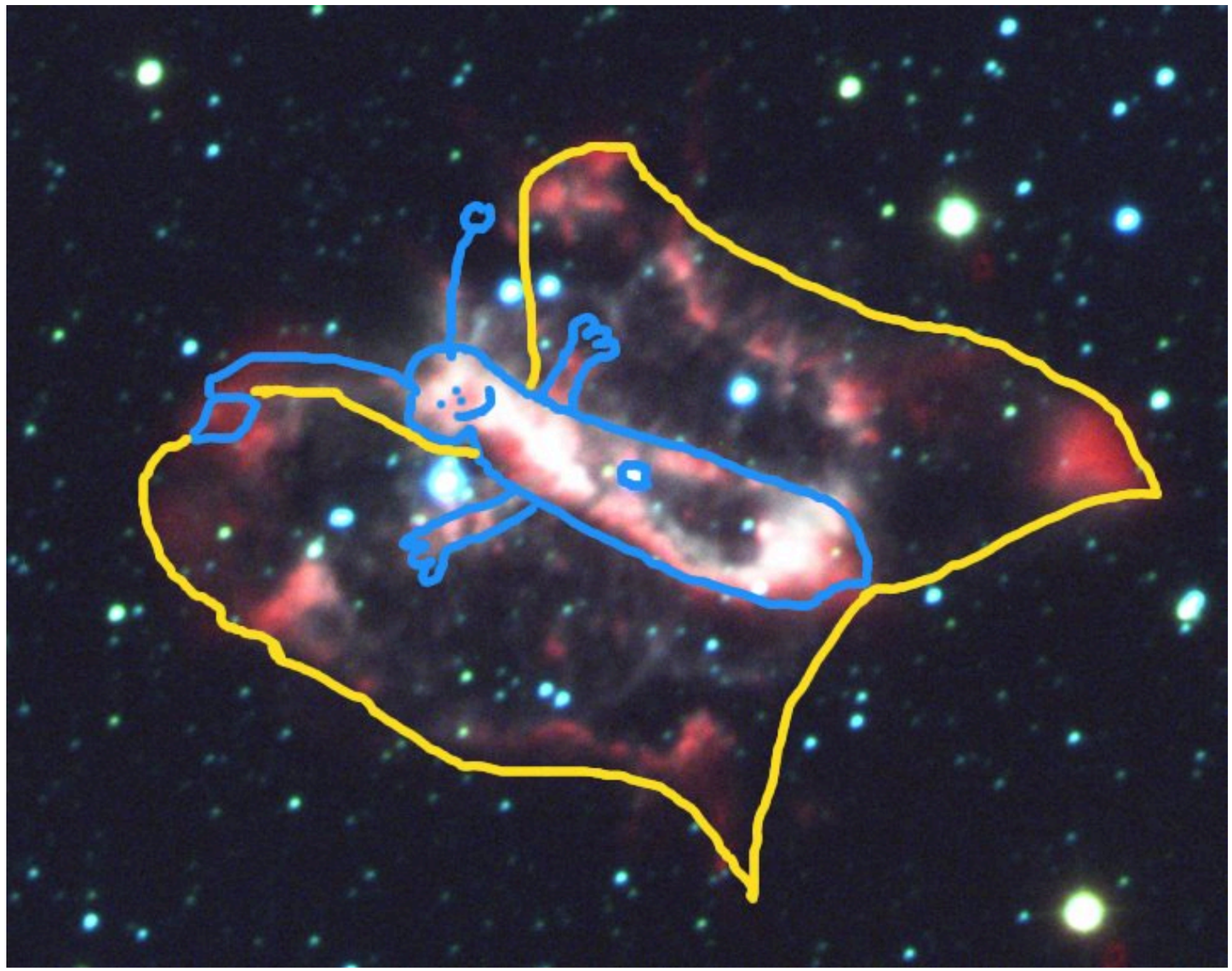
Reducing large amounts of RSS longslit data

Brent Miszalski
SALT Astronomer
brent@saao.ac.za
http://miszalski.saao.ac.za/



Background

- It is possible to reduce hundreds of SALT spectra in a semi-automatic way
- Some may prefer fully automatic, but semi-automatic allows for careful oversight of data quality => Good scientific practice!
- Uses IRAF and scripting to streamline process, using a crib sheet to keep track of steps to apply to data
- Follows Alexei Kniazev's SALT doc which outlines steps of IRAF reduction
  - http://saao.ac.za/~brent/Kniazev_longslit.pdf

Aim of this talk

- To give you an overview of my own approach to reducing RSS data (see also Alexei's talk)
- It is not in any way meant to be a pipeline that you can run in this workshop!
- Instead, to challenge you to think about how to streamline aspects of your approach
- Current code a mix of IRAF and perl, python, pyraf and shell scripts/commands put together of years
- You can probably do it better and faster in python!

Outline of steps

- Pre-processing [organising data]
- Cosmic ray cleaning [always before flat fielding!]
- (Flat fielding) [internal quartz lamp flats]
- Arc identification on 2D frame
- Transforming science data into rectified wavelength calibrated 2D frames
- Copying missing header info to rectified frames
- Extraction of 1D spectra [using apall]
- Spectrophotometric calibration

Pre-processing

- Take product data from SALT daily pipeline (mbxgpP*.fits)
- In top level directory, arrange data into folders named after each night data taken in (e.g. 0801 is 1st Aug)
- Generate summary of data from night using dfits and fitsort (C programs available from ESO)
- dfits m*P*.fits | fitsort object exptime camang grating lampid filter ccdsum grtilt > summary.txt
- Top level directory contains scripts for handling data in subdirs representing each night for easy access using '../' path

Pre-processing

FILE	OBJECT	EXPTIME	CAMANG	GRATING	LAMPID	FILTER	CCDSUM	GRTILT
mbxgpP201508010013.fits	2MASS J17072529-0138093	600.204	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010014.fits	2MASS J17072529-0138093	600.206	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010015.fits	ARC	2.195	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010016.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010017.fits	FLAT	1.194	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010018.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010019.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010020.fits	FLAT	1.194	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010021.fits	2MASS J15485834-1636018	600.206	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010022.fits	2MASS J15485834-1636018	600.208	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010023.fits	ARC	2.198	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010024.fits	FLAT	1.196	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010025.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010026.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010027.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010028.fits	FLAT	1.196	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010035.fits	2MASS J01165283-6455570	600.21	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010036.fits	2MASS J01165283-6455570	600.211	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010037.fits	ARC	2.198	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010038.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010039.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010040.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010041.fits	FLAT	1.199	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010042.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000

Tip: Check arcs and flats taken with same settings as your science data!
Especially check CAMANG and FILTER. Good idea to view all data in ds9 too.
(pipeline may include similar data not associated with your program)

Pre-processing

- Product data have multiple extensions. To simplify things and get started, I copy the science extensions to new images with simplified naming convention using a perl script to export an IRAF script

```
#!/usr/bin/perl

open FILE,"<","summary.txt";
while(<FILE>){
   if($_ =~ /fits/){
      @a = split;
      $fname = $a[0];
      $type= $a[1];
      $cam= $a[3];
      $fname =~ /.*([0-9]{3}).fits/;
      $num = $1;
      $num =~ s/^0+//g;

      if($type =~ /ARC/){
         print "imcopy $fname\[SCI\]\[*,300:700\] arc$num.fits\n";
      } elsif($type =~ /FLAT/){
         print "imcopy $fname\[SCI\]\[*,300:700\] f$num.fits\n";
      } else {
         print "imcopy $fname\[SCI\]\[*,300:700\] obj$num.fits\n";
      }
   }
}
close FILE;
```

bash shell> ../parse.pl > parse.cl [in terminal]
ecl> cl < parse.cl [in IRAF xgterm window]

```
etoile:0801 brent$ ls arc*[0-9].fits obj*[0-9].fits flat*[0-9].fits
arc15.fits      arc42.fits      flat38.fits     obj21.fits      obj36.fits
arc23.fits      flat16.fits     obj13.fits      obj22.fits
arc37.fits      flat24.fits     obj14.fits      obj35.fits
```

data trimmed in y axis (spatial axis) to central portion

↓

Warning! This doesn't copy over the header from the science frame. I deal with that later...

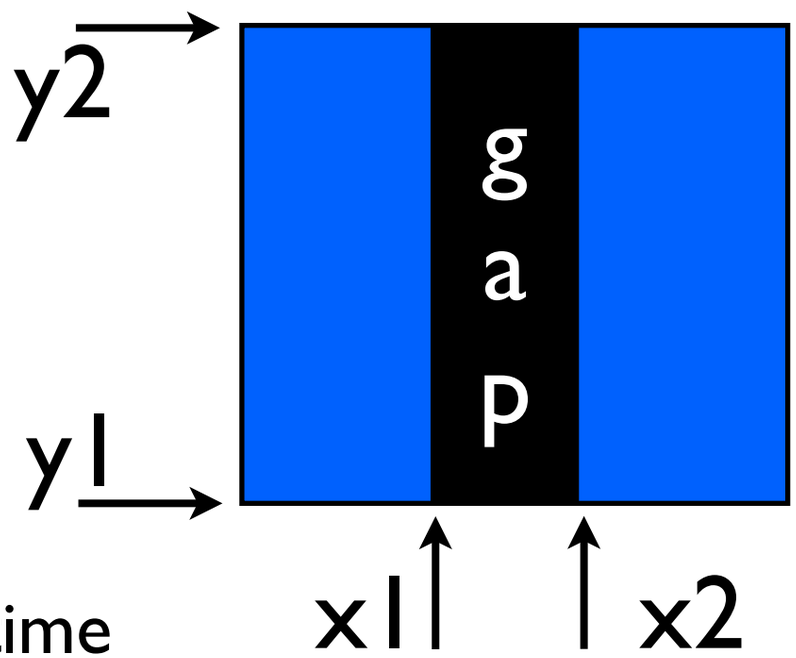
Pre-processing

- I then run an IRAF script trim.cl to remove junk header keywords from arc*fits obj*fits f*fits (mostly WCS related: CRPIX1, CRVAL1, etc.)
- For science and flat field frames, I use ofixpix in the obsolete IRAF package to interpolate over chip gaps. This is done at the end of the trim.cl script

```
obsolete
ofixpix obj*fits badpixel=../box.txt
ofixpix f*fits badpixel=../box.txt
bye
```

box.txt:

```
1020 1073 1 401
2096 2145 1 401
```



Warning! ofixpix overwrites your science data
Make a backup. Chip gaps also move from time to time

Cosmic ray cleaning

- Laplacian Cosmic Ray Identification and removal (van Dokkum 2001, PASP, 113,1420)
- http://www.astro.yale.edu/dokkum/lacosmic/
- I use the IRAF imaging version (install as package, uses STSDAS), but there's also a python version

- `ls obj*.fits | awk '{gsub(/.fits/,"", $1);gsub(/obj/,"",$1);printf("lacos input=obj%s.fits output=obj%scc.fits outmask=mask%s.fits\n",$1,$1,$1)}' > clean.cl` [in terminal]
- `stsdas` [load package in IRAF]
- `cl < clean.cl` [in IRAF]

Cosmic ray cleaning

```
PACKAGE = clpackage
   TASK = lacos_im

input   = ▉          obj5.fits  input image
output  =         obj5cc.fits  cosmic ray cleaned output image
outmask =          mask5.fits  output bad pixel map (.pl)
(gain   =                 1.5) gain (electrons/ADU) (0=unknown)
(readn  =                  7.) read noise (electrons) (0=unknown)
(statsec=                 *,*) section to use for automatic computation of gain
(skyval =                  0.) sky level that has been subtracted (ADU)
(sigclip=                 15.) detection limit for cosmic rays (sigma)
(sigfrac=                0.15) fractional detection limit for neighbouring pixe
(objlim =                  2.) contrast limit between CR and underlying object
(niter  =                   3) maximum number of iterations
(verbose=                 yes)
(mode   =                  ql)
```

Tips

sigclip - higher than ~10 for bright moon data (with high background), less for dark/grey moon data

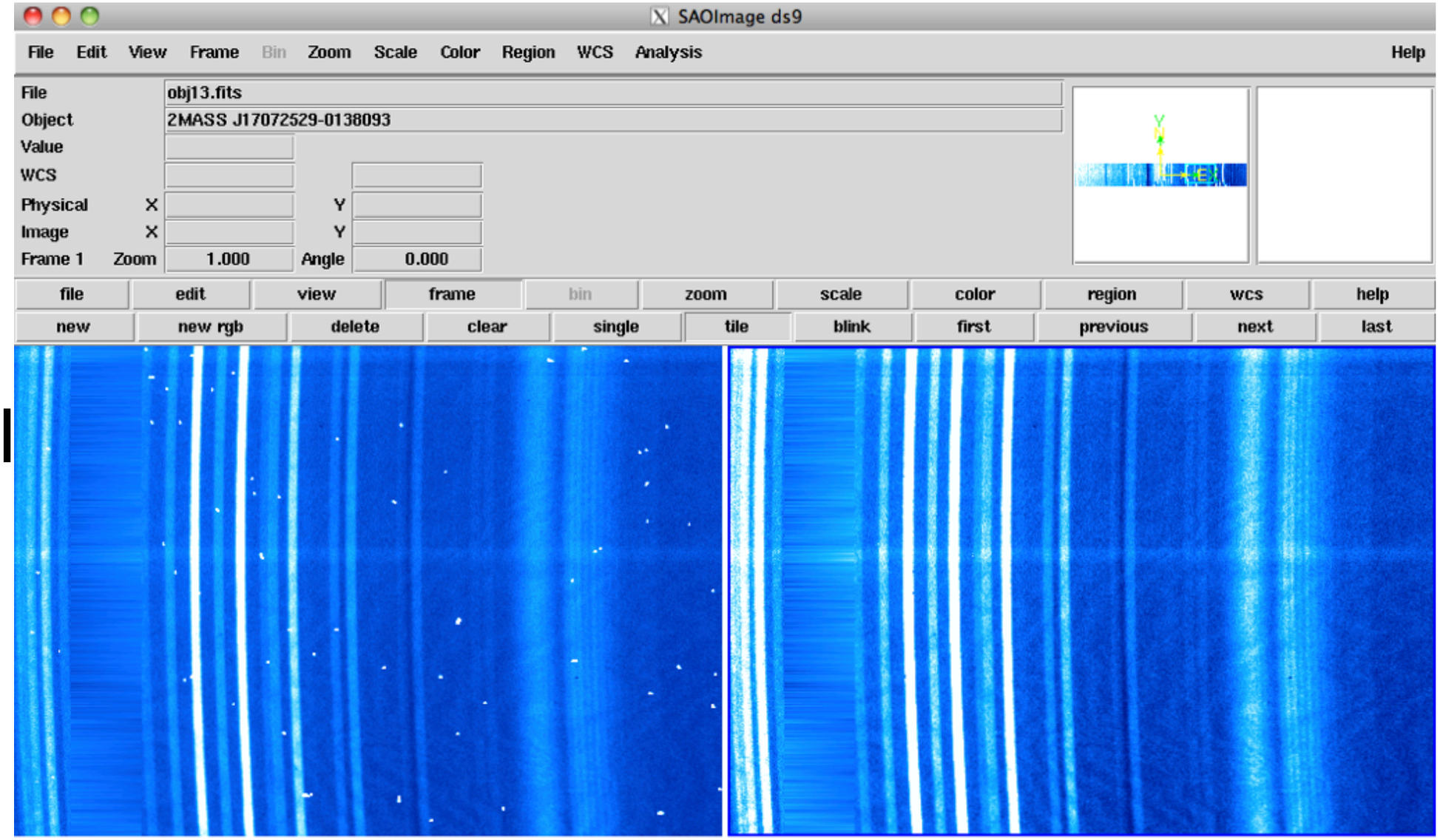
Reduce sigfrac and objlim to get more thorough clean.

Decreasing sigclip in particular can have adverse effect of eating into bright continuum or emission lines in some cases.

Cosmic ray cleaning

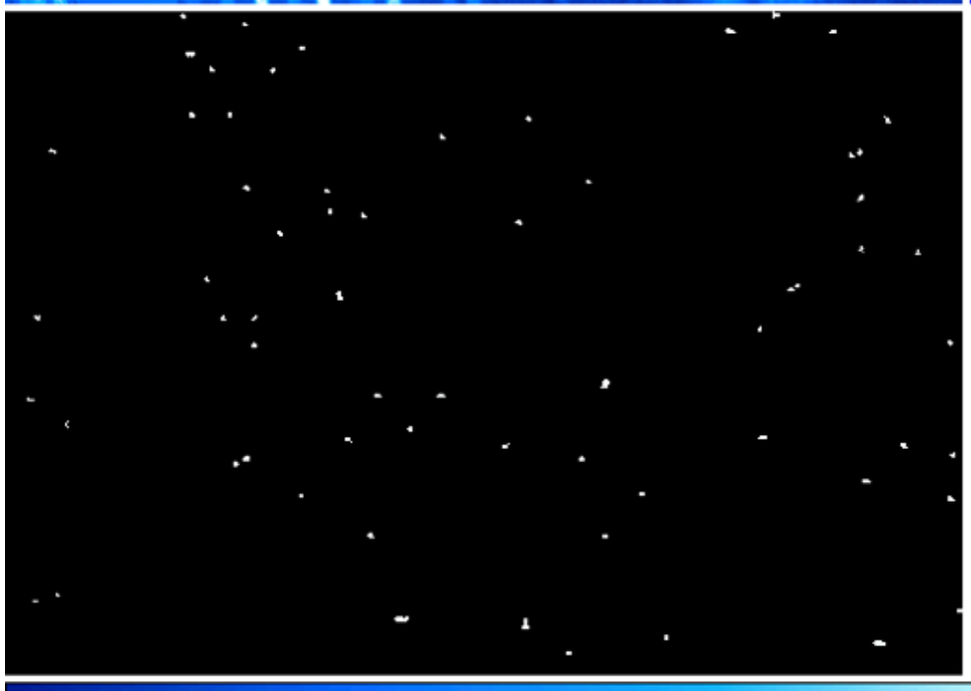
- View your clean quality in ds9
- `for x in `ls obj*[0-9].fits | sed 's/obj//g' | sed 's/.fits//g'`; do echo $x; echo "ds9 -geometry **2500x1300** obj_.fits obj_cc.fits mask_.fits -single -frame 1" | sed "s/_/$x/g" | sh;done`
- This will load ds9 with three frames containing
  - original image
  - mask image (bitmask showing cleaned pixels)
  - cleaned image
- Note: 2500x1300 is window size; reduce if needed

original



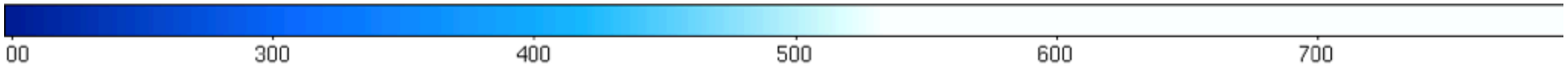
cleaned

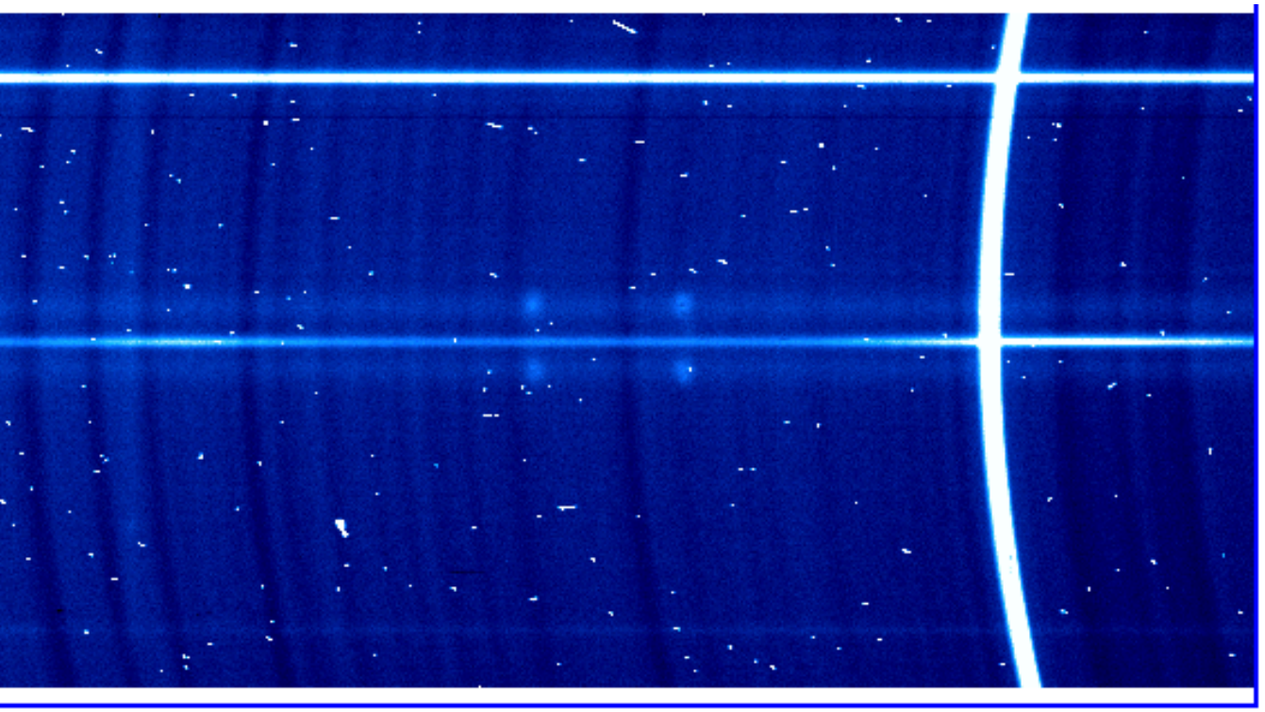
mask

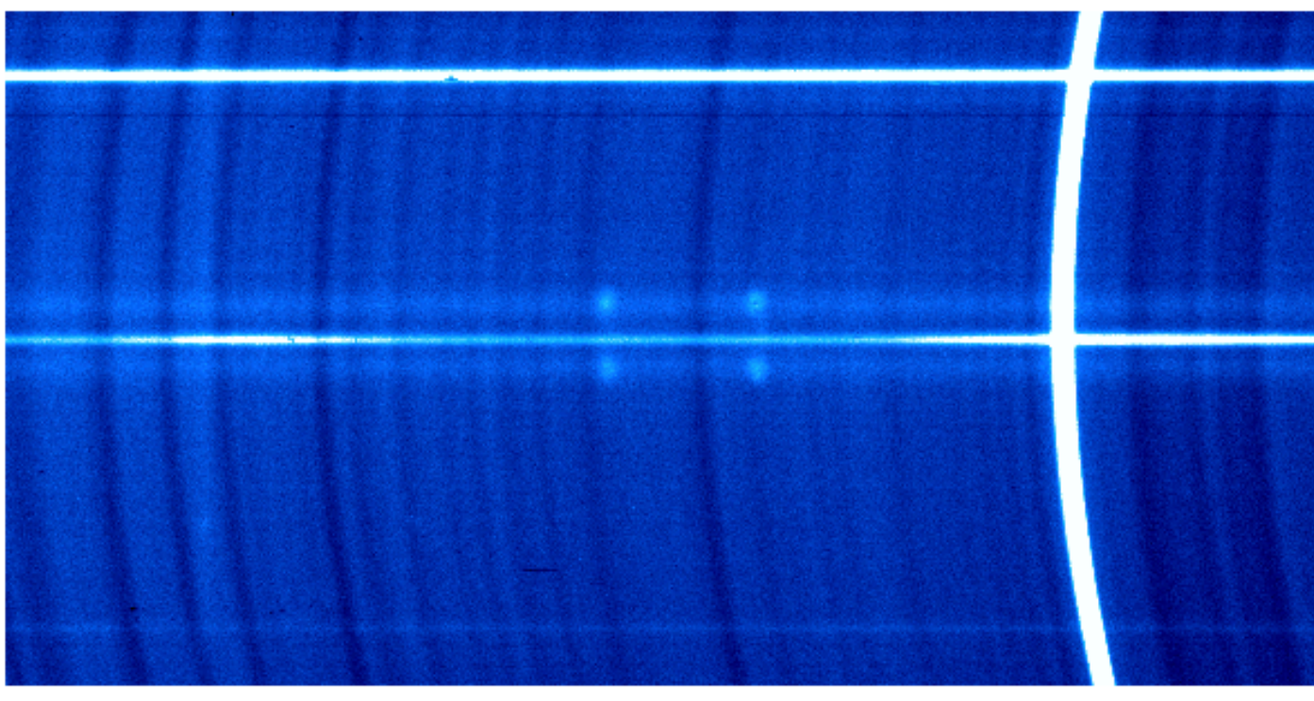


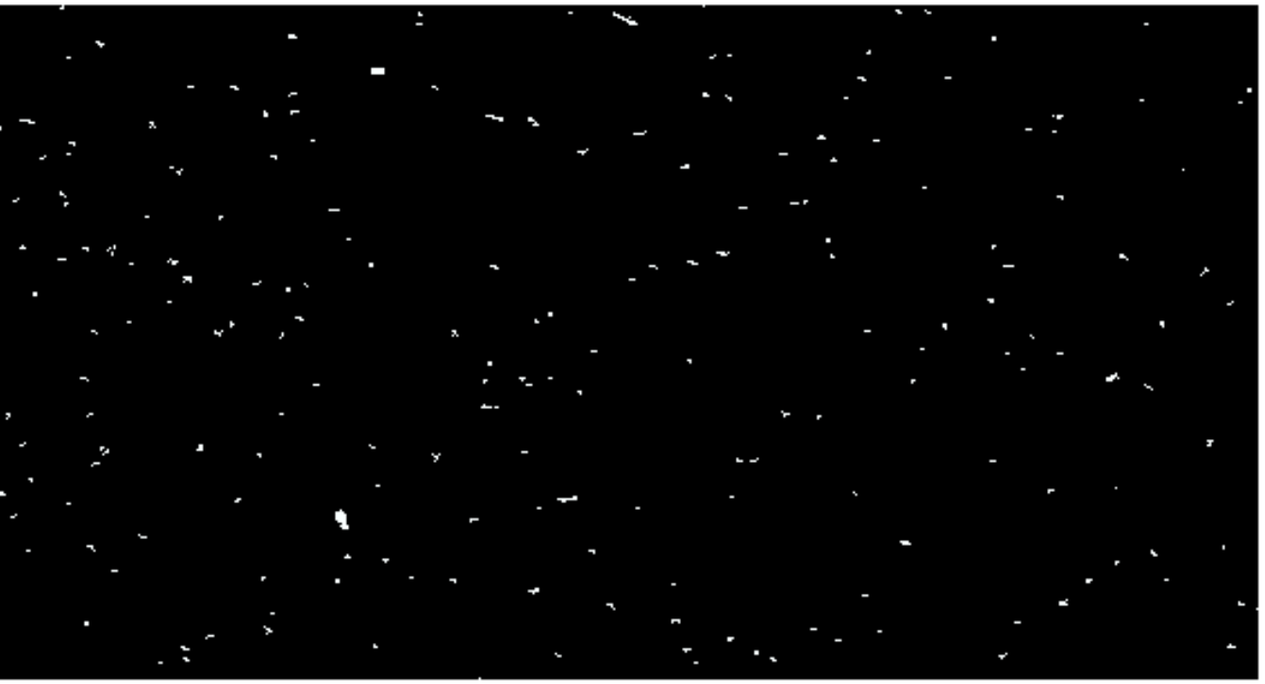
Tip: Use TAB to blink all three together

Pan using middle button and then menu Frame->match frames->Image









NGC6369
1800s
PG2300
4x2 binning

Tip: Adjust scaling in ds9 to see how continuum was handled

Flat-fielding

A somewhat complex example.
Can script the creation and application of flats...

FILE	OBJECT	EXPTIME	CAMANG	GRATING	LAMPID	FILTER	CCDSUM	GRTILT
mbxgpP201508010013.fits	2MASS J17072529-0138093	600.204	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010014.fits	2MASS J17072529-0138093	600.206	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010015.fits	ARC	2.195	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010016.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010017.fits	FLAT	1.194	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010018.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010019.fits	FLAT	1.195	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010020.fits	FLAT	1.194	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010021.fits	2MASS J15485834-1636018	600.206	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010022.fits	2MASS J15485834-1636018	600.208	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010023.fits	ARC	2.198	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010024.fits	FLAT	1.196	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010025.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010026.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010027.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010028.fits	FLAT	1.196	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010035.fits	2MASS J01165283-6455570	600.21	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010036.fits	2MASS J01165283-6455570	600.211	41.50	PG0900	NONE	PC04600	2 4	20.750000
mbxgpP201508010037.fits	ARC	2.198	41.50	PG0900	Ne	PC04600	2 4	20.750000
mbxgpP201508010038.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010039.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010040.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010041.fits	FLAT	1.199	41.50	PG0900	QTH2	PC04600	2 4	20.750000
mbxgpP201508010042.fits	FLAT	1.198	41.50	PG0900	QTH2	PC04600	2 4	20.750000

flatme.pl

```
#!/usr/bin/perl

#Parse the summary file
open FILE,"<","summary.txt";
while(<FILE>){
    if($_ =~ /FLAT/){
        @a = split / /;
        #Get the image number expected of each flat using regex
        $a[0] =~ /.*${dir}0+([0-9]+).fits/;
        #we now have the flat filename of our trimmed flat
        $fname = "f$1.fits";
        #an array storing the flat names
        push(@flats,$fname);
    }
}
close FILE;
```

flatme.pl

```
#for each flat
$nflats =scalar @flats;
for($i=0;$i<$nflats;$i++){
   $flats[$i] =~ /([0-9]+)/;
   $fnum = $1;
   #take the first flat and the following 4 flats to create a combined flat
   print "imcombine input=\"";
   push(@newflats,$fnum);
   for($j=$fnum;$j<$fnum+5;$j++){
      if($j != $fnum+4){
         print "f$j.fits,";
      } else {
         print "f$j.fits\" output=\"flat$fnum.fits\" combine=average reject=minmax\n";
         #make a smoothed median of the flat
         print "median flat$fnum.fits flat${fnum}med.fits xwindow=100 ywindow=100 boundary=nearest constant=0\n";
         #divide it out to create our flat ready for dividing out of science data
         print "imarith flat$fnum.fits / flat${fnum}med.fits flat${fnum}done.fits\n";
         print "\n";
      }
   }
   #skip those flats we've already considered in j for loop
   $i += 4;
}
```

flatme.pl

```
#find the cleaned object frames and make commands to apply the flats
@objs = glob "obj*cc.fits";
 foreach $o (@objs){
       $o =~ /([0-9]+)/;
       $on = $1;
   #only get those with image numbers close to our object frame
   #note: good to check output in case numbers from telescope are not all close together
   #and manually edit produced script if needbe.
   foreach $n (@newflats){
      if($n-$on > 0 and $n-$on  <= 4){
              print "imarith obj${on}cc.fits / flat${n}done.fits obj${on}ccf.fits\n";
         }
      }
}
```

../flatme.pl > flat.cl [in terminal]

cl < flat.cl [in IRAF]

```
imcombine input="f16.fits,f17.fits,f18.fits,f19.fits,f20.fits" output="flat16.fits" combine=average reject=minmax
median flat16.fits flat16med.fits xwindow=100 ywindow=100 boundary=nearest constant=0
imarith flat16.fits / flat16med.fits flat16done.fits

imcombine input="f24.fits,f25.fits,f26.fits,f27.fits,f28.fits" output="flat24.fits" combine=average reject=minmax
median flat24.fits flat24med.fits xwindow=100 ywindow=100 boundary=nearest constant=0
imarith flat24.fits / flat24med.fits flat24done.fits

imcombine input="f38.fits,f39.fits,f40.fits,f41.fits,f42.fits" output="flat38.fits" combine=average reject=minmax
median flat38.fits flat38med.fits xwindow=100 ywindow=100 boundary=nearest constant=0
imarith flat38.fits / flat38med.fits flat38done.fits
```

```
imarith obj13cc.fits / flat16done.fits obj13ccf.fits
imarith obj14cc.fits / flat16done.fits obj14ccf.fits
imarith obj21cc.fits / flat24done.fits obj21ccf.fits
imarith obj22cc.fits / flat24done.fits obj22ccf.fits
imarith obj35cc.fits / flat38done.fits obj35ccf.fits
imarith obj36cc.fits / flat38done.fits obj36ccf.fits
```

Flat-fielding notes

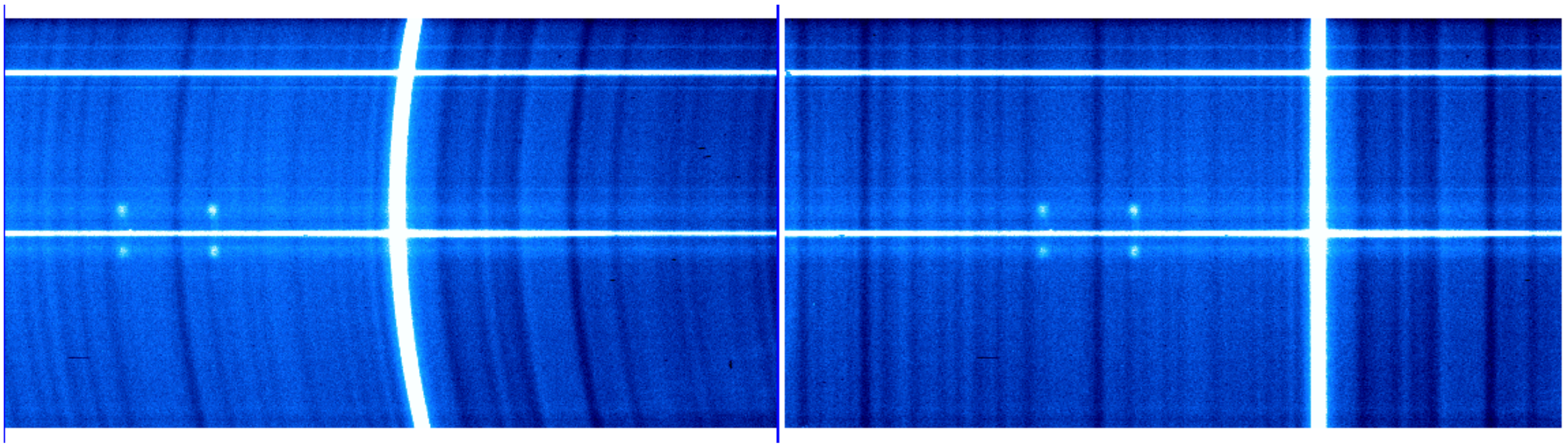
- Internal flats may or may not remove most of your fringes
- Dithering along slit, also, may or may not work
- Using flats does help reduce impact of bad pixels
- Good idea to request them for wavelengths >~7400 A
- Essential to clean cosmics before flat-fielding. Lacosmic won't work properly if flat-fielding data before cleaning.

Arc calibration

- Your science frames are now almost ready
- Time to put a wavelength scale on them and correct for camber (convex shape) in spatial direction

Before

After



Approach

- Usually have multiple arcs per night of same setup. Sometimes mixed lamps/setups, but these can be dealt with separately.
- Identify first arc in sequence using identify task
- Then run reidentify on first arc and all subsequent arcs to get identification of groups of rows (automatic!)
- Works if shifts between arcs are small (usually case)
- May have to repeat identify from scratch if large shift gives you larger than usual RMS in reidentify
- If you get large RMS, always check identify on suspicious arcs individually

Before you start

- Find arc linelist and arc line identification atlas for the lamp used with your data (see LAMPID header)
- I have put together a compilation of both at http://miszalski.saao.ac.za/salt-rss/

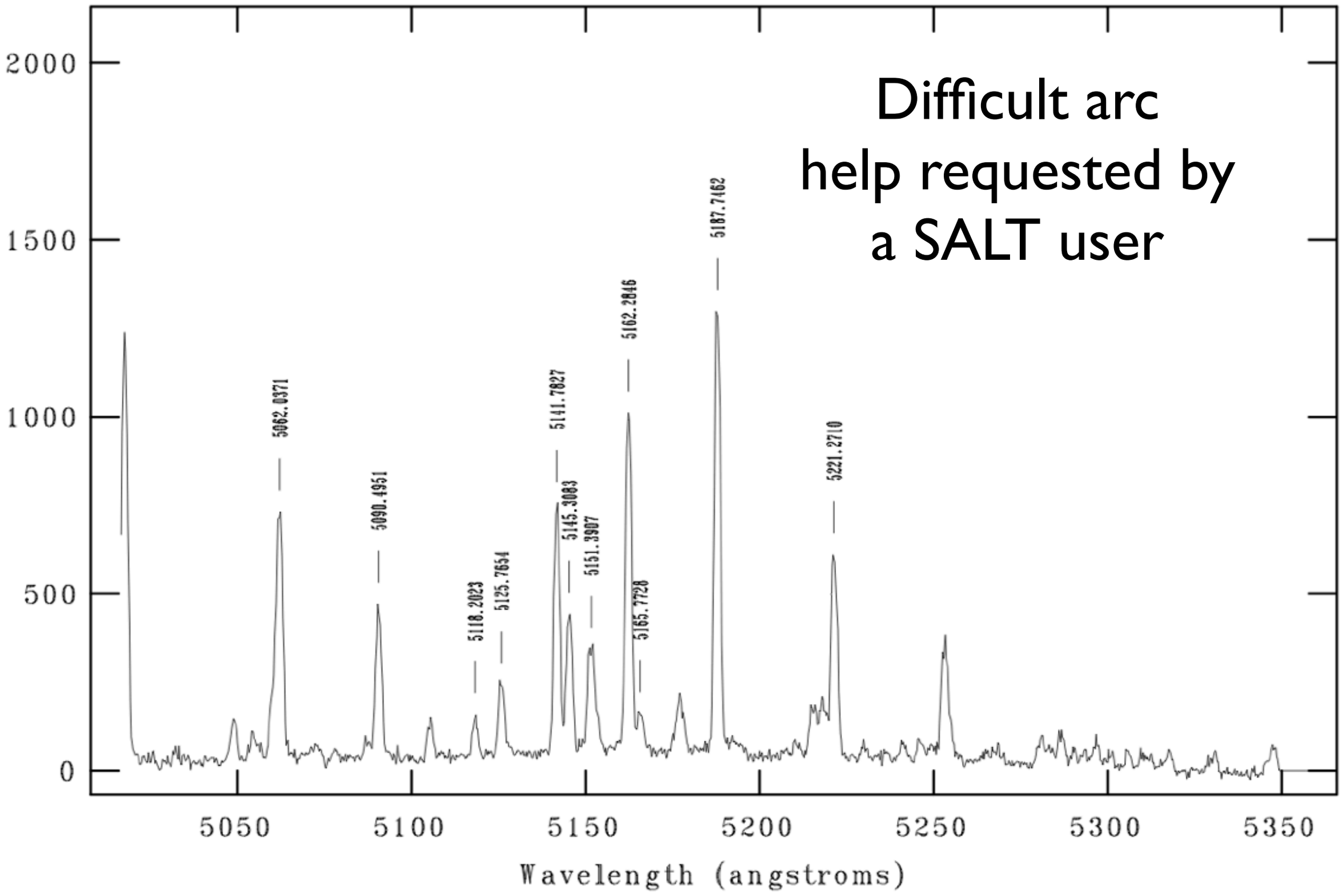
Lamp	Grating	Camang (deg)	Slit (")	Lambda (A)	Files
CuAr	PG2300	61.0	1.5	3836-4923	link
CuAr	PG2300	64.0	0.6	4047-5102	link
CuAr	PG2300	71.5	1.5	4539-5549	link
CuAr	PG2300	79.0	1.5	5012-5972	link
ThAr	PG2300	61.0	1.5	3836-4923	link
ThAr	PG2300	70.0	1.5	4443-5462	link
NeAr	PG900	40.75	1.5	6152-9151	link
Xe	PG900	30.25	1.5	4190-7270	link
Xe	PG900	31.0	1.5	4331-7407	link
Xe	PG900	42.25	1.5	6436-9423	link
Ar	PG900	28.75	1.5	3906-6996	link
Ar	PG2300	79.0	1.5	5007-5979	link
Ar	PG2300	83.5	1.5	5290-6212	link

Tip: Contact me if you need help identifying a difficult arc and I'll do my best to help you out and update the page

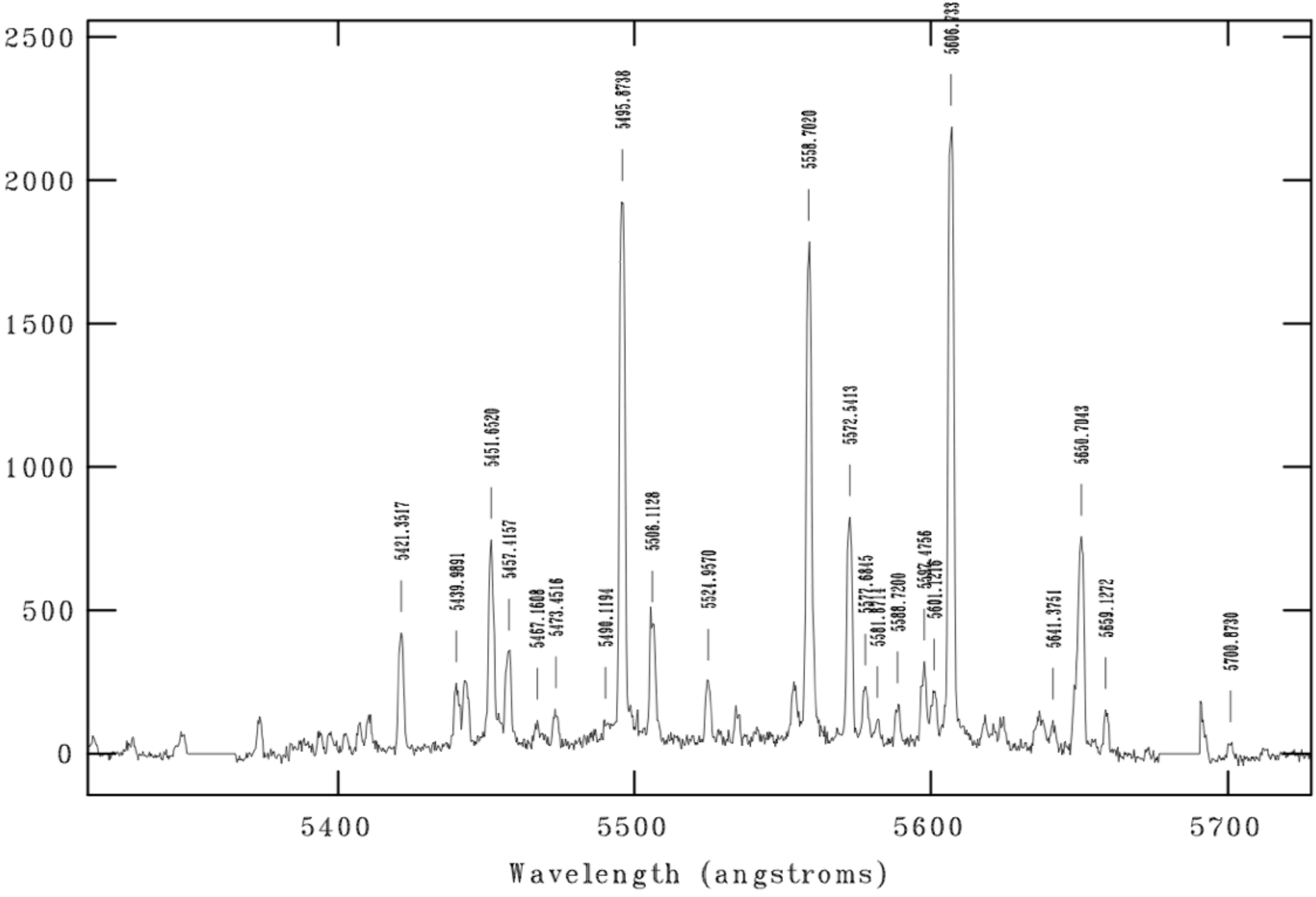
Some also:
http://pysalt.salt.ac.za/lineatlas/lineatlas.html

http://iraf.noao.edu/specatlas/

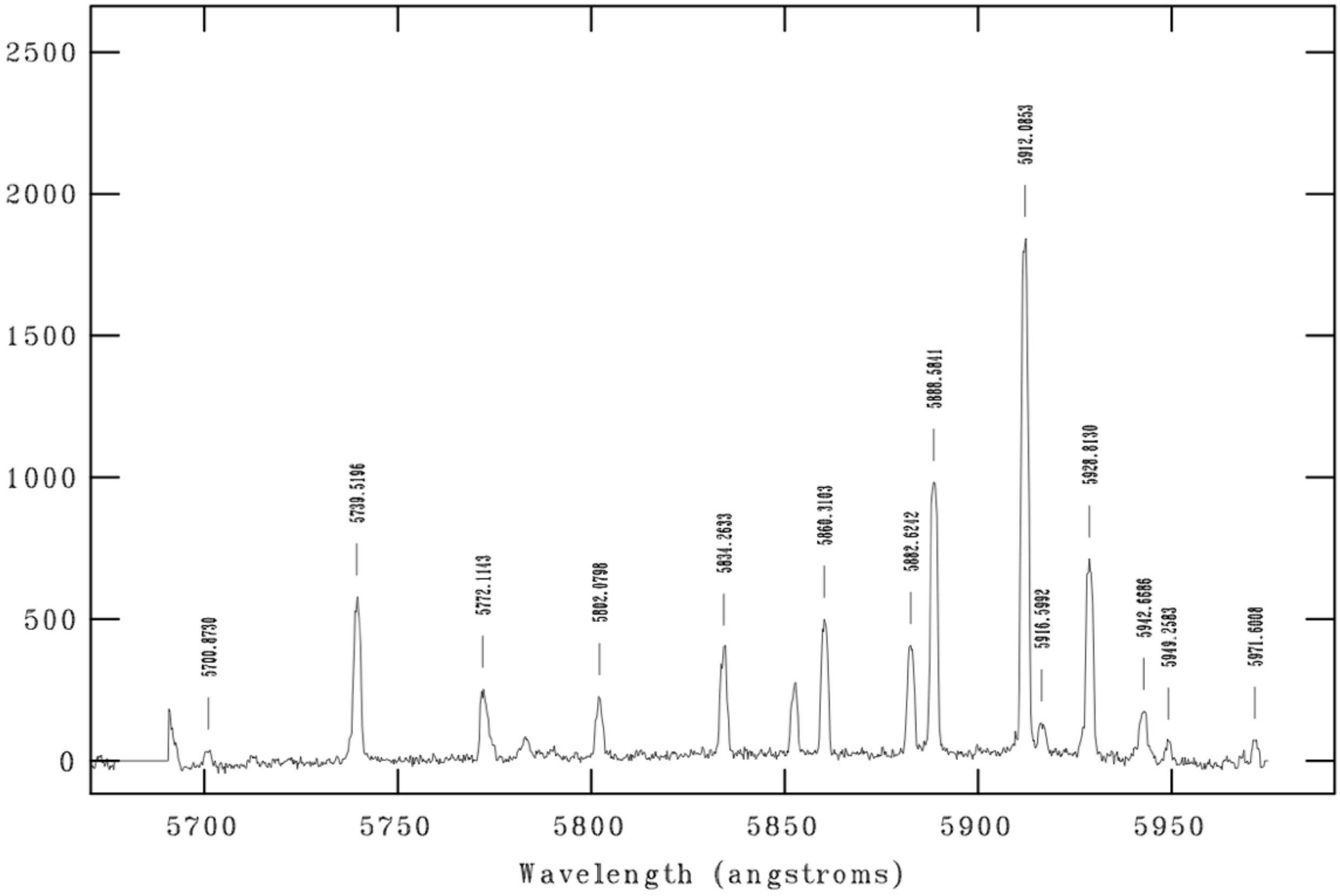
Example: CuAr PG2300; CAMANG 79.0 deg

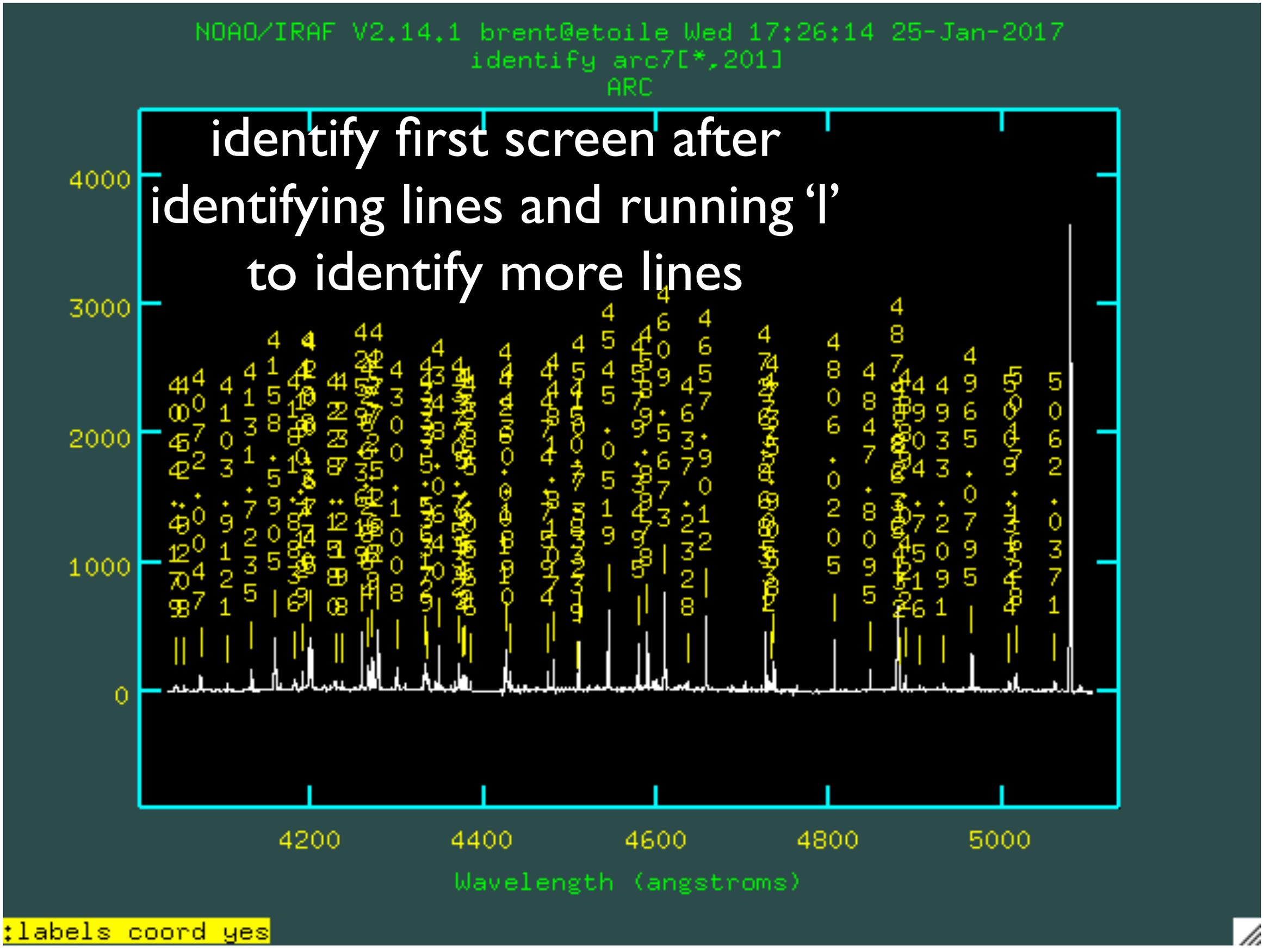


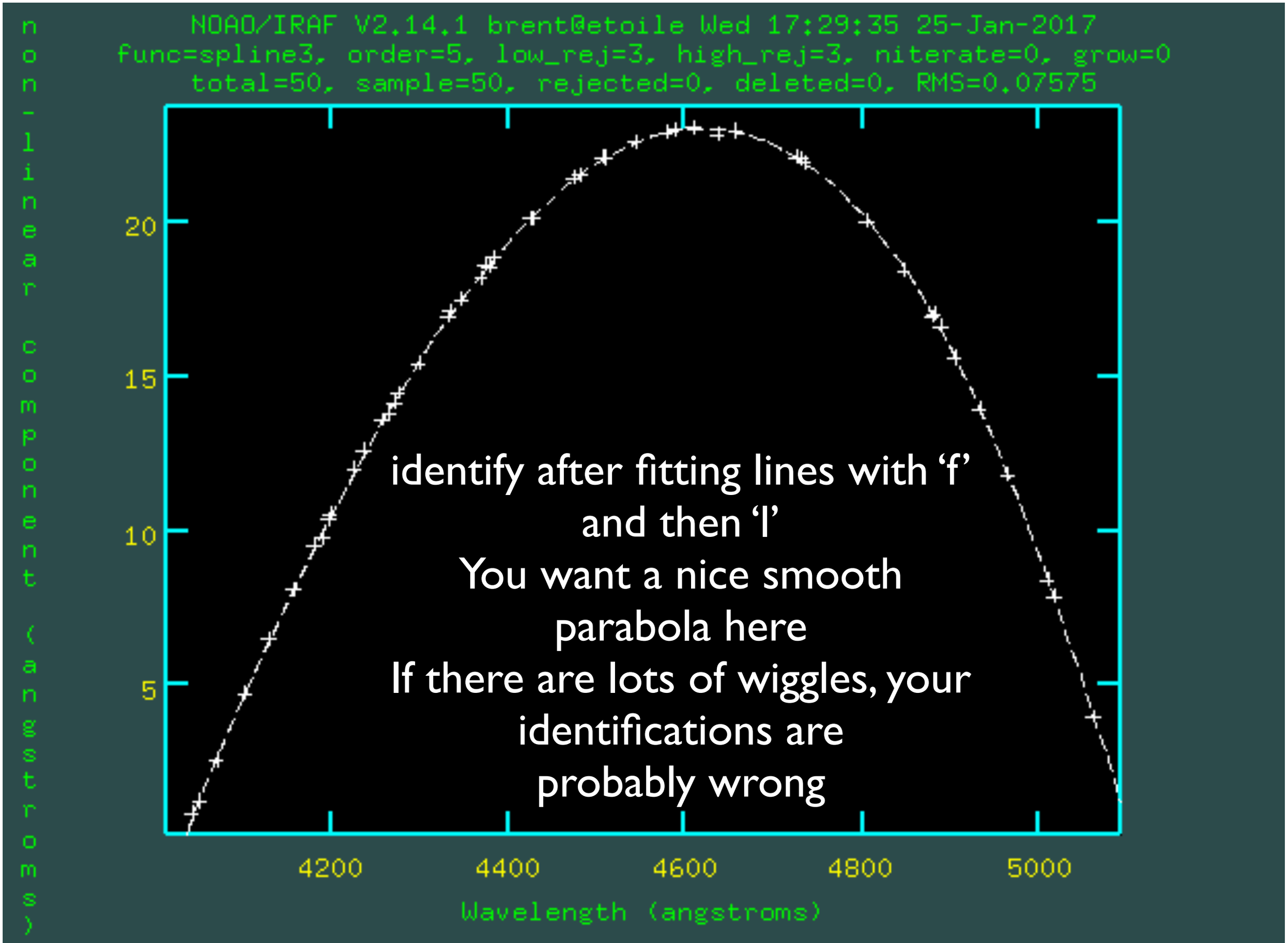
Example: CuAr PG2300; CAMANG 79.0 deg

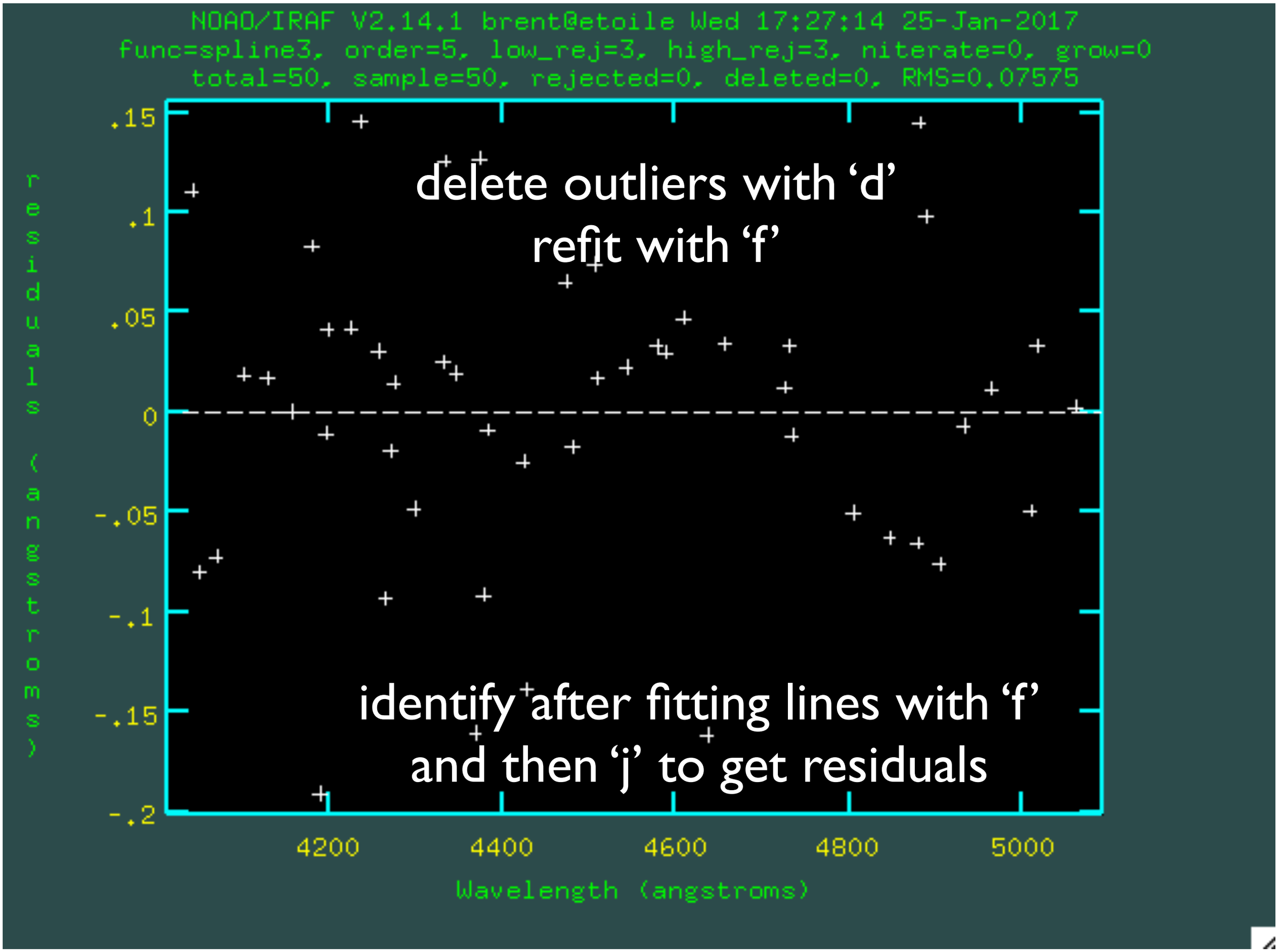


Example: CuAr PG2300; CAMANG 79.0 deg




NOAO/IRAF V2.14.1 brent@etoile Wed 17:26:14 25-Jan-2017
identify arc7[*,201]
ARC
identify first screen after
identifying lines and running 'l'
to identify more lines
4000
3000
2000
1000
0
4200
4400
4600
4800
5000
Wavelength (angstroms)
:labels coord yes


NOAO/IRAF V2.14.1 brent@etoile Wed 17:29:35 25-Jan-2017
func=spline3, order=5, low_rej=3, high_rej=3, niterate=0, grow=0
total=50, sample=50, rejected=0, deleted=0, RMS=0.07575
non-linear component (angstroms)
20
15
10
5
4200
4400
4600
4800
5000
Wavelength (angstroms)
identify after fitting lines with ‘f’
and then ‘l’
You want a nice smooth
parabola here
If there are lots of wiggles, your
identifications are
probably wrong


NOAO/IRAF V2.14.1 brent@etoile Wed 17:27:14 25-Jan-2017
func=spline3, order=5, low_rej=3, high_rej=3, niterate=0, grow=0
total=50, sample=50, rejected=0, deleted=0, RMS=0.07575
delete outliers with ‘d’
refit with ‘f’
identify after fitting lines with ‘f’
and then ‘j’ to get residuals
.15
.1
.05
0
-.05
-.1
-.15
-.2
residuals (angstroms)
4200
4400
4600
4800
5000
Wavelength (angstroms)

Handling all arcs in one night and between nights

- Run identify on `arc1.fits` (first arc)
- ls arc*fits > arcin [an ascii list of arcs in night]
- reidentify ref=`arc1.fits` images=@arcin interac-
- Remember to check RMS (rightmost column)
- To use arc1.fits in another night (if arc1.fits is not already a filename in another night)
- From the other night dir:

```
ls arc*fits > arcin
mkdir database
cp ../0815/arc1.fits .
cp ../0815/database/idarc1 database/
```

copies over arc and its solution for use in new dir

Multiple types of arcs

- Identify first arc of each lamp type as template. Create lists filtered on CAMANG:

- `cat summary.txt | grep ARC | grep 79 | sed 's/.*\([0-9][0-9]\).fits/\1/g' | awk '{printf("arc%s.fits\n",$1)}' > arc79`

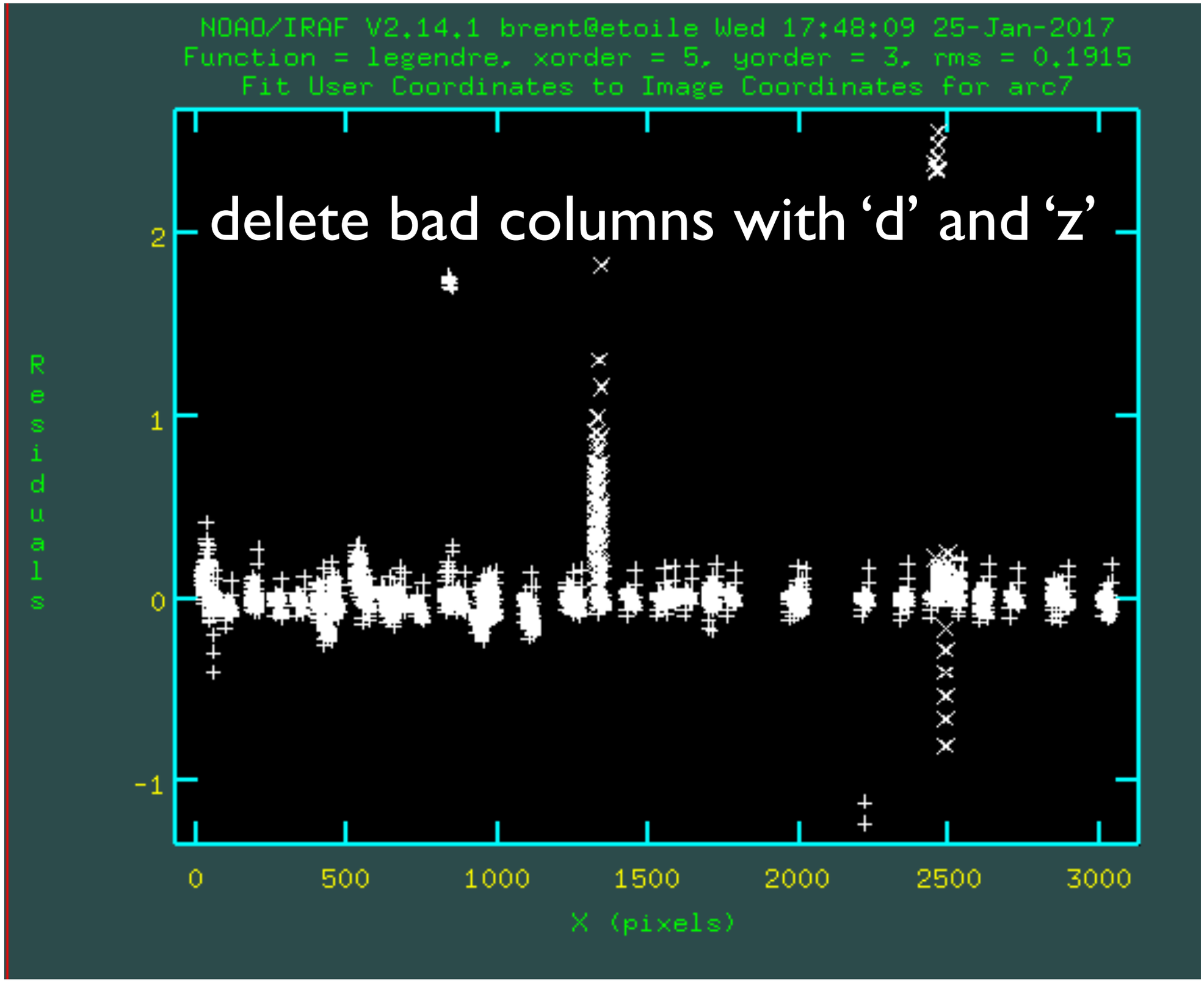
- `cat summary.txt | grep ARC | grep 83.5 | sed 's/.*\([0-9][0-9]\).fits/\1/g' | awk '{printf("arc%s.fits\n",$1)}' > arc83`

- Reidentify in two separate batches: using separate filename lists, arc line identification lists and different arc templates

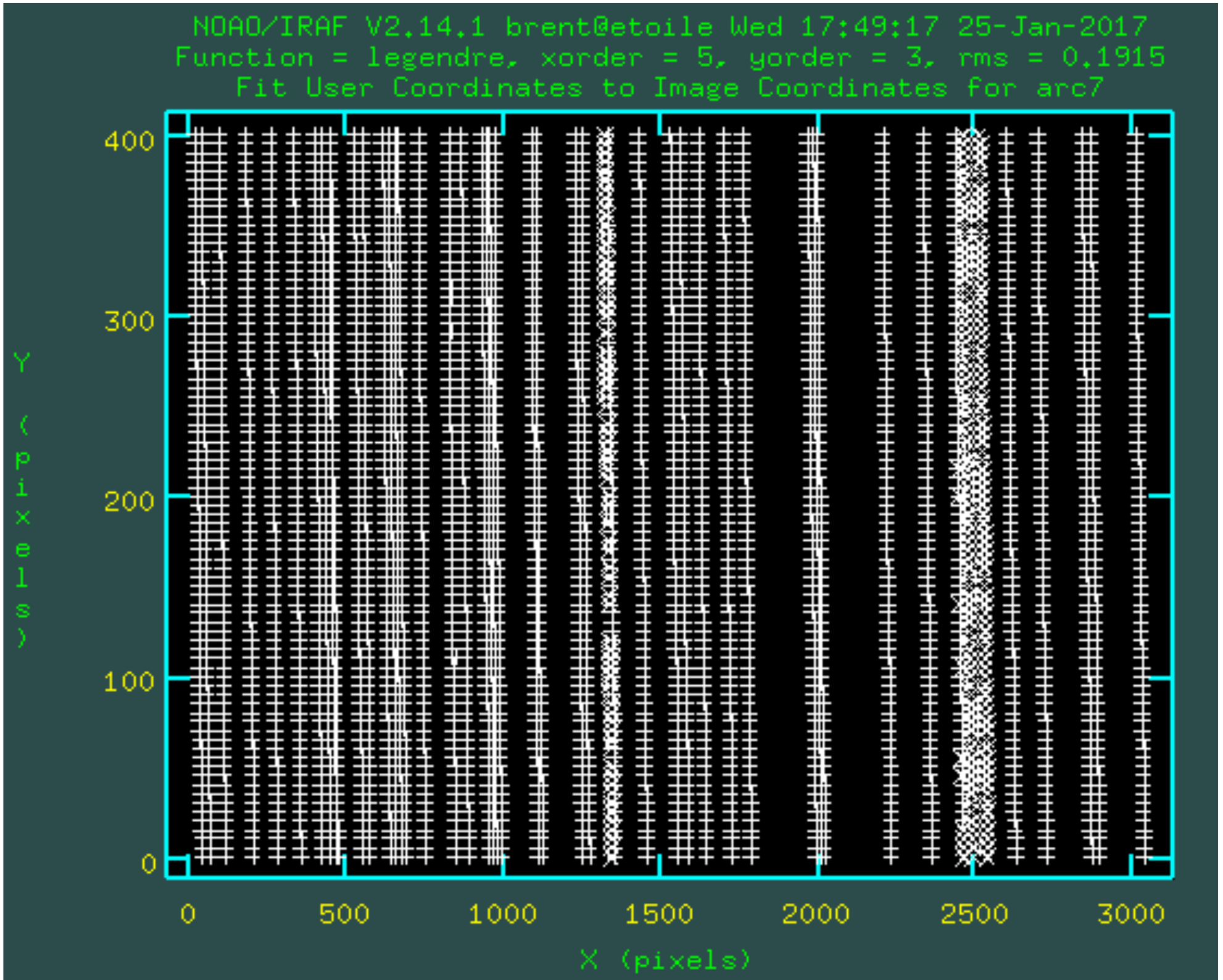
Final arc step

- Load twodspec and longslit IRAF packages
- Run fitcoords arcXX where XX is the arc number, e.g. fitcoords arc13
- fitcoords writes a coordinate map to database, needed to transform the data (next step)

CuAr arc from earlier



CuAr arc from earlier



after typing: ‘x’ ‘x’ ‘y’ ‘y’ you see identified lines

Each mark is identified line in a row group

Transforming science data

- Also in twodspec, longslit package is transform; e.g. input=obj12cc.fits, output=obj12tr.fits and fitnames=arc11,arc13
- Fitnames can be one or more arcs to use for arc solution and transformation.
- [N.B. must leave off '.fits' for fitnames]
- If you have a lot of data in one night, typing all the transform commands can be tedious!
- You can script this up - an example follows, with arcs expected before and after each science exposure

../arcme.pl > arc.cl [in terminal]

cl < arc.cl [in IRAF]

```
#!/usr/bin/perl
use Cwd;
$dir = getcwd();
$dir =~ s/.*\///g;
print "hedit obj*fits DISPAXIS \"1\" add+ ver- show+\n";
@objs = glob "obj*cc*fits";
foreach $o (@objs){
   $o =~ /([0-9]+)/;
   $on = $1;
   $fits = "";
   $an1=$on-1;
   $an2=$on+1;
   print "transform obj${on}cc.fits obj${on}tr.fits fitnames=arc$an1,arc$an2\n";
}

print "cp /Users/brent/login.cl .\n";
print "!/opt/local/bin/python2.7 ../xferheaders.py\n";
```

You now have transformed frames almost ready for science

```
hedit obj*fits DISPAXIS "1" add+ ver- show+
transform obj32cc.fits obj32tr.fits fitnames=arc31,arc33
transform obj44cc.fits obj44tr.fits fitnames=arc43,arc45
transform obj47cc.fits obj47tr.fits fitnames=arc46,arc48
cp /Users/brent/login.cl .
!/opt/local/bin/python2.7 ../xferheaders.py
```

xferheaders.py

```
#!/opt/local/bin/python2.7
from pyraf import iraf
import pyfits
import re
import os, sys, glob, shutil

iraf.images()
iraf.imutil()
rmod = re.compile('\d+')
trlist = glob.glob('*tr*.fits')
for i in range(0,len(trlist)):
   red=trlist[i]
   res = rmod.search(red)
   num=res.group()
   srcglb = "mb*%s.fits" % res.group()
   srclist = glob.glob(srcglb)
   ofname="%s[0]" % srclist[0]

   #Right Ascension (use TELRA since RA is sometimes broken in headers)
   iraf.imgets(image=ofname,param="TELRA")
   RA = iraf.imgets.getParam("value")

   #Declination (use TELDEC since DEC is sometimes broken in headers)
   iraf.imgets(image=ofname,param="TELDEC")
   DEC = iraf.imgets.getParam("value")

   #Sidereal time
   iraf.imgets(image=ofname,param="LST-OBS")
   LST = iraf.imgets.getParam("value")

   #AIRMASS
   iraf.imgets(image=ofname,param="AIRMASS")
   AIRMASS = iraf.imgets.getParam("value")

   #EXPTIME
   iraf.imgets(image=ofname,param="EXPTIME")
   EXPTIME = iraf.imgets.getParam("value")
```

xferheaders.py

```
#UTC
iraf.imgets(image=ofname,param="UTC-OBS")
UT = iraf.imgets.getParam("value")

#JD
iraf.imgets(image=ofname,param="JD")
JD = iraf.imgets.getParam("value")

#DATE-OBS
iraf.imgets(image=ofname,param="DATE-OBS")
DOBS = iraf.imgets.getParam("value")

print red,num,srclist[0],RA,DEC,LST,AIRMASS,EXPTIME,UT,JD,DOBS
hdulist = pyfits.open(red,mode='update')
prihdr = hdulist[0].header
prihdr.set('RA',RA)
prihdr.set('DEC',DEC)
prihdr.set('AIRMASS',AIRMASS)
prihdr.set('ST',LST)
prihdr.set('LST',LST)
prihdr.set('UT',UT)
prihdr.set('UTC',UT)
  prihdr.set('EXPTIME',EXPTIME)
  prihdr.set('JD',JD)
  prihdr.set('DATE-OBS',DOBS)
  hdulist.flush()
  hdulist.close()

f(os.path.exists("pyraf")):
  shutil.rmtree("pyraf")
```

```
etoile:0801 brent$ ../xferheaders.py
Created directory /Users/brent/2015-1/0801/pyraf for cache
obj13tr.fits 13 mbxgpP201508010013.fits 17:07:29.13 -01:39:05.16 16:44:10 1.16890121934402 600.204 18:30:35.302 2457236.27123843 2015-08-01
obj13trf.fits 13 mbxgpP201508010013.fits 17:07:29.13 -01:39:05.16 16:44:10 1.16890121934402 600.204 18:30:35.302 2457236.27123843 2015-08-01
obj14tr.fits 14 mbxgpP201508010014.fits 17:07:29.07 -01:39:35.92 17:02:11 1.16331831473172 600.206 18:48:33.559 2457236.28371528 2015-08-01
obj14trf.fits 14 mbxgpP201508010014.fits 17:07:29.07 -01:39:35.92 17:02:11 1.16331831473172 600.206 18:48:33.559 2457236.28371528 2015-08-01
obj21tr.fits 21 mbxgpP201508010021.fits 15:49:04.47 -16:36:20.72 18:15:12 1.24439926827466 600.206 20:01:21.954 2457236.33427083 2015-08-01
obj21trf.fits 21 mbxgpP201508010021.fits 15:49:04.47 -16:36:20.72 18:15:12 1.24439926827466 600.206 20:01:21.954 2457236.33427083 2015-08-01
obj22tr.fits 22 mbxgpP201508010022.fits 15:49:04.58 -16:36:34.56 18:27:05 1.2856370025688 600.208 20:13:13.365 2457236.34251157 2015-08-01
obj22trf.fits 22 mbxgpP201508010022.fits 15:49:04.58 -16:36:34.56 18:27:05 1.2856370025688 600.208 20:13:13.365 2457236.34251157 2015-08-01
obj35tr.fits 35 mbxgpP201508010035.fits 01:16:48.17 -64:55:11.36 22:46:57 1.29927104589882 600.21 00:32:22.356 2457236.52247685 2015-08-02
obj35trf.fits 35 mbxgpP201508010035.fits 01:16:48.17 -64:55:11.36 22:46:57 1.29927104589882 600.21 00:32:22.356 2457236.52247685 2015-08-02
obj36tr.fits 36 mbxgpP201508010036.fits 01:16:48.87 -64:54:57.79 22:58:30 1.28166618472974 600.211 00:43:52.711 2457236.53046296 2015-08-02
obj36trf.fits 36 mbxgpP201508010036.fits 01:16:48.87 -64:54:57.79 22:58:30 1.28166618472974 600.211 00:43:52.711 2457236.53046296 2015-08-02
```

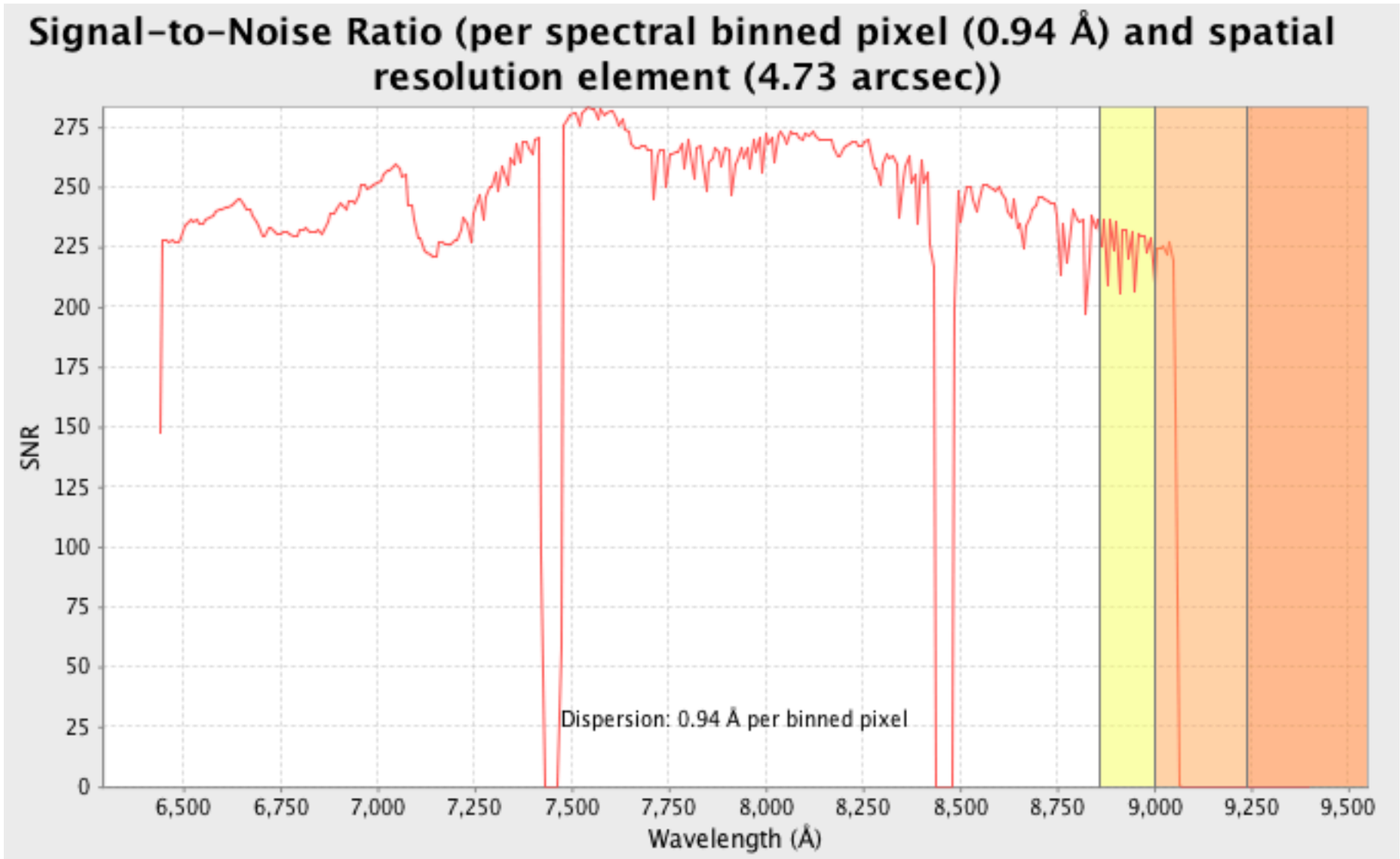
apall: 1D spectra

- Fits function to stellar continuum and traces to correct for tilt in spatial direction
- Use nsum to control number of rows summed for extracted spectrum
- Use lower and upper to choose where relative to trace to sum. Stellar e.g. -10,10 or a nebula component one side of e.g. star 20,30. If no star in slit, use reference from other spec.
- Sky regions can be specified (use near your trace). Full 2D sky subtraction (e.g. skysub) not recommended for objects w/ extended nebulae!

Spectrophotometric calibration

- Use ‘standard’ and ‘sensfunc’, then ‘calibrate’ Reduce spectrophotometric standards in similar way.
- Only relative calibration correct with SALT
- Caveat: sometimes second order contamination from grating cannot be corrected for (e.g. >~ 8800A with PG900)
- Extinction data for Sutherland: see http://astronomers.salt.ac.za/data/data-reduction-faq/#faq_4

Second order contamination

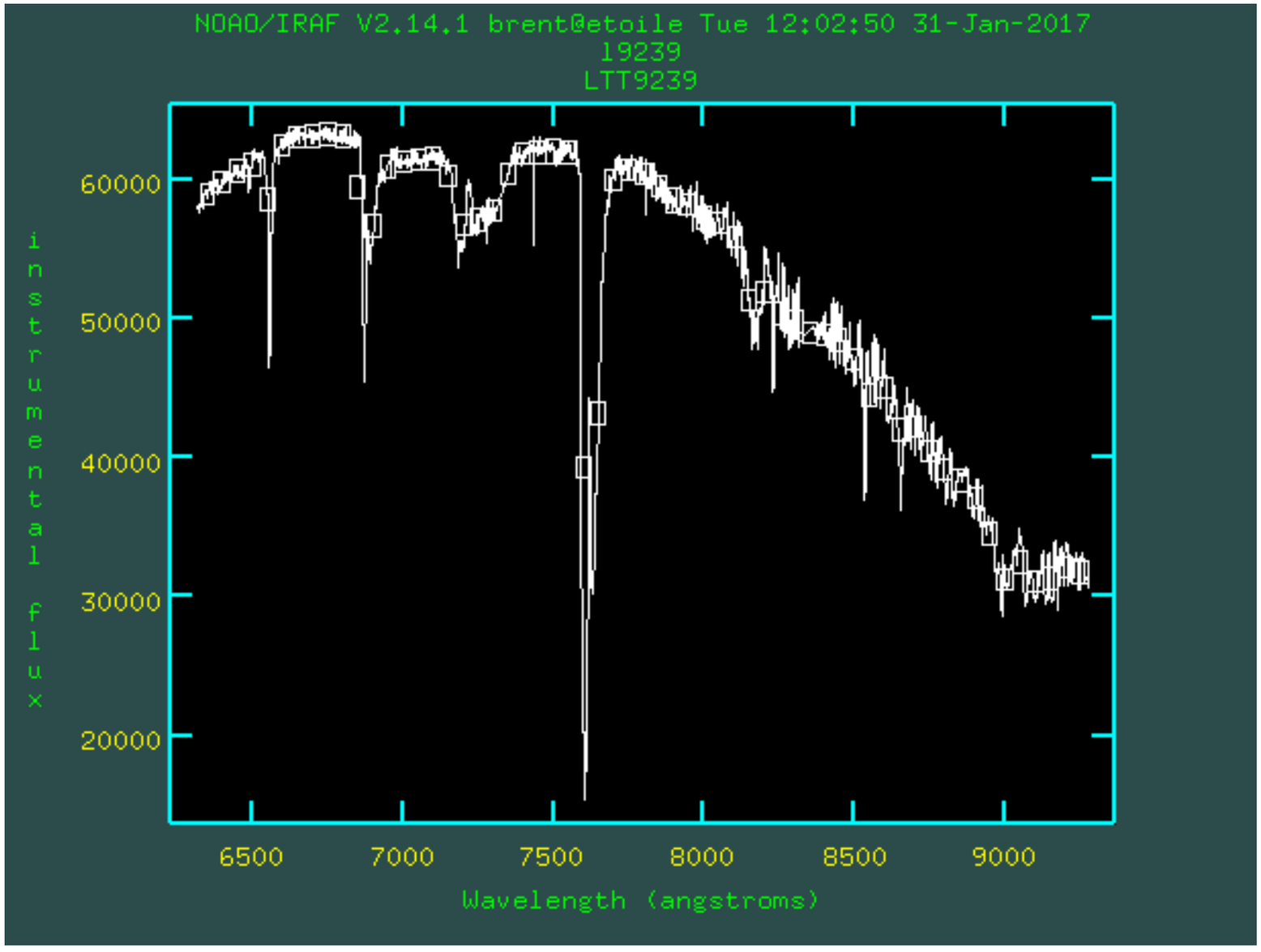


1-10% (yellow)
10-50% (orange)
>50% (dark orange)

best to check your data carefully!

Example: LTT9239 PG900; CAMANG 41.5

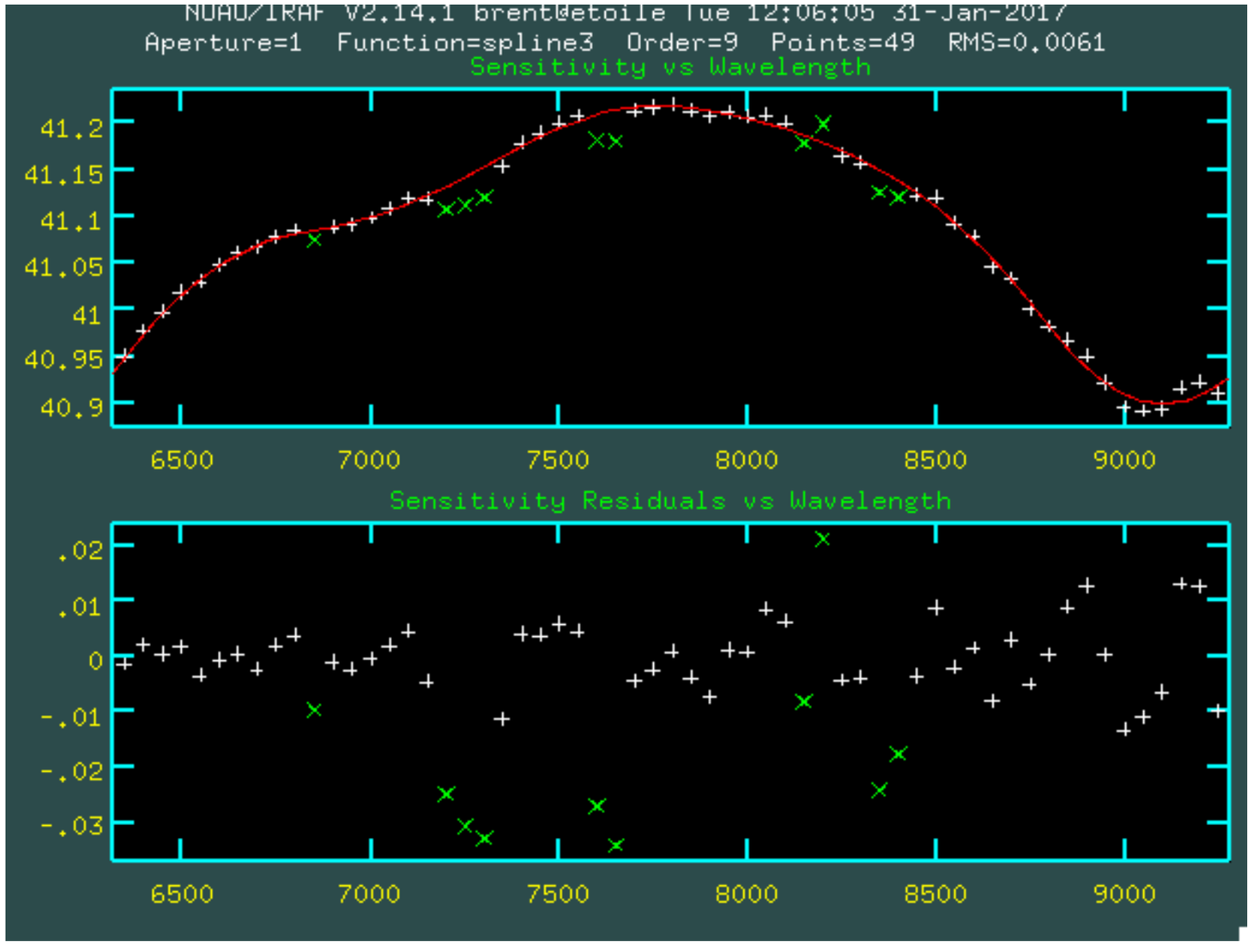
standard



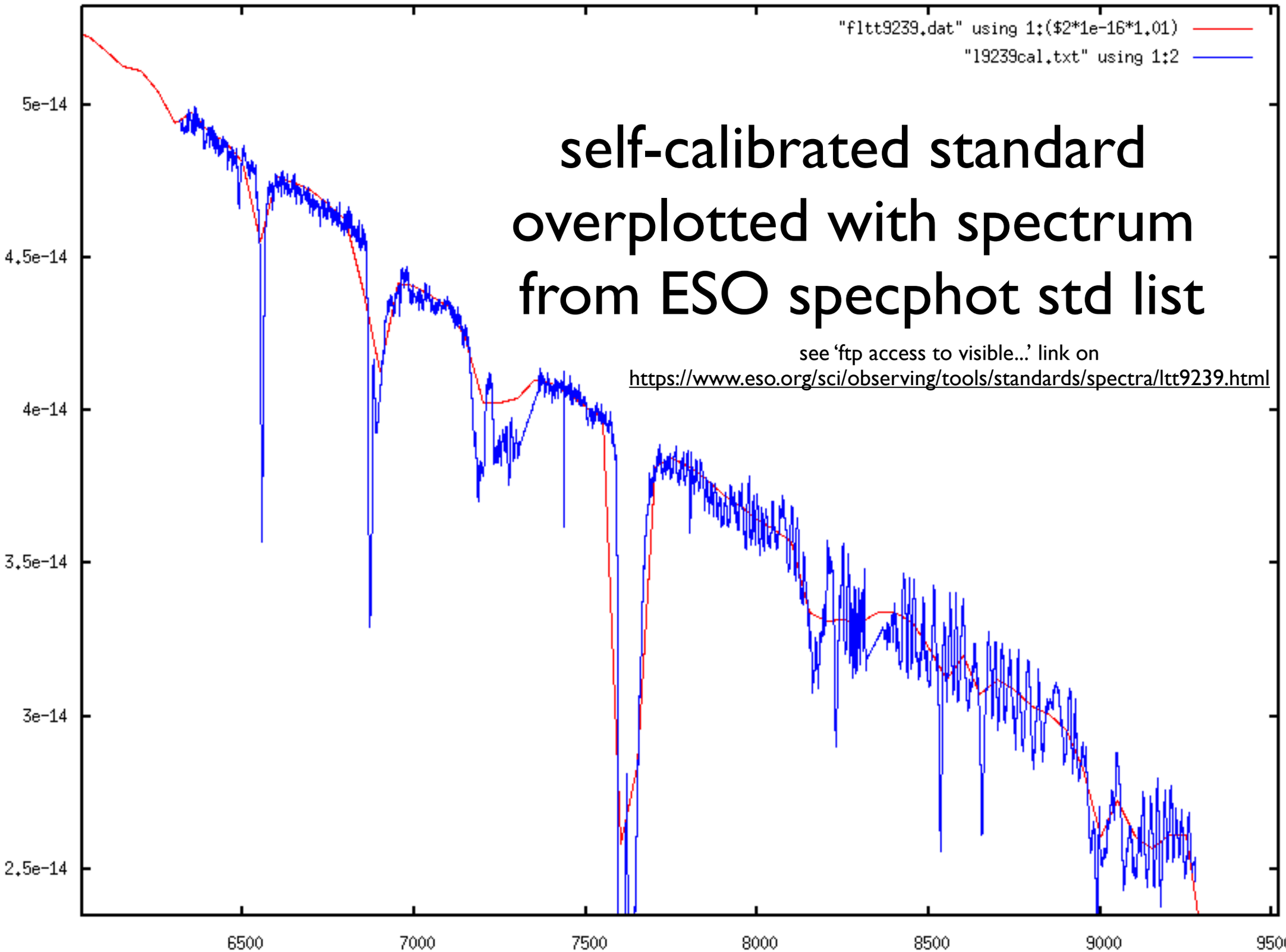
Example: LTT9239 PG900; CAMANG 41.5

sensfunc

green points deleted (telluric)



self calibration check



self-calibrated standard overplotted with spectrum from ESO specphot std list

see 'ftp access to visible...' link on
https://www.eso.org/sci/observing/tools/standards/spectra/ltt9239.html

Filename conventions

- XX is the image number e.g. in mbxgpP20150801 00013 XX=13
- arcXX.fits, objXX.fits, fXX.fits (arc, object and flat) copied out of SCI extension from mbxgpP*fits
- objXcc.fits - cosmic ray cleaned science frame
- objXtr.fits - transformed objXcc.fits with arc solution