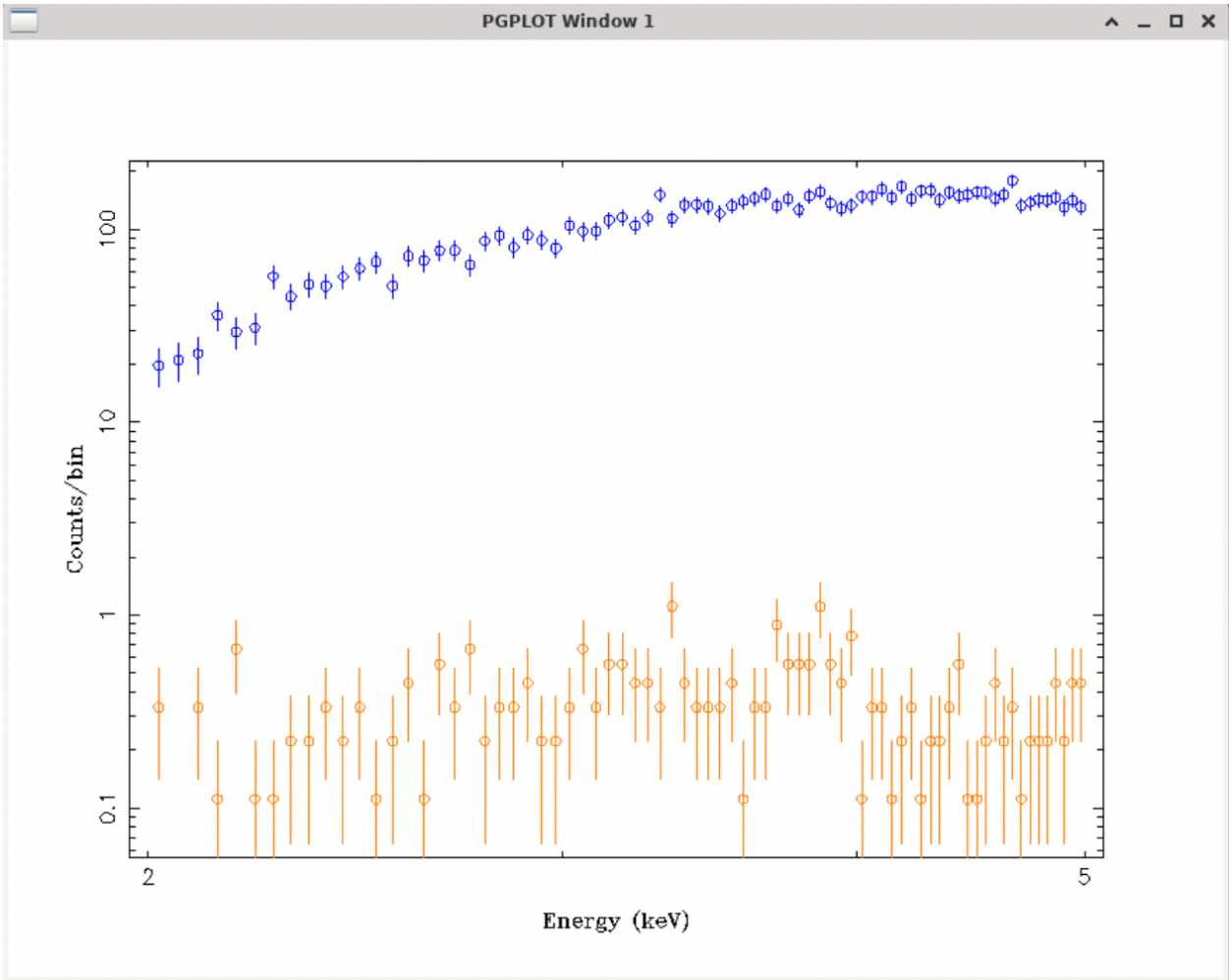
nu90501623002_B_sr.pha, source and background



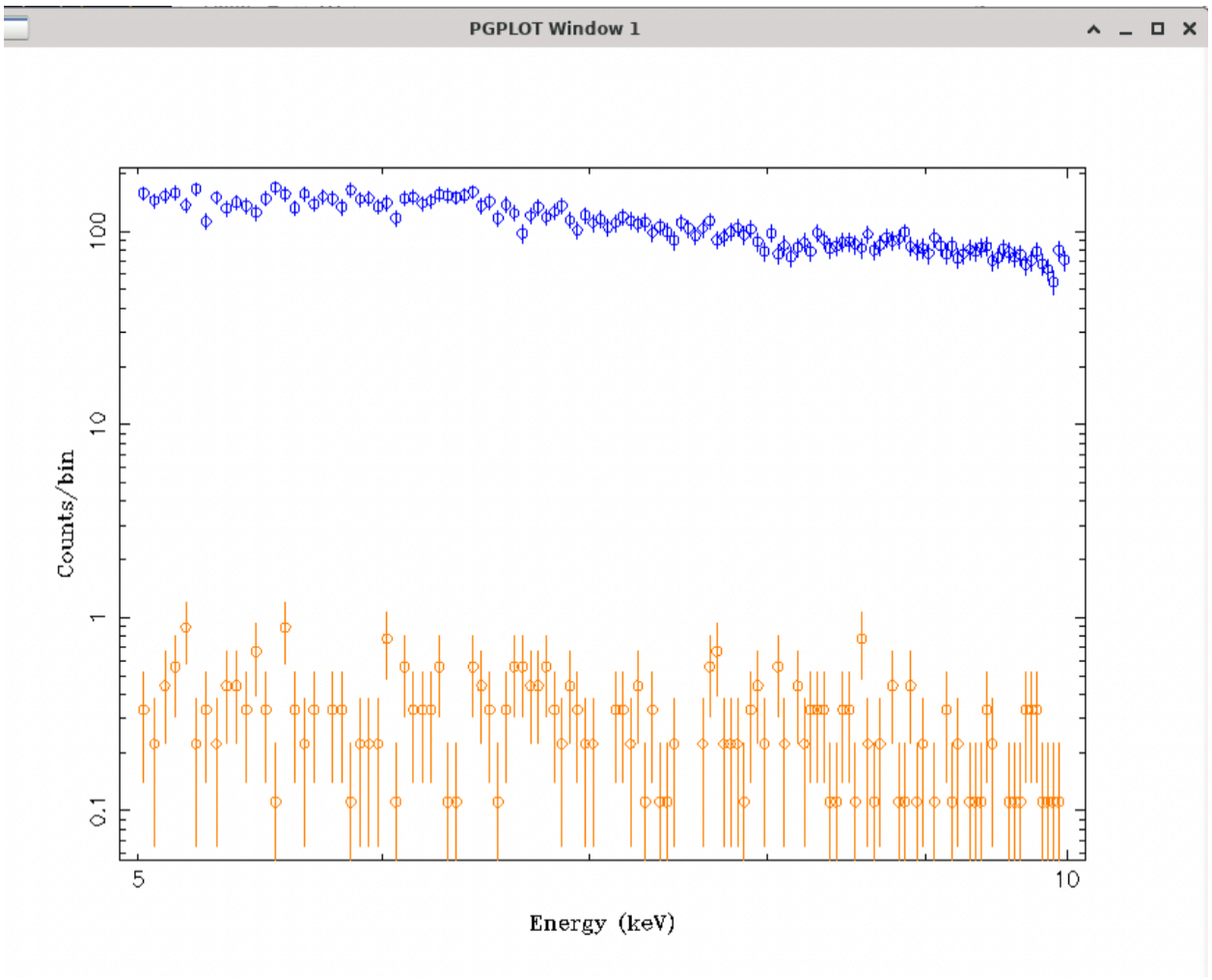
Bins from 0.5 - 2 keV, 2 - 5 keV, 5 - 10 keV
group(1; min_sn=5, bounds=3, unit="kev");
notice_values(1,0.5,2; unit="kev");
plot_counts({1,1}; bkg={0,-1}, dcol={4,8});
bins = 10 sr, 9 bkg (must use C-statistic or Bayesian)


PGPLOT Window 1
Counts/bin
10
1
0.1
2
Energy (keV)

group(1; min_sn=5, bounds=3, unit="kev");
notice_values(1,0.5,2; unit="kev");
plot_counts({1,1}; bkg={0,-1}, dcol={4,8});
bins =



group(1; min_sn=5, bounds=3, unit="kev");
notice_values(1,5,10; unit="kev");
plot_counts({1,1}; bkg={0,-1}, dcol={4,8});
bins =



These do not work as it is difficult to analyze the first bin (not Gaussian), and it would be better to have consistently analyzed bins. So, split the bins up in terms of equal counts per bin rather than characteristic energy ranges.

Run the following code to figure out bin intervals. Repeat for each data set (so twice; this uses the .pha files of each observation):

d = get_data_counts(index_number_here);
(d_kev_lo,d_kev_hi) = _A(d.bin_lo, d.bin_hi);

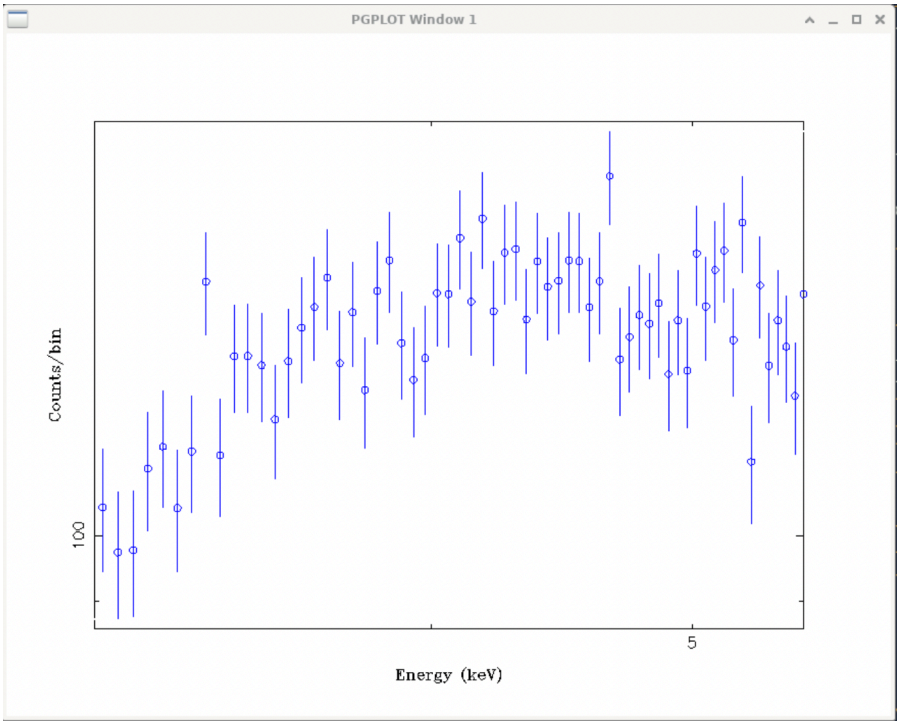
counts_kev = reverse (d.value); % d.value here returns the number of the bin, not the actual value of energy

cs = cumsum(counts_kev) / sum (counts_kev);
idx1 = wherefirst(cs >= 0.33);
idx2 = wherefirst(cs >= 0.66);
% print(idx1); % prints bin where up to 33% of data is (0 to idx1).
% print(idx2); % prints bin where up to 66% of data is (idx1 to idx2);
% Now, to find the keV range we want, we must look at the data inside the bins.
print(d_kev_lo[idx1]); % prints the leftmost edge of the bin
print(d_kev_hi[idx1]) % rightmost edge of the bin

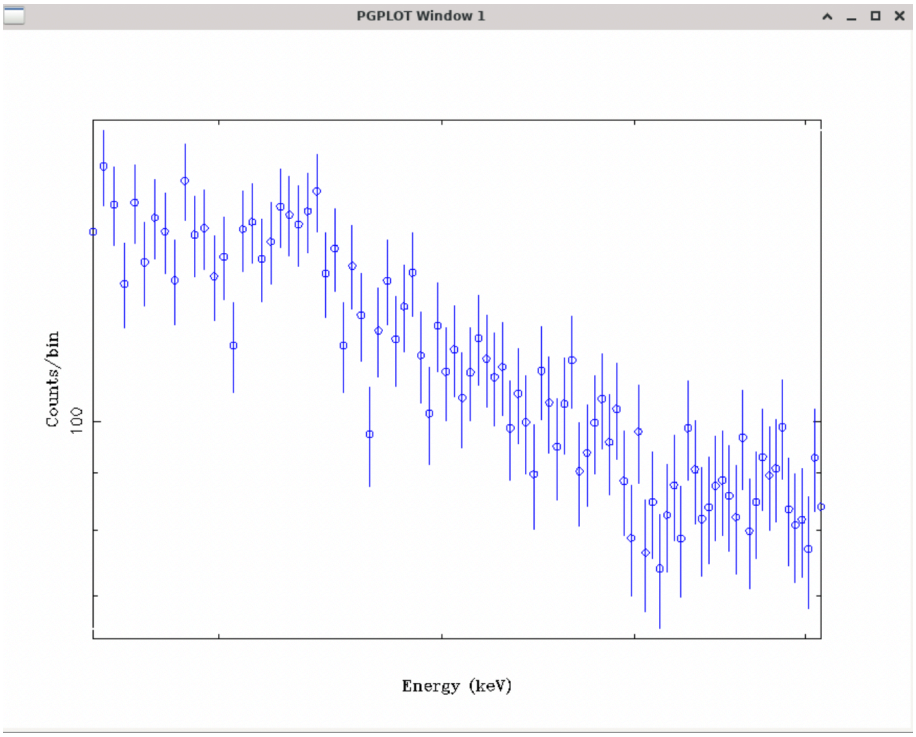
% Taking the average of the two values printed above, we find the increment, in keV, that split the first third of the data. Do it again for idx2 to get the increment for two-thirds of the data, then make your new three graphs (per observation), based on these increments. This will allow us to split the data up reasonably for further, more scrutinized analysis.

For nu60160371002_A, we got 5.50 and 9.10 keV as our increments. So, splitting up in three graphs,

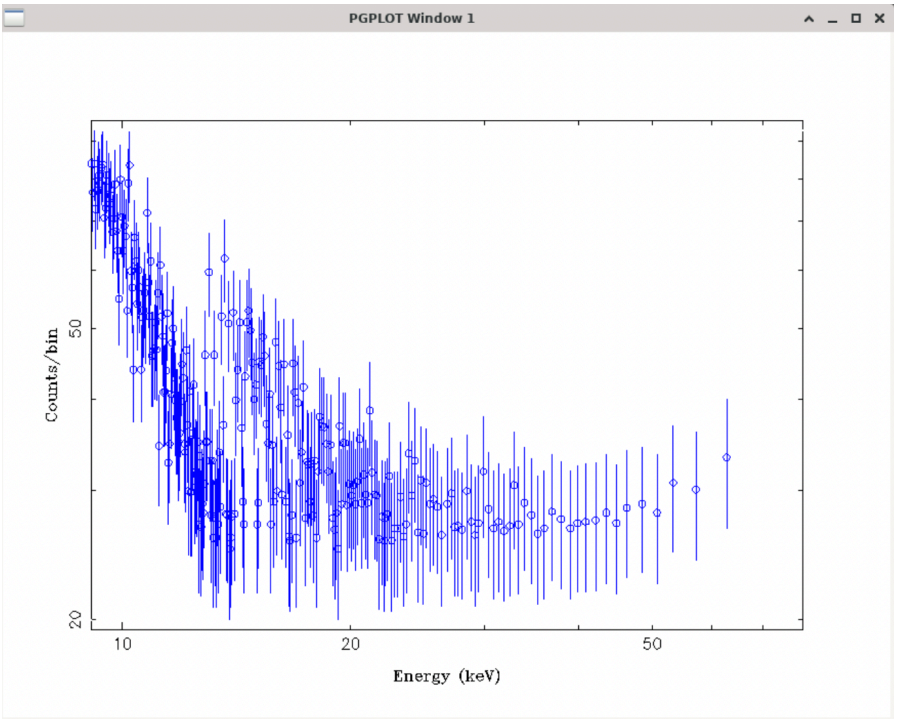
One from 3 - 5.50 keV,



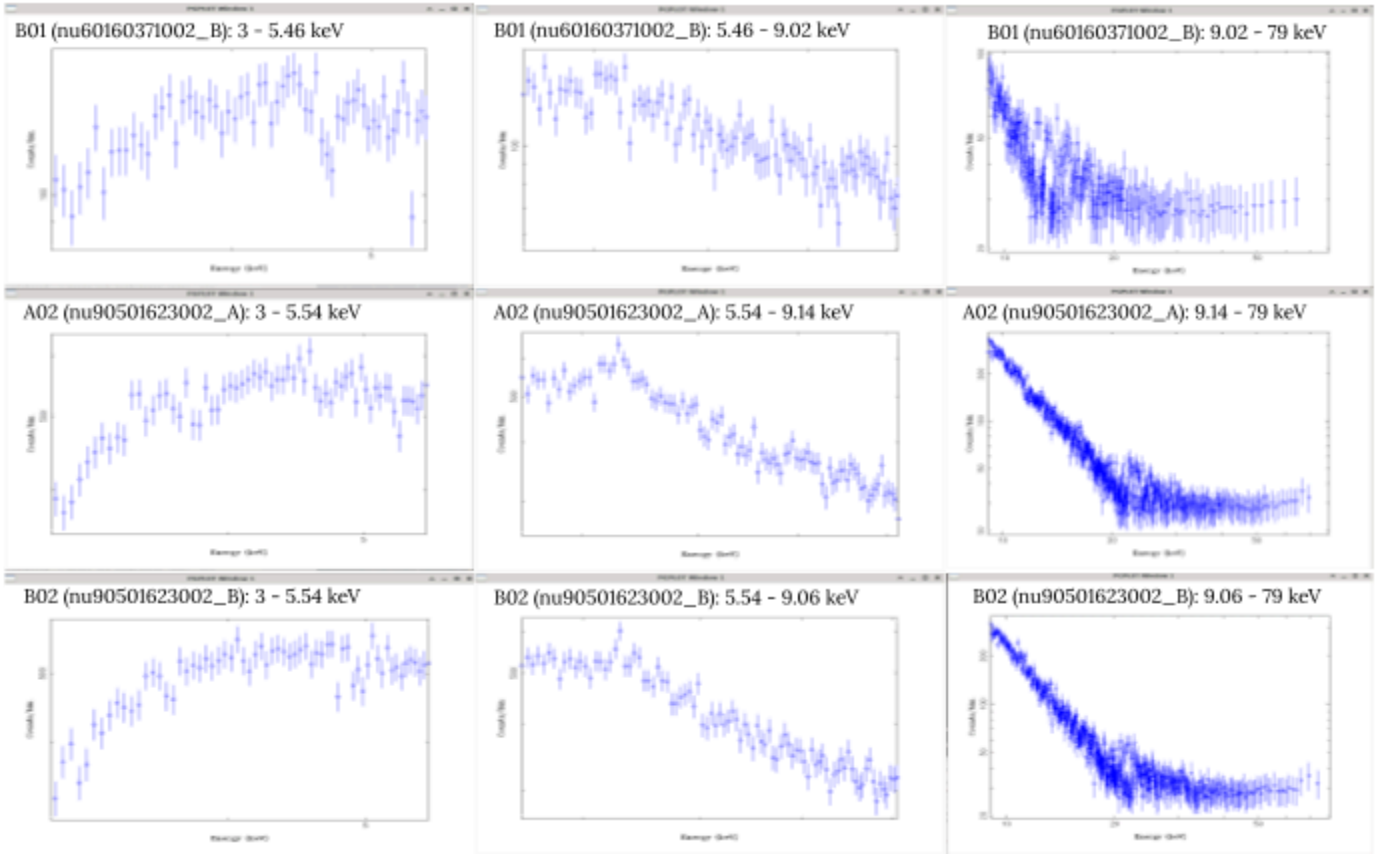
One from 5.50 - 9.10 keV,



One from 9.10 - 79 keV,



Repeating the process above for all observations and focal plane modules,



The next steps are located in my other document, cleaning_and_extraction_script.sl, marked starting with Pt.3.

Pt.4:

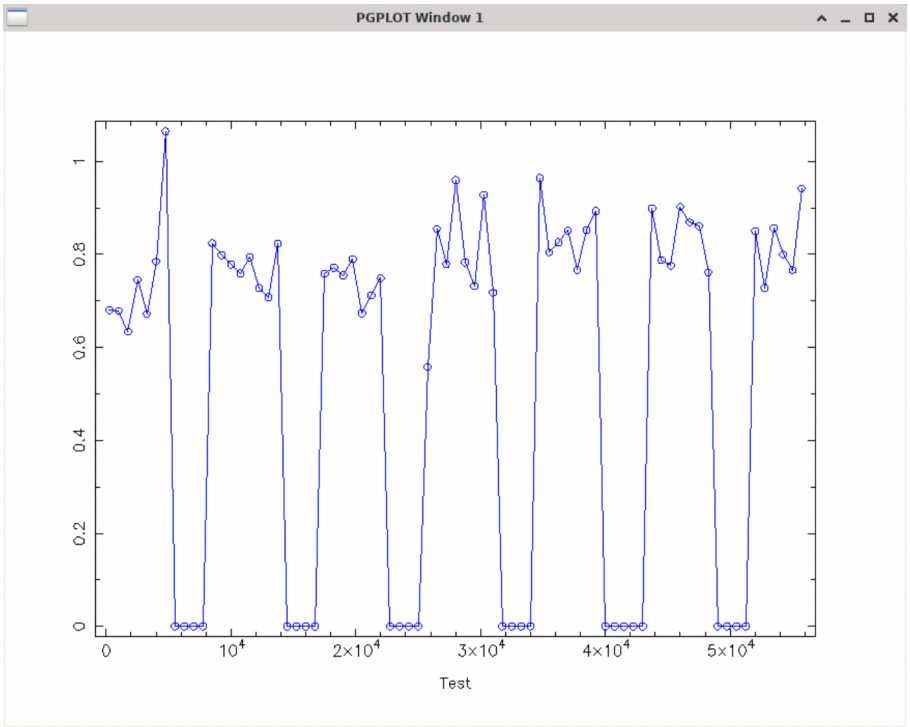
We have generated our 12 light curves and will analyze them for variability in ISIS. Running the code below shows an example of how to load a light curve into the software:

% Assigns data from each extension in the lc value to a vector defined under t, r, re
(t,r,re) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_35_98_A01_sr.lc", "time", "rate", "error");

% Test that it worked, you should see a non-rebinned graph (dark blue and not very easy to tell the difference between points).
plotxy(t,,r,re);

% Rebins the graph so that the variability of the data is more visible; lc is a structure
lc = sitar_rebin_rate(t,500,r); % 500 is a test value; 500 seconds for each bin.

% lc = sitar_rebin_rate(t,500,r;delgap); % To remove areas of nondata / concatenate the data
plotxy(lc.bin_lo+lc_bin_hi / 2, lc.rate);



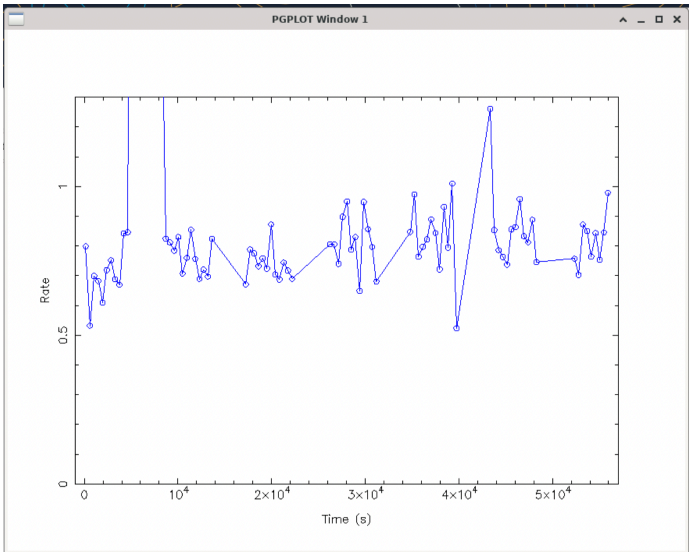
Now, I found the best parameters for the 60160371002 observations, to be:

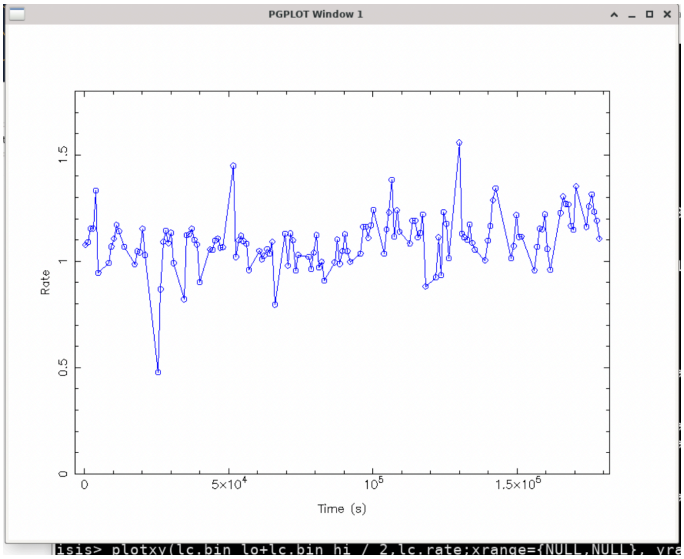
lc = sitar_rebin_rate(t,300,r;delgap);
plotxy(lc.bin_lo + lc.bin_hi / 2, lc.rate; xrange={NULL,NULL}, yrange={0,1.3});

And the best for 90501623002 to be:

lc = sitar_rebin_rate(t,600,r;delgap); % About double. Sanity check: There are about double the amount of counts in this observation than in the previous one.
plotxy(lc.bin_lo + lc.bin_hi / 2, lc.rate; xrange={NULL,NULL}, yrange={0,1.8});

Note, these parameters work best for looking at the data of FPMA and FPMB separately. Here is an example for each observation in the lowest energy bands in A01 (channels 35 - 98 for nu60160371002 at the top and channels 35 - 99 for nu90501623002 at the bottom, respectively; data is concatenated to exclude times where data was not collected):





Here is an energy band for 60160371002 after concatenating FPMA and FPMB data (this will provide us with better statistical accuracy, as both my mentor and I agree). The following code was executed to generate the following graphs for the first energy band of this observation (channels 35 - 98):

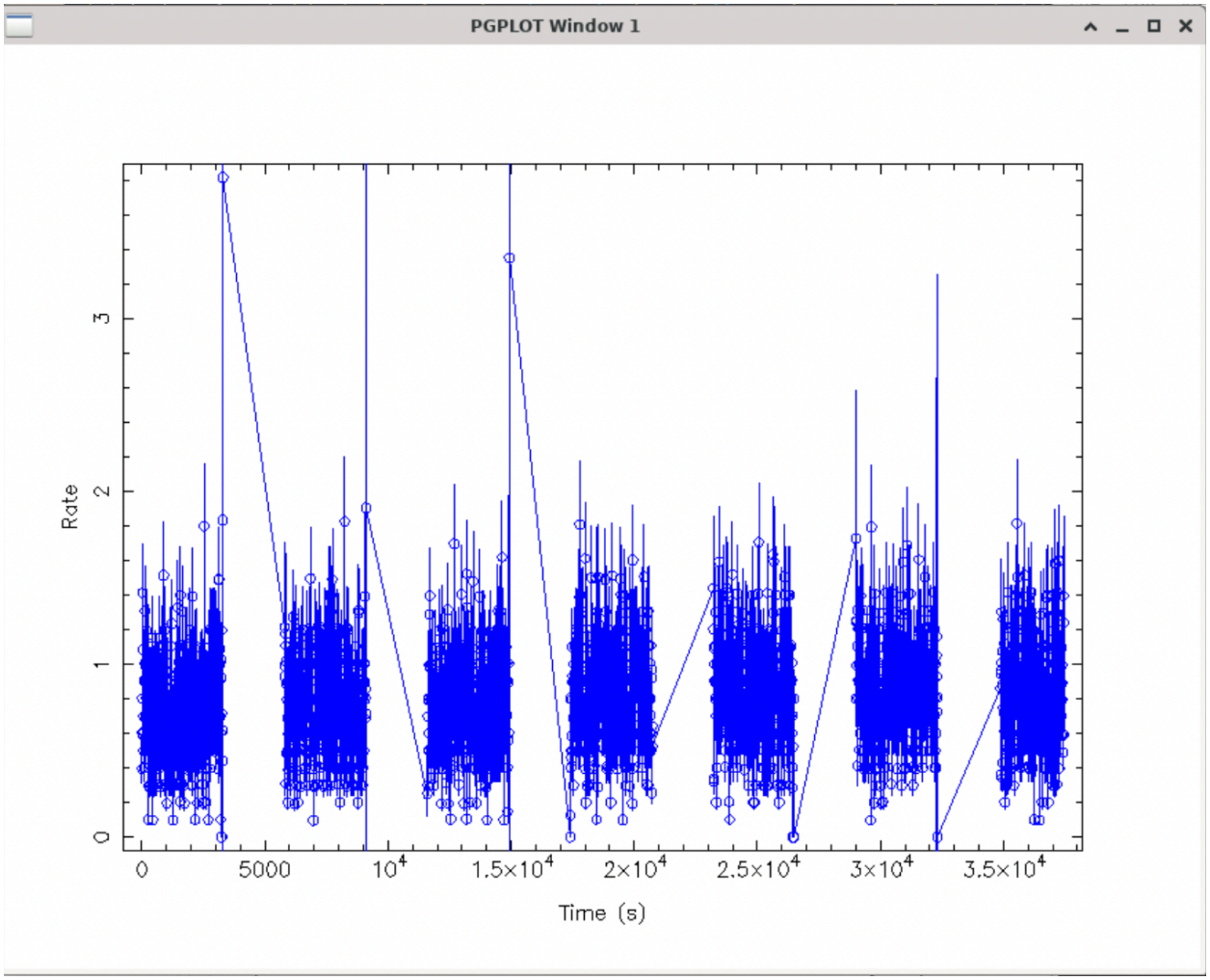
(t,ra,ra_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_35_98_A01_sr.lc", "time", "rate", "error");

(t,rb,rb_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2?nu_35_97_B01_sr.lc", "time", "rate", "error");

(t,rcon, rcon_e) = (t, (ra+rb) / 2, ((ra_e)^2+(rb_e)^2)^1/2);

ploxy(t,,,rcon,rcon_e,rcon_e);

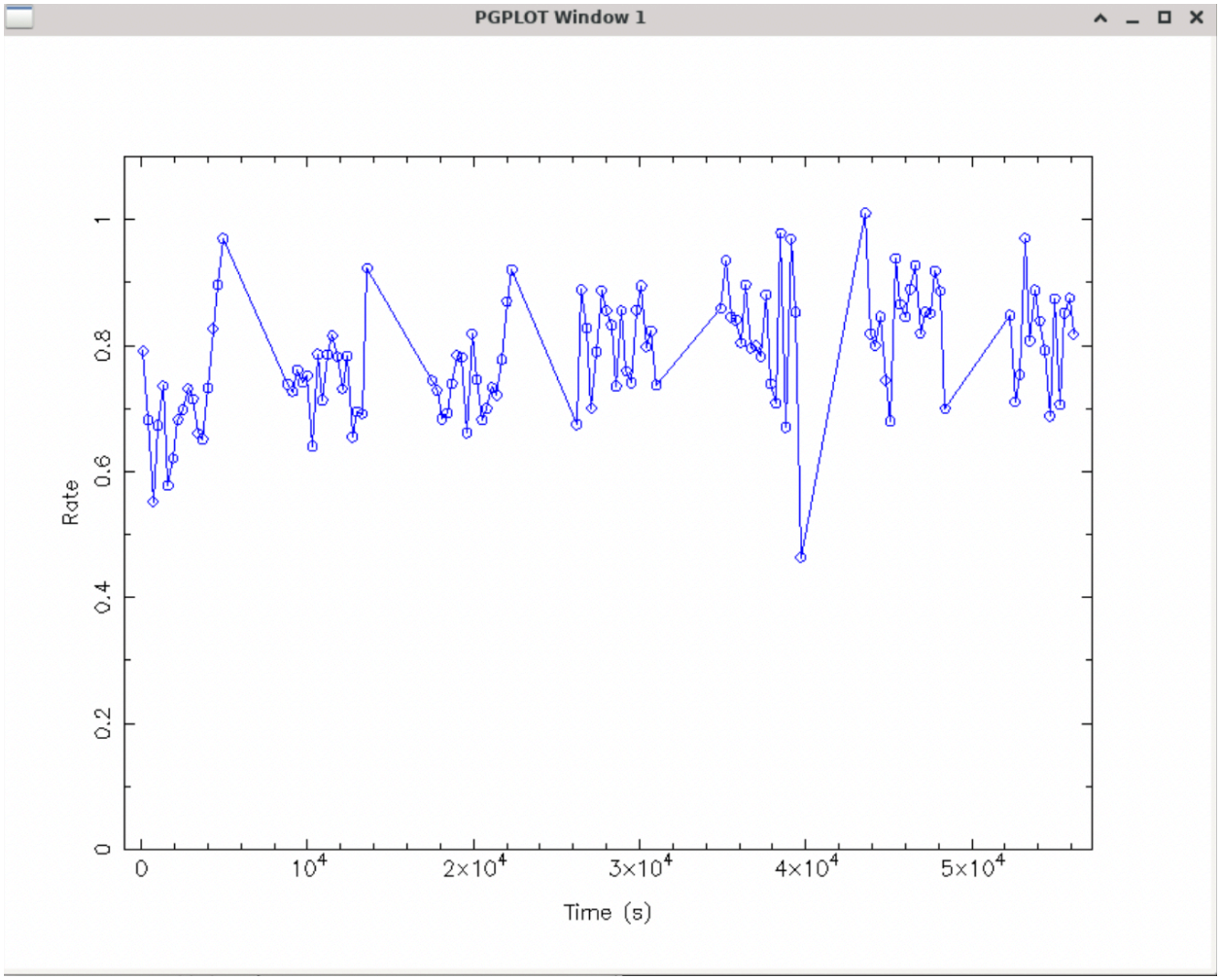
% Note: this code only works because, miraculously, the data for FPMA and FPMB in these exact channels has the same number of bins/counts. The code below accounts for increments where the FPMA and FPMB total counts are not identical, labeled by a **&&**.



The concatenated data makes it difficult to observe the object variability. Adjusting the parameters as follows

lc_con = sitar_rebin_rate(t,200,rcon;delgap);

plotxy(lc_con.bin_lo +lc_con.bin_hi / 2, lc_con.rate; xrange={NULL, NULL}, yrange{0,1.1});



The code above calculated the new concatenated rates and the error on those rates. Those calculations are represented via the following equations:

$$Concatenated\ Rate_{error} = \sqrt{(RateA)^2_{error} + (RateB)^2_{error}}$$

Where

$$Concatenated\ Rate = \frac{RateA + RateB}{2}$$

And all t values are kept the same.

Now, we must graph the colors (color-color diagrams) of each rate value and the respective bands, and compare them to the bins/counts, rates, and times of each relevant energy band to find trends in the data. For this analysis, we will start by looking at the 90501623002 as they are longer, noting distinct relationships between the variables, and then observing the same parameters in the 60160371002 data to see if it is consistent in both data sets.

For the first three increments of observation 90501623002, where,

x → 3 to 5.54 keV (Channels ~35 to ~99; FPMA + FPMB)
y → 5.54 to 9.14 keV (Channels ~98 to ~189; FPMA + FPMB)
z → 9.14 to 79 keV (Channels ~188 to ~1909; FPMA + FPMB)

The following code was run before running ISIS; the code was run in the directory above the directory containing the light curves for observation 90501623002. Unique variables are used to avoid ambiguous use of variables, although it is possible to use the same variable for many of these commands as they are overwritten after the command is executed:

```
&&
 setxspec 34
 caldb
 caldbinit
 isis -g
 isis > add_to_isis_load_path("/man1/PACKAGES/isisscripts/share");
 isis >require("isisscripts");

% A and B for x.
(xta,xra,xra_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_
35_99_A01_sr.lc", "time", "rate", "error");
(xtb,xrb,xrb_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_
35_99_B01_sr.lc", "time", "rate", "error");

xa = sitar_rebin_rate(xta,600,xra,xra_e;delgap); % make sure to do delgap on both
or neither only; bin sizes determined earlier
xb = sitar_rebin_rate(xtb,600,xrb,xrb_e;delgap, usr_bin=xa);

xcon = struct{rate, err};
xcon.rate = (xa.rate + xb.rate) / 2;
xcon.err = sqrt(xa.err^2+xb.err^2);

% A and B for y
(yta,yra,yra_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_
98_189_A01_sr.lc", "time", "rate", "error");
```

(ytb,yrb, yrb_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_98_187_B01_sr.lc", "time", "rate", "error");

ya = sitar_rebin_rate(yta,600,yra,yra_e;delgap); % make sure to do delgap on both or neither only; bin sizes determined earlier
yb = sitar_rebin_rate(ytb,600,yrb,yrb_e;delgap, usr_bin=ya);

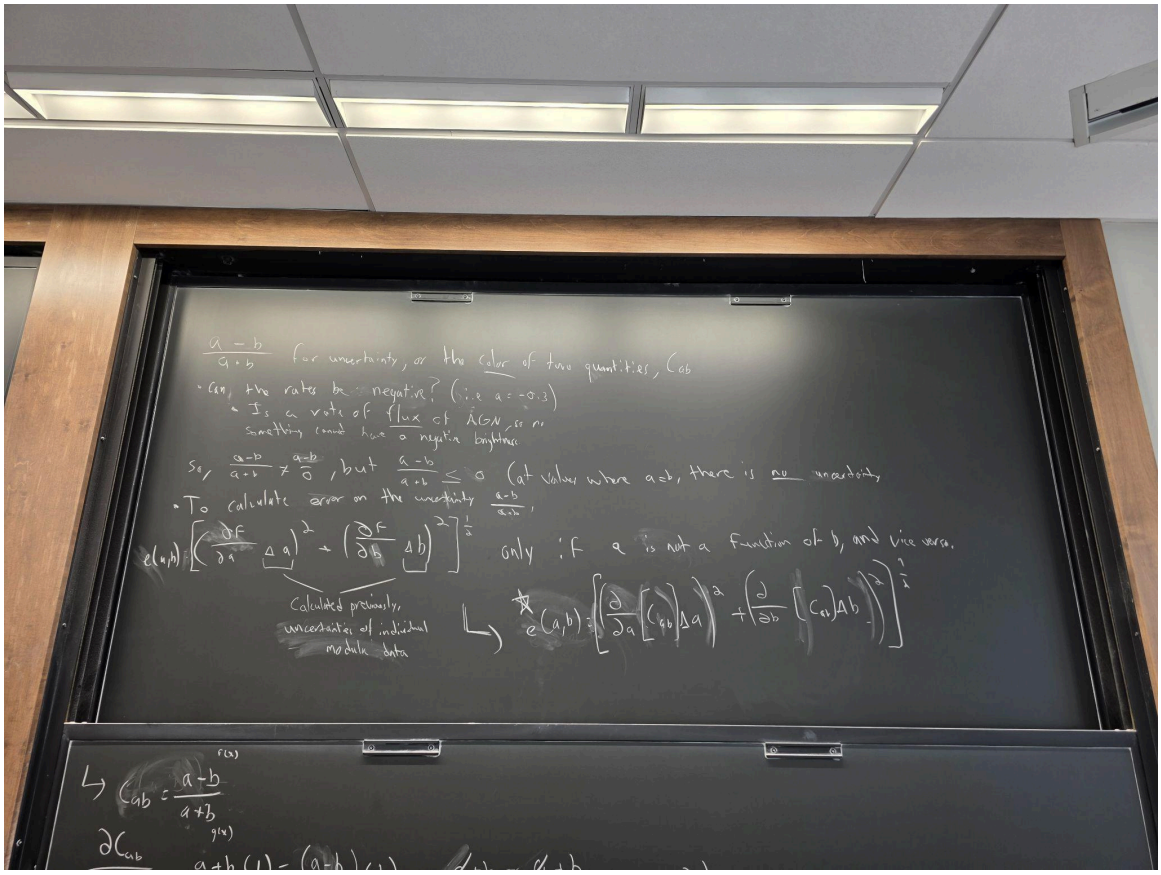
ycon = struct{rate, err};
ycon.rate = (ya.rate + yb.rate) / 2;
ycon.err = sqrt(ya.err^2+yb.err^2);

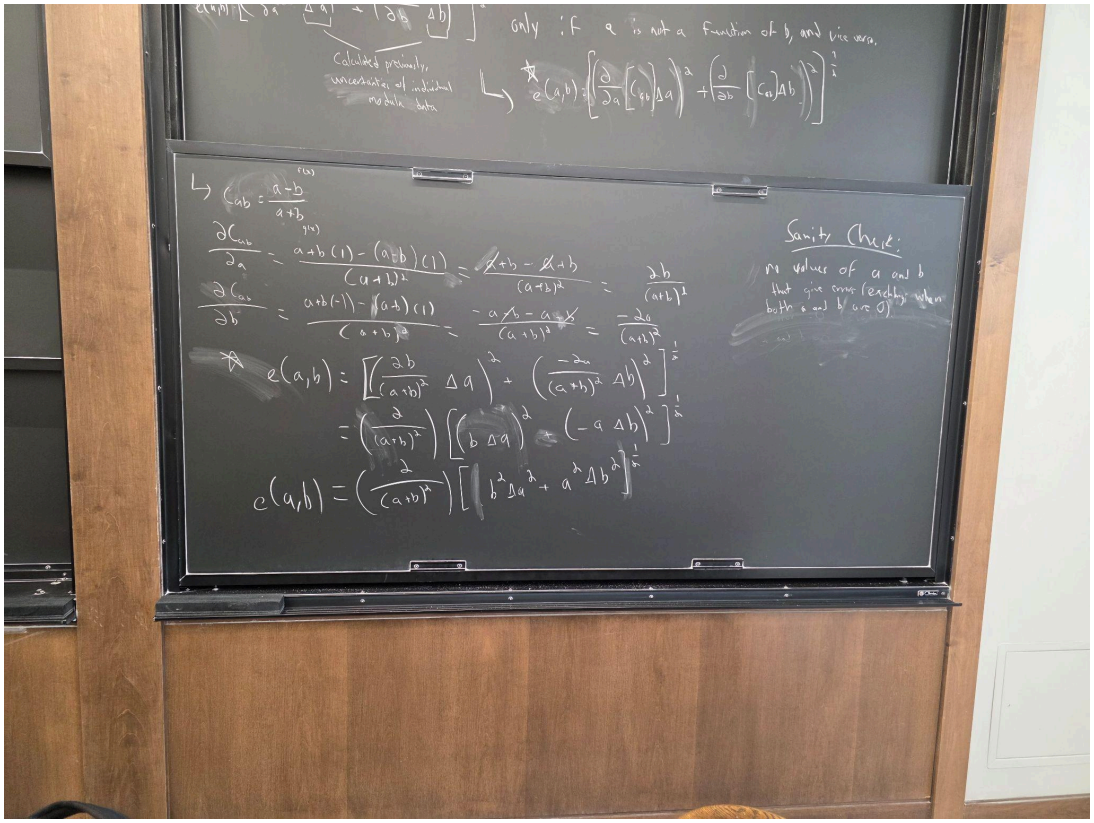
% A and B for z
(zta,zra,zra_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_188_1909_A01_sr.lc", "time", "rate", "error");
(ztb,zrb, zrb_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/90501623002/event_cl2/nu_186_1909_B01_sr.lc", "time", "rate", "error");

za = sitar_rebin_rate(zta,600,zra,zra_e;delgap); % make sure to do delgap on both or neither only; bin sizes determined earlier
zb = sitar_rebin_rate(ztb,600,zrb,zrb_e;delgap, usr_bin=za);

zcon = struct{rate, err};
zcon.rate = (za.rate + zb.rate) / 2;
zcon.err = sqrt(za.err^2+zb.err^2);

So, we now have three structures representing the concatenated data of FPMA and FPMB in our three relevant energy bands. To write the colors, according to my mentor, use the formulas $\frac{a-b}{a+b}$ for each pair of the three concatenated energy band data sets, which will represent different physical qualities and are written in order after c in the code (so cxy represents $\frac{x-y}{x+y}$, cyx represents $\frac{y-x}{y+x}$). It is important to note that the error for these calculated quantities is as follows:


$\frac{a-b}{a+b}$ for uncertainty, or the color of two quantities, C_{ab}
• Can the rates be negative? (i.e $a = -0.3$)
• Is a rate of flux of AGN is a something cannot have a negative brightness
So, $\frac{a-b}{a+b} \neq \frac{a-b}{0}$, but $\frac{a-b}{a+b} \leq 0$ (at values where $a \neq b$, there is no uncertainty
• To calculate error on the uncertainty $\frac{a-b}{a+b}$,
$e(a,b) = \left[\left(\frac{\partial f}{\partial a}\Delta a\right)^2 + \left(\frac{\partial f}{\partial b}\Delta b\right)^2\right]^{\frac{1}{2}}$ only if a is not a function of b, and vice versa.
Calculated previously, uncertainties of individual module data
$\hookrightarrow$ $e(a,b) = \left[\left(\frac{\partial}{\partial a}\left[\left(\frac{a-b}{a+b}\right)\Delta a\right]\right)^2 + \left(\frac{\partial}{\partial b}\left[\left(\frac{a-b}{a+b}\right)\Delta b\right]\right)^2\right]^{\frac{1}{2}}$
$\hookrightarrow C_{ab} = \frac{a-b}{a+b}$
$f(x)$
$g(x)$
$\frac{\partial C_{ab}}{\partial a}$


only if a is not a function of b, and vice versa.
Calculated previously, uncertainties of individual module data
$\hookrightarrow$ $e(a,b) = \left[\left(\frac{\partial}{\partial a}\left[\left(\frac{a-b}{a+b}\right)\Delta a\right]\right)^2 + \left(\frac{\partial}{\partial b}\left[\left(\frac{a-b}{a+b}\right)\Delta b\right]\right)^2\right]^{\frac{1}{2}}$
$\hookrightarrow C_{ab} = \frac{a-b}{a+b}$
$f(x)$
$g(x)$
$\frac{\partial C_{ab}}{\partial a} = \frac{a+b(1) - (a-b)(1)}{(a+b)^2} = \frac{a+b-a+b}{(a+b)^2} = \frac{2b}{(a+b)^2}$
$\frac{\partial C_{ab}}{\partial b} = \frac{a+b(-1) - (a-b)(1)}{(a+b)^2} = \frac{-a-b-a+b}{(a+b)^2} = \frac{-2a}{(a+b)^2}$
Sanity Check:
no values of a and b that give same (exactly) when both a and b are 0)
$e(a,b) = \left[\left(\frac{2b}{(a+b)^2}\Delta a\right)^2 + \left(\frac{-2a}{(a+b)^2}\Delta b\right)^2\right]^{\frac{1}{2}}$
$= \left(\frac{2}{(a+b)^2}\right)\left[\left(b\Delta a\right)^2 + \left(-a\Delta b\right)^2\right]^{\frac{1}{2}}$
$e(a,b) = \left(\frac{2}{(a+b)^2}\right)\left[b^2\Delta a^2 + a^2\Delta b^2\right]^{\frac{1}{2}}$

cxy = struct{cons,err};
cxy.cons = (xcon.rate-ycon.rate)/(xcon.rate+ycon.rate);
cxy.err =
2/(xcon.rate+ycon.rate)^2*(ycon.rate^2*xcon.err^2+xcon.rate^2*ycon.err^2)^1/2;

cyx = struct{cons,err};
cyx.cons = (ycon.rate-xcon.rate)/(ycon.rate+xcon.rate);
cyx.err =
2/(ycon.rate+xcon.rate)^2*(xcon.rate^2*ycon.err^2+ycon.rate^2*xcon.err^2)^1/2;

cyz = struct{cons,err};
cyz.cons = (ycon.rate-zcon.rate)/(ycon.rate+zcon.rate);
cyz.err =
2/(ycon.rate+zcon.rate)^2*(zcon.rate^2*ycon.err^2+ycon.rate^2*zcon.err^2)^1/2;

czy = struct{cons,err};
czy.cons = (zcon.rate-ycon.rate)/(zcon.rate+ycon.rate);
czy.err =
2/(zcon.rate+ycon.rate)^2*(ycon.rate^2*zcon.err^2+zcon.rate^2*ycon.err^2)^1/2;

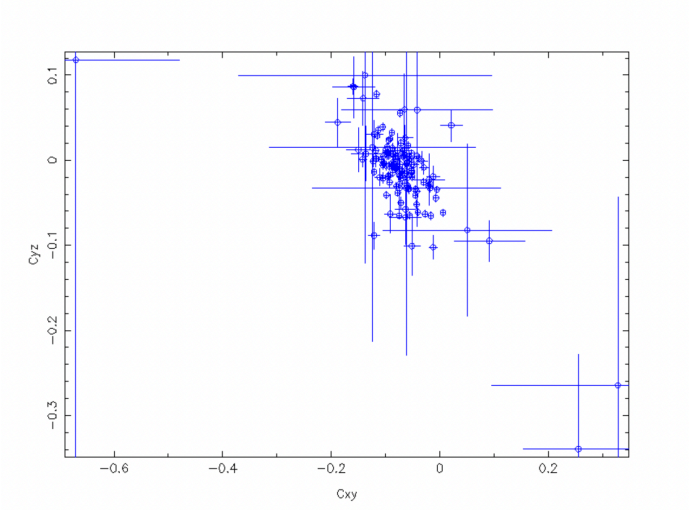
cxz = struct{cons,err};
cxz.cons = (xcon.rate-zcon.rate)/(xcon.rate+zcon.rate);
cxz.err =
2/(xcon.rate+zcon.rate)^2*(zcon.rate^2*xcon.err^2+xcon.rate^2*zcon.err^2)^1/2;

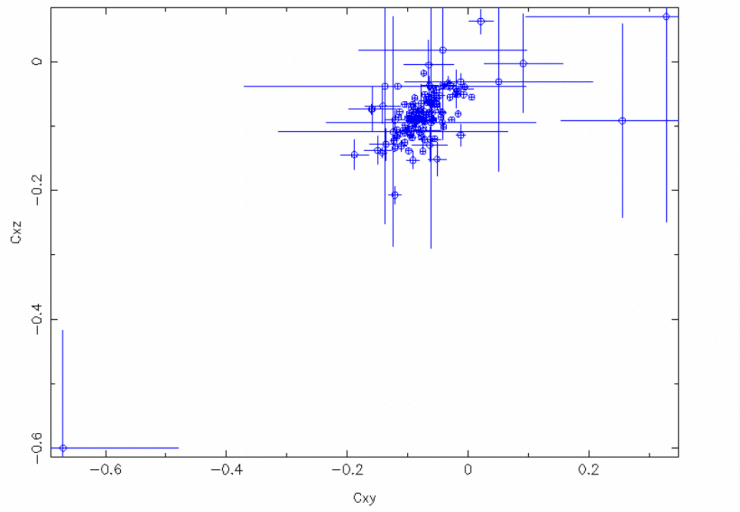
czx = struct{cons,err};
czx.cons = (zcon.rate-xcon.rate)/(zcon.rate+xcon.rate);
czx.err =
2/(zcon.rate+xcon.rate)^2*(xcon.rate^2*zcon.err^2+zcon.rate^2*xcon.err^2)^1/2;

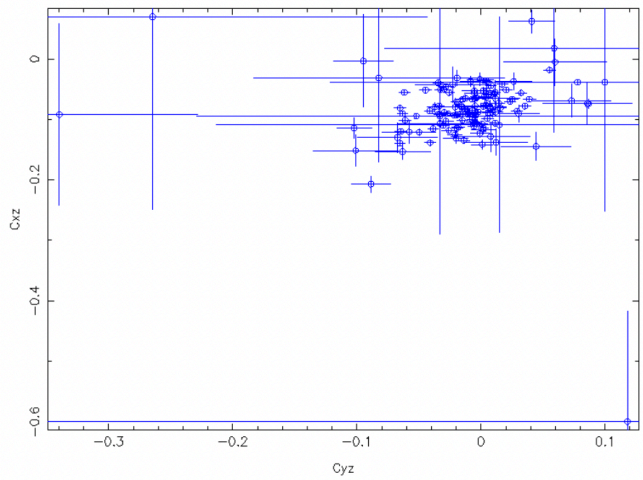
Plotting the colors in all possible combinations of the bolded code, following the code below:

connect_points(0); % For better visibility
plotxy(cxy.cons,cxy.err,cxy.err,cyz.cons,cyz.err,cyz.err);
plotxy(cxy.cons,cxy.err,cxy.err,cxz.cons,cxz.err,cxz.err);
plotxy(cyz.cons,cyz.err,cyz.err,cxz.cons,cxz.err,cxz.err);

The non bolded code are the other possible combinations that may be used in later analysis.


0.1
0
−0.1
−0.2
−0.3
Cyz
−0.6
−0.4
−0.2
0
0.2
Cxy


0
−0.2
−0.4
−0.6
Cxz
−0.6
−0.4
−0.2
0
0.2
Cxy


0
−0.2
−0.4
−0.6
Cxz
−0.3
−0.2
−0.1
0
0.1
Cyz

Now, let's do the same with the 60160371002 observation, achieving the same three graphs for the colors. The code above was replicated and slightly modified to do this. Note that our variables have a "b_" extension in the front. This is to indicate variables related to the shorter observation, 60160371002:

The following code was run before running ISIS; the code was run in the directory above the directory containing the light curves for observation 60160371002:

```
setxspec 34
caldb
caldbinit
isis -g
isis > add_to_isis_load_path("/man1/PACKAGES/isisscripts/share");
isis >require("isisscripts");

% A and B for x
(b_xta,b_xra,b_xra_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_3
5_98_A01_sr.lc", "time", "rate", "error");
(b_xtb,b_xrb, b_xrb_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_3
5_97_B01_sr.lc", "time", "rate", "error");

b_xa = sitar_rebin_rate(b_xta,300,b_xra,b_xra_e;delgap); % make sure to do
delgap on both or neither only; bin sizes determined earlier
b_xb = sitar_rebin_rate(b_xtb,300,b_xrb,b_xrb_e;delgap, usr_bin=b_xa);

b_xcon = struct{rate, err};
b_xcon.rate = (b_xa.rate + b_xb.rate) / 2;
b_xcon.err = sqrt(b_xa.err^2+b_xb.err^2);

% A and B for y
(b_yta,b_yra,b_yra_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_9
7_188_A01_sr.lc", "time", "rate", "error");
(b_ytb,b_yrb, b_yrb_e) =
fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_9
6_186_B01_sr.lc", "time", "rate", "error");
```

b_ya = sitar_rebin_rate(b_yta,300,b_yra,b_yra_e;delgap); % make sure to do delgap on both or neither only; bin sizes determined earlier
b_yb = sitar_rebin_rate(b_ytb,300,b_yrb,b_yrb_e;delgap, usr_bin=b_ya);

b_ycon = struct{rate, err};
b_ycon.rate = (b_ya.rate + b_yb.rate) / 2;
b_ycon.err = sqrt(b_ya.err^2+b_yb.err^2);

% A and B for z
(b_zta,b_zra,b_zra_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_187_1909_A01_sr.lc", "time", "rate", "error");
(b_ztb,b_zrb, b_zrb_e) = fits_read_col("/man1/gisaac/NGC_2992/Extracted_Data/60160371002/event_cl2/nu_185_1909_B01_sr.lc", "time", "rate", "error");

b_za = sitar_rebin_rate(b_zta,300,b_zra,b_zra_e;delgap); % make sure to do delgap on both or neither only; bin sizes determined earlier
b_zb = sitar_rebin_rate(b_ztb,300,b_zrb,b_zrb_e;delgap, usr_bin=b_za);

b_zcon = struct{rate, err};
b_zcon.rate = (b_za.rate + b_zb.rate) / 2;
b_zcon.err = sqrt(b_za.err^2+b_zb.err^2);

% Colors
b_cxy = struct{cons,err};
b_cxy.cons = (b_xcon.rate-b_ycon.rate)/(b_xcon.rate+b_ycon.rate);
b_cxy.err = 2/(b_xcon.rate+b_ycon.rate)^2*(b_ycon.rate^2*b_xcon.err^2+b_xcon.rate^2*b_ycon.err^2)^1/2;

b_cyx = struct{cons,err};
b_cyx.cons = (b_ycon.rate-b_xcon.rate)/(b_ycon.rate+b_xcon.rate);
b_cyx.err = 2/(b_ycon.rate+b_xcon.rate)^2*(b_xcon.rate^2*b_ycon.err^2+b_ycon.rate^2*b_xcon.err^2)^1/2;

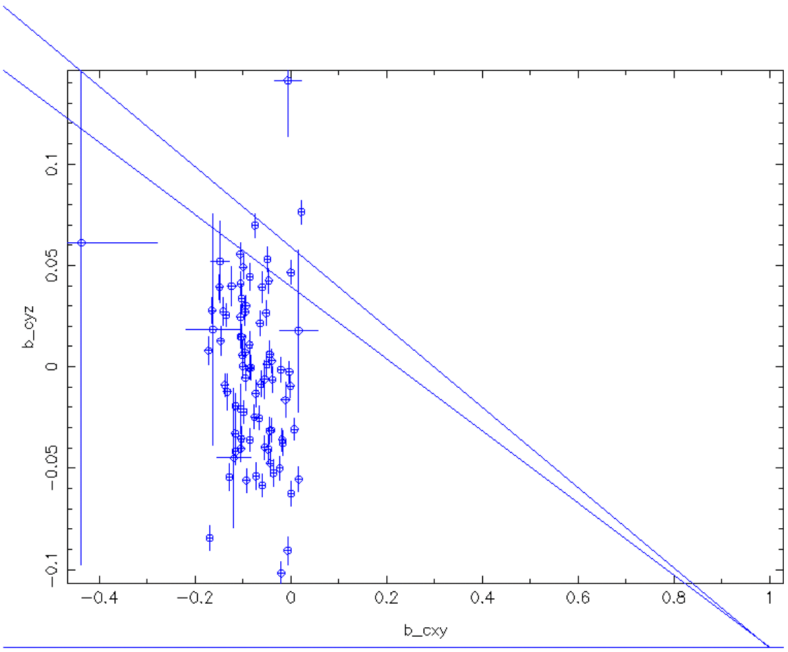
b_cyz = struct{cons,err};
b_cyz.cons = (b_ycon.rate-b_zcon.rate)/(b_ycon.rate+b_zcon.rate);

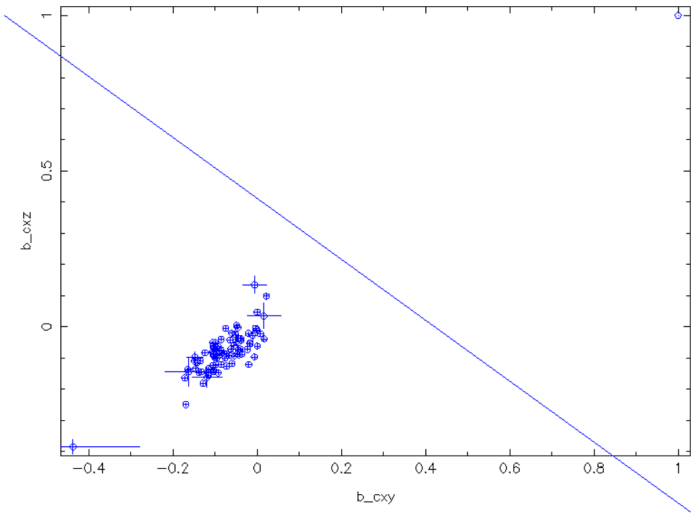
b_cyz.err = 2/(b_ycon.rate+b_zcon.rate)^2*(b_zcon.rate^2*b_ycon.err^2+b_ycon.rate^2*b_zcon.err^2)^1/2;

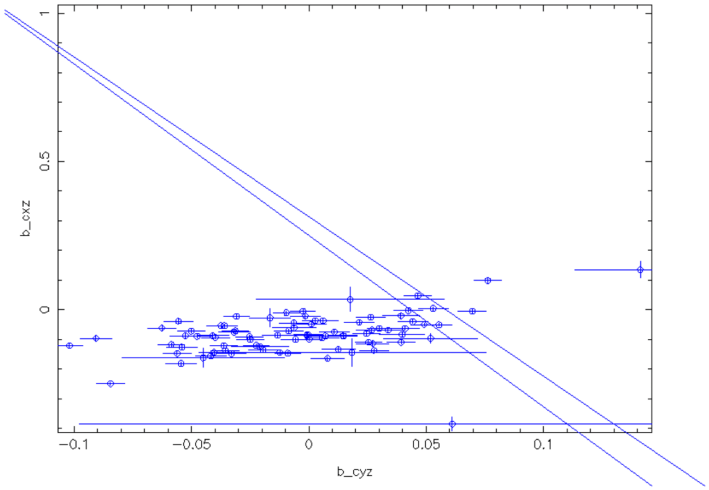
b_czy = struct{cons,err};
b_czy.cons = (b_zcon.rate-b_ycon.rate)/(b_zcon.rate+b_ycon.rate);
b_czy.err = 2/(b_zcon.rate+b_ycon.rate)^2*(b_ycon.rate^2*b_zcon.err^2+b_zcon.rate^2*b_ycon.err^2)^1/2;

b_cxz = struct{cons,err};
b_cxz.cons = (b_xcon.rate-b_zcon.rate)/(b_xcon.rate+b_zcon.rate);
b_cxz.err = 2/(b_xcon.rate+b_zcon.rate)^2*(b_zcon.rate^2*b_xcon.err^2+b_xcon.rate^2*b_zcon.err^2)^1/2;

b_czx = struct{cons,err};
b_czx.cons = (b_zcon.rate-b_xcon.rate)/(b_zcon.rate+b_xcon.rate);
b_czx.err = 2/(b_zcon.rate+b_xcon.rate)^2*(b_xcon.rate^2*b_zcon.err^2+b_zcon.rate^2*b_xcon.err^2)^1/2;







As you can see, there are some issues with this observation; the error bars and colors, in some cases, appear to be really large or nonexistent values, so we must remove these. The previous observation was also checked for these issues, but none were found.

gb_cxy_temp = wherenot(isnan(b_cxy.cons) or isinf(b_cxy.cons) or isnan(b_cxy.err) or isinf(b_cxy.err) or isnan(b_cyz.cons) or isinf(b_cyz.cons) or isnan(b_cyz.err) or isinf(b_cyz.err) or isnan(b_cxz.cons) or isinf(b_cxz.cons) or isnan(b_cxz.err) or isinf(b_cxz.err)); % Excludes all bad data from all light curve increments so that this can plot properly with any other color in this observation.

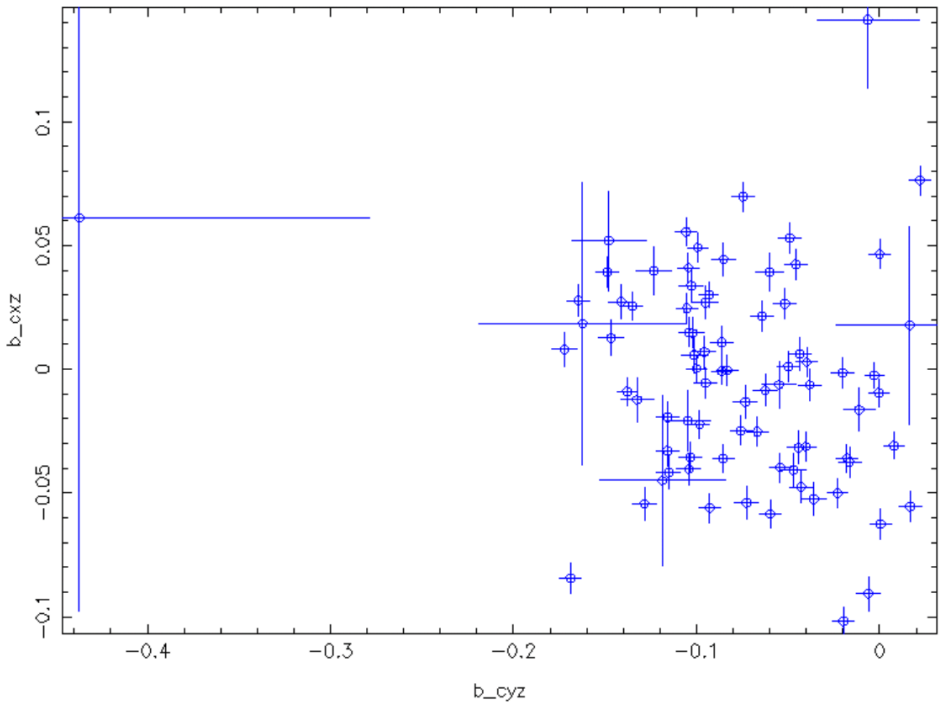
gb_cxy = struct{cons,err};

gb_cxy.cons = b_cxy.cons[gb_cxy_temp];

gb_cxy.err = b_cxy.err[gb_cxy_temp];

gb_cyz_temp = wherenot(isnan(b_cxy.cons) or isinf(b_cxy.cons) or isnan(b_cxy.err) or isinf(b_cxy.err) or isnan(b_cyz.cons) or isinf(b_cyz.cons) or isnan(b_cyz.err) or isinf(b_cyz.err) or isnan(b_cxz.cons) or isinf(b_cxz.cons) or isnan(b_cxz.err) or isinf(b_cxz.err));
gb_cyz = struct{cons,err};
gb_cyz.cons = b_cyz.cons[gb_cyz_temp];
gb_cyz.err = b_cyz.err[gb_cyz_temp];

gb_cxz_temp = wherenot(isnan(b_cxy.cons) or isinf(b_cxy.cons) or isnan(b_cxy.err) or isinf(b_cxy.err) or isnan(b_cyz.cons) or isinf(b_cyz.cons) or isnan(b_cyz.err) or isinf(b_cyz.err) or isnan(b_cxz.cons) or isinf(b_cxz.cons) or isnan(b_cxz.err) or isinf(b_cxz.err));
gb_cxz = struct{cons,err};
gb_cxz.cons = b_cxz.cons[gb_cxz_temp];
gb_cxz.err = b_cxz.err[gb_cxz_temp];

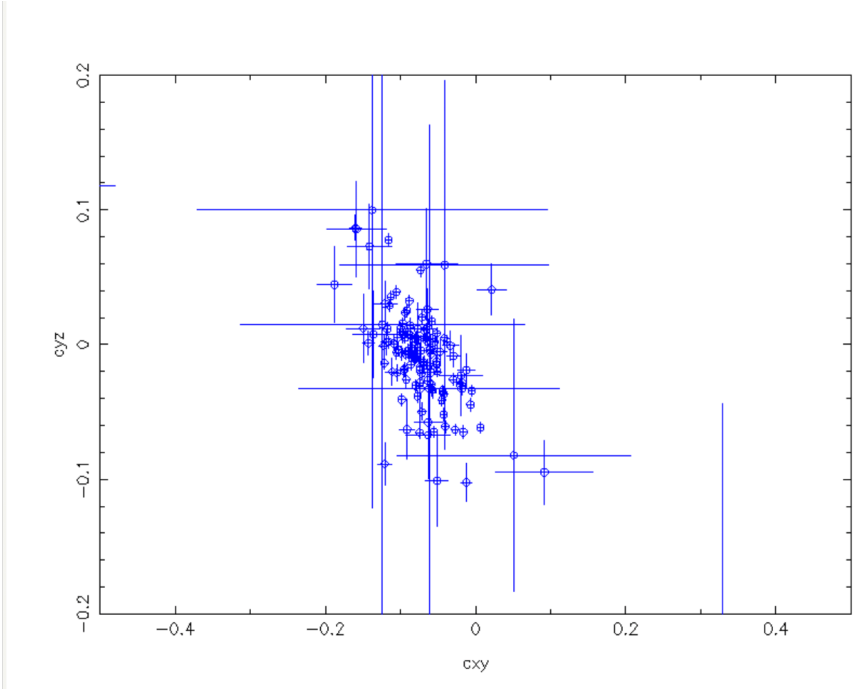


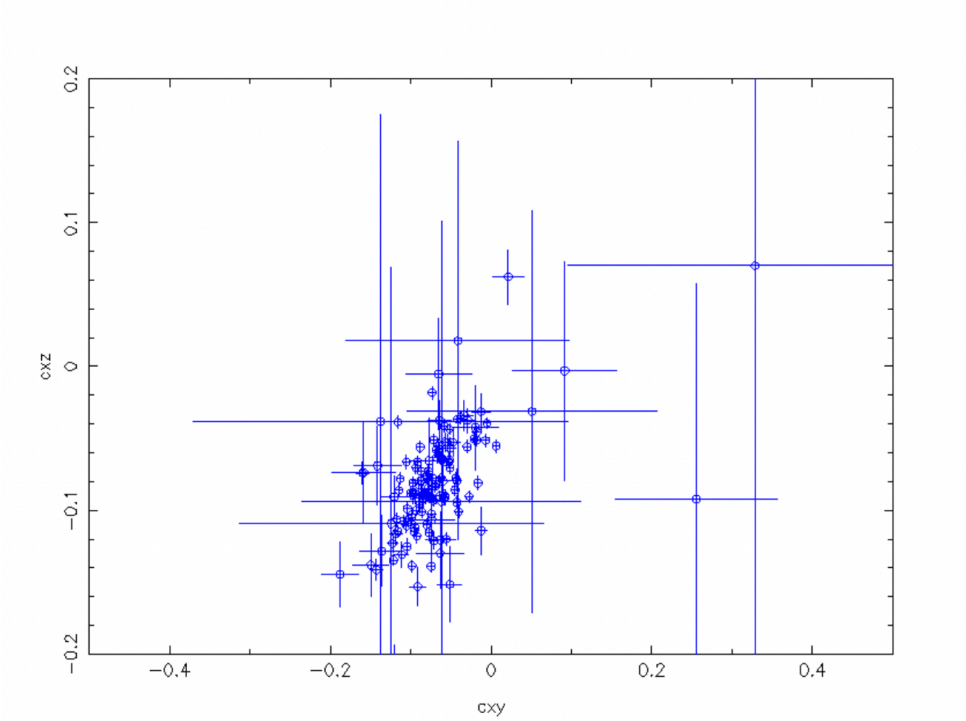
Fixes our problem, but now we will scale all the colors of both observations to the same parameters for a more digestible analysis:

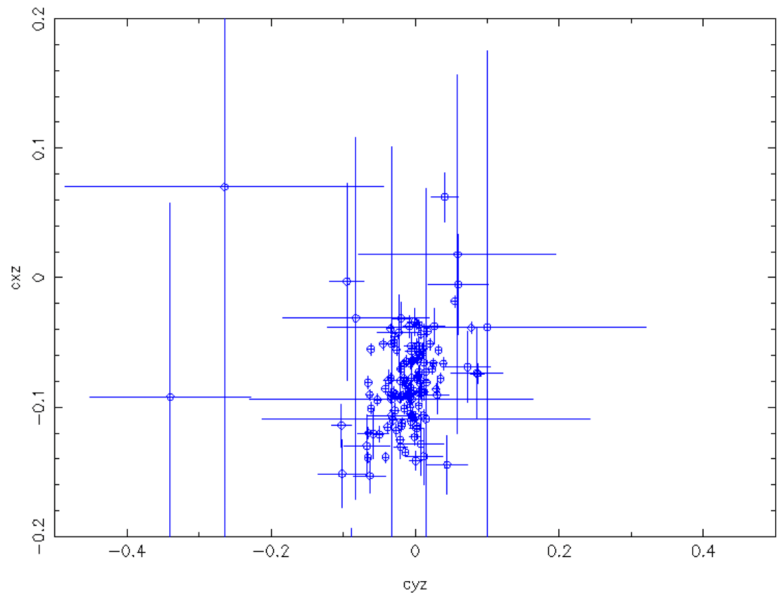
plotxy(...; xrange={-0.5,0.5}, yrange={-0.2,0.2}); % General form

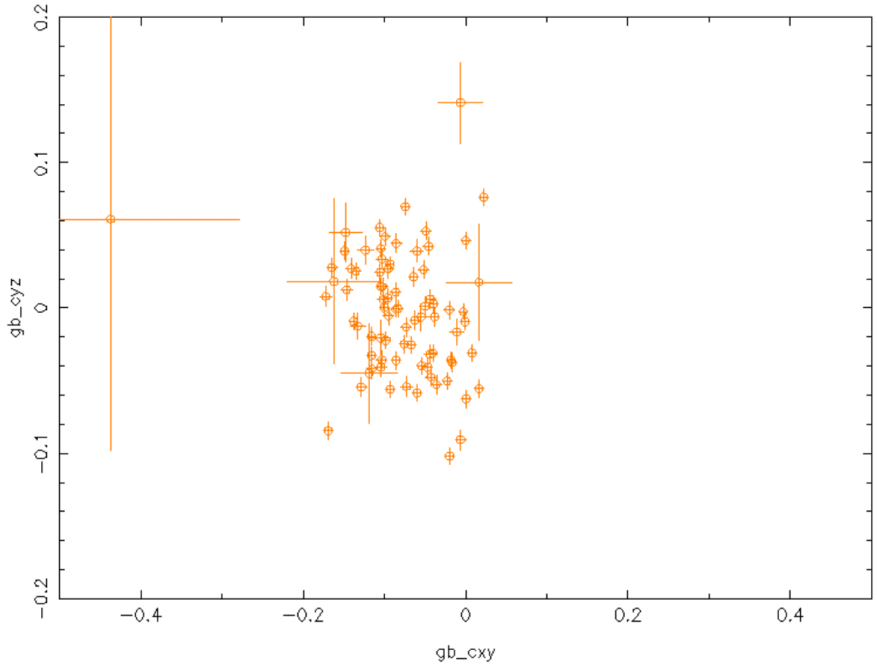
plotxy(cxy.cons,cxy.err,cxy.err,cyz.cons,cyz.err,cyz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2});
plotxy(cxy.cons,cxy.err,cxy.err,cxz.cons,cxz.err,cxz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2});
plotxy(cyz.cons,cyz.err,cyz.err,cxz.cons,cxz.err,cxz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2});

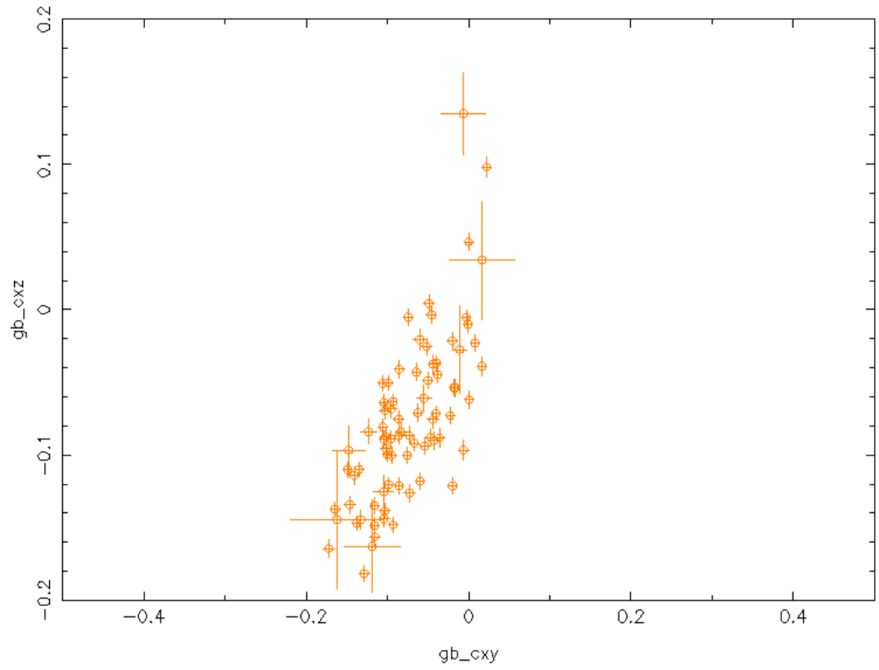
plotxy(gb_cxy.cons,gb_cxy.err,gb_cxy.err,gb_cyz.cons,gb_cyz.err,gb_cyz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2}, dcol=8); % Remember, b_ structures are for the bad data with infinite or undefined error bars (no bueno).
plotxy(gb_cxy.cons,gb_cxy.err,gb_cxy.err,gb_cxz.cons,gb_cxz.err,gb_cxz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2}, dcol=8);
plotxy(gb_cyz.cons,gb_cyz.err,gb_cyz.err,gb_cxz.cons,gb_cxz.err,gb_cxz.err; xrange={-0.5,0.5}, yrange={-0.2,0.2}, dcol=8);

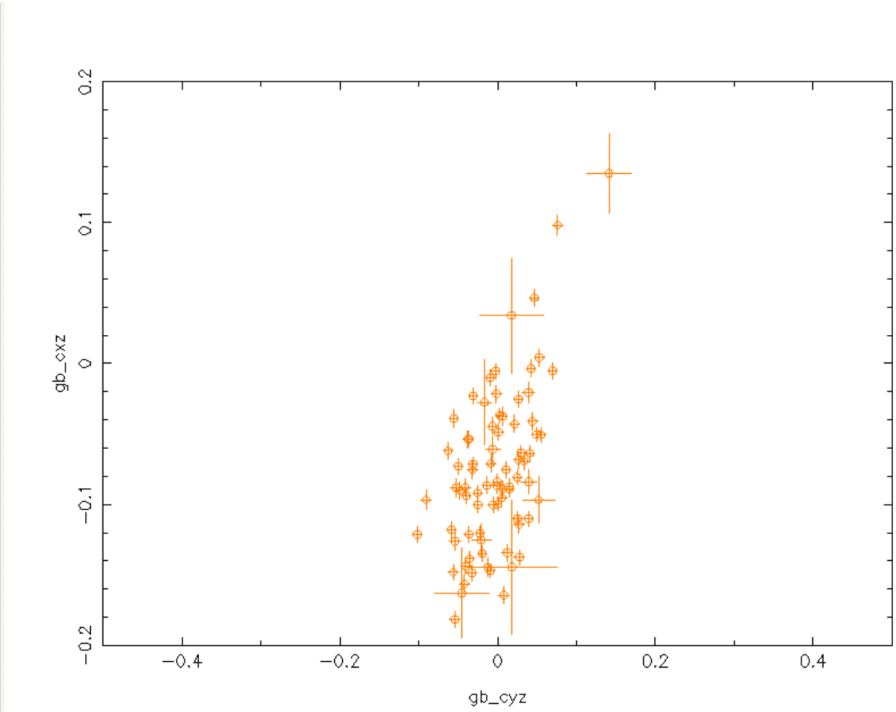



0.2
0.1
0
−0.1
−0.2
cxz
−0.4
−0.2
0
0.2
0.4
cxy

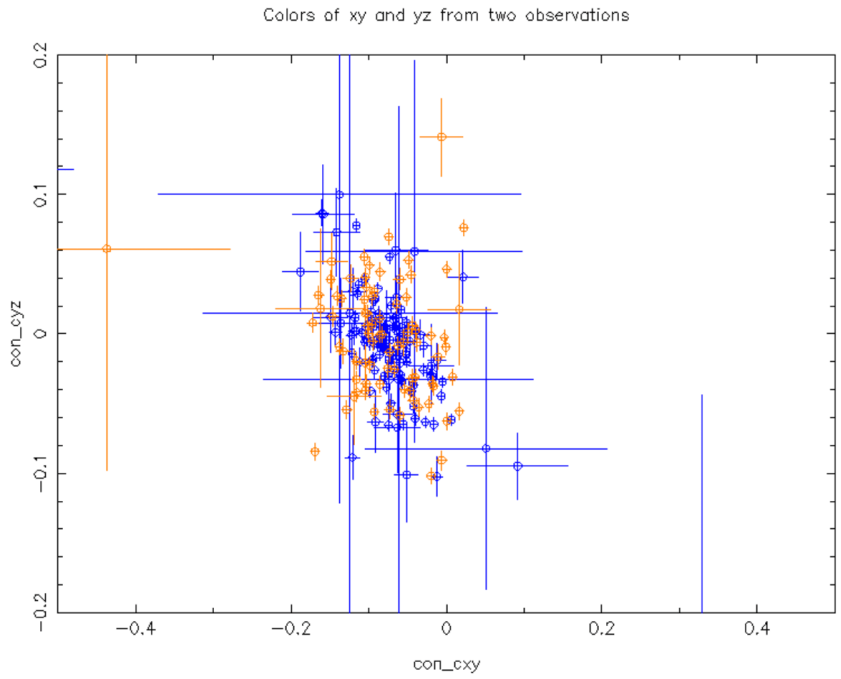

0.2
0.1
0
−0.1
−0.2
cxz
−0.4
−0.2
0
0.2
0.4
cyz

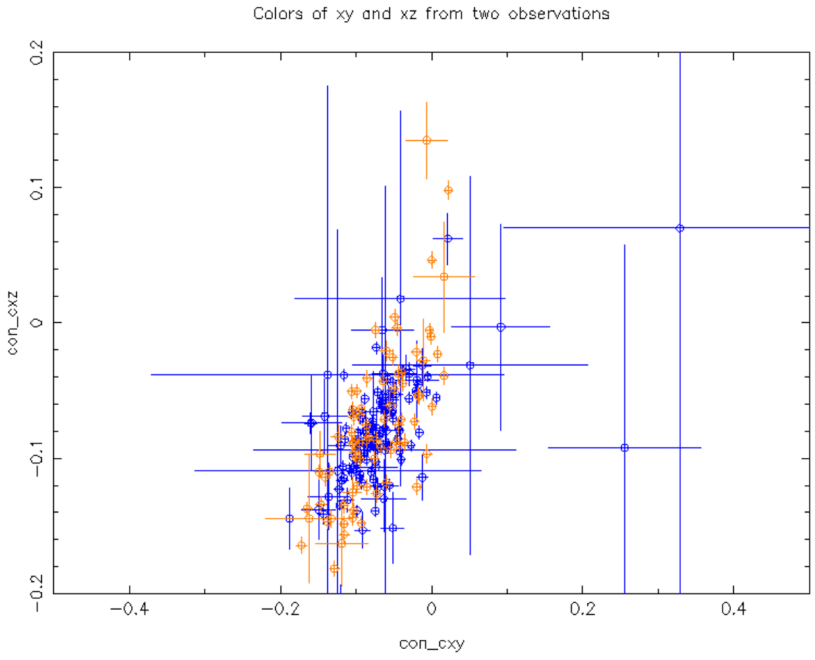

gb_cyz
gb_cxy
0.2
0.1
0
−0.1
−0.2
−0.4
−0.2
0
0.2
0.4


gb_cxz
gb_cxy
0.2
0.1
0
−0.1
−0.2
−0.4
−0.2
0
0.2
0.4



Let's combine the colors of both observations (labeled with "con_" on the axis), times, and sum of rates/intensities matching a particular color (so the sums of rates of x and y versus cxy). Blue is data from observation 90501623002 and orange is from 60160371002:


Colors of xy and yz from two observations
con_cyz
con_cxy
0.2
0.1
0
−0.1
−0.2
−0.4
−0.2
0
0.2
0.4


Colors of xy and xz from two observations
con_cxz
con_cxy
0.2
0.1
0
−0.1
−0.2
−0.4
−0.2
0
0.2
0.4