AGA0414
Data Reduction

Prof. Alessandro Ederoclite

No imager is perfect

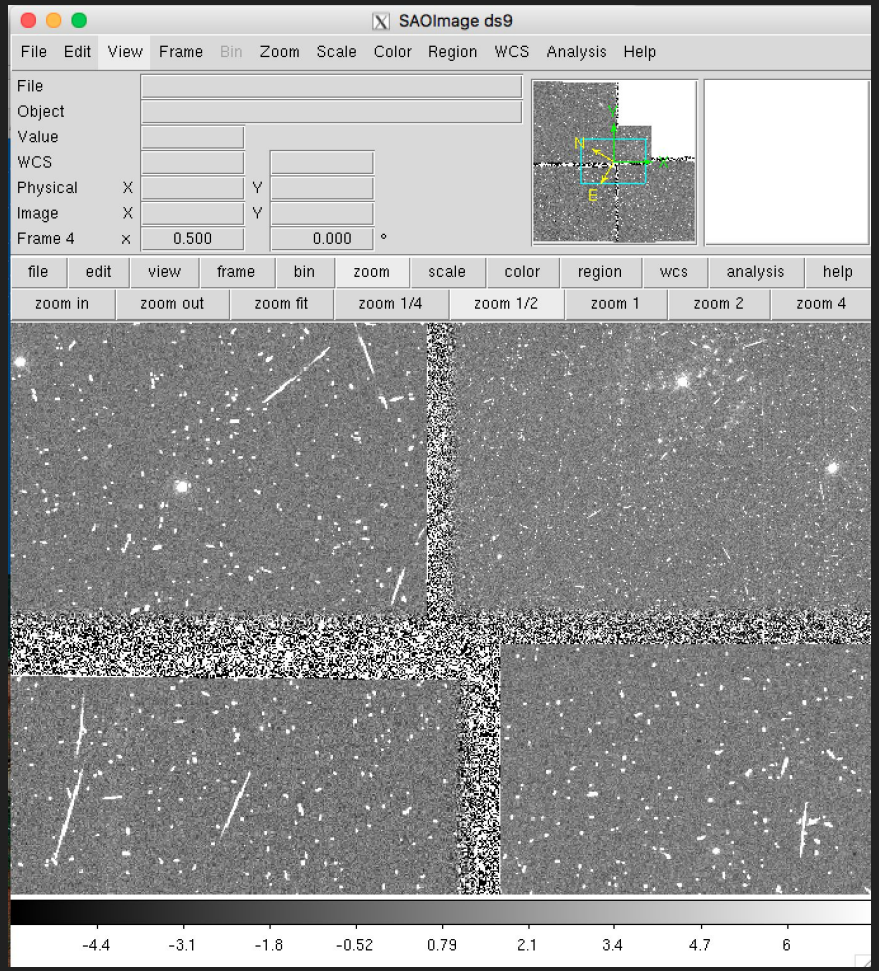
- Background noise
- Differences in illumination of the field of view
- “Sky concentration”
- Fringes
- Cosmic rays

(on the right a raw image from HST/WFPC2)

Every time you touch data, you affect them -> noise increases!

Data reduction is painful and everybody makes mistakes.

“Standard” data reduction is a myth.



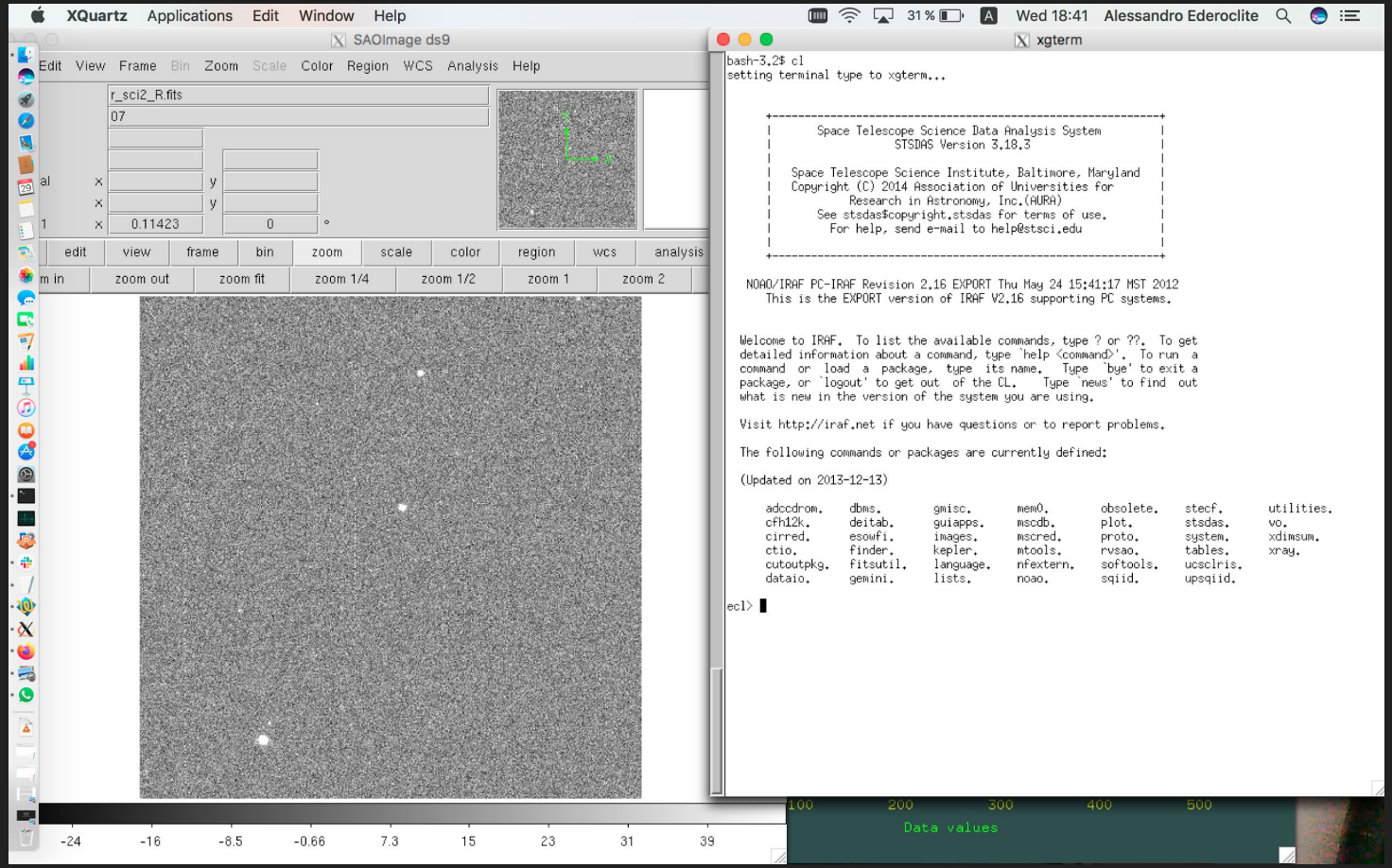
I will now guide you through a full reduction of an image (the IRAF way)

If you have the VM, ds9 and IRAF in the xgterm should open easily.

If you are not running the VM (but have ds9 and IRAF installed on your computer remember):

- Launch `ds9` first
- Go to the directory where your `login.cl` is (you probably have run `mkiraf` there)
- Open the `xgterm` ; I do `xgterm -sb &`
- Make your `xgterm` big (but not too big)
- Start IRAF typing `cl`

My screen looks like this



login.cl

This is the configuration file of IRAF

A couple of important tips

- `stdimage` is the maximum size of image that IRAF will handle, my setting is:

```
set stdimage     = imt4096
```

- To make sure you can use fits:

```
set imtype       = "fits"

set imextn       = "oif:imh fxf:fits,fit fxb:fxb plf:pl qpf:qp stf:hhh,??h"
```

Before you start, check your data

Always important: go to the directory where your data are and check them out:

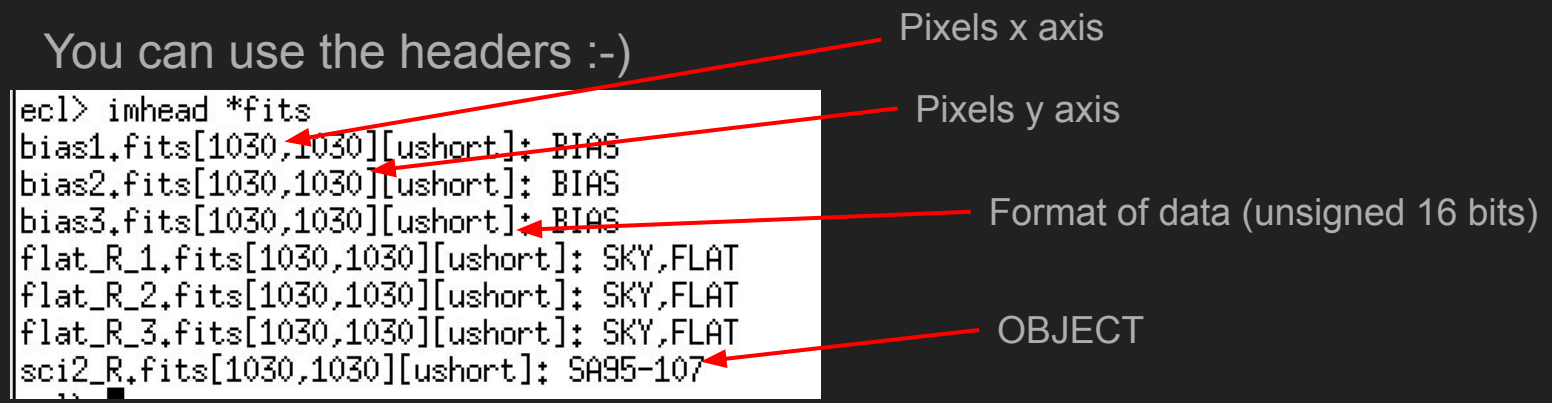
```
ecl> pwd
/Users/alessandroederoclite/iraf
ecl> cd ../Desktop/USP/Lectures/UndergraduateCourses/lectures_2020A/12_datareduction/work/
```

A good thing is to get a list of the content of your directory

```
ecl> ls
bias1.fits    bias2.fits    bias3.fits    flat_R_1.fits flat_R_2.fits flat_R_3.fits sci2_R.fits
```

You can use the headers :-)

```
ecl> imhead *fits
bias1.fits[1030,1030][ushort]: BIAS
bias2.fits[1030,1030][ushort]: BIAS
bias3.fits[1030,1030][ushort]: BIAS
flat_R_1.fits[1030,1030][ushort]: SKY,FLAT
flat_R_2.fits[1030,1030][ushort]: SKY,FLAT
flat_R_3.fits[1030,1030][ushort]: SKY,FLAT
sci2_R.fits[1030,1030][ushort]: SA95-107
```



Let’s do some statistics!

Let’s check the properties of the images:

```
ecl> imstat *fits
#                IMAGE      NPIX      MEAN    STIDEV       MIN       MAX
           bias1.fits   1060900     219.6     11.23        0.      3190.
           bias2.fits   1060900     222.6     13.06        0.      5094.
           bias3.fits   1060900     223.5      9.46        0.       660.
       flat_R_1.fits   1060900     10315.     1488.        0.     30248.
       flat_R_2.fits   1060900     37472.     5403.        0.     65535.
       flat_R_3.fits   1060900     37207.     5365.        0.     65535.
         sci2_R.fits   1060900     230.2      23.8         0.      6920.
```

Imstat is one of my favourite commands! It’s very powerful but…

...with great power, comes great responsibility

We can check the parameters that a command accepts with lpar

If you want to modify the ones in parenthesis, you will have to do it explicitly
We will see how

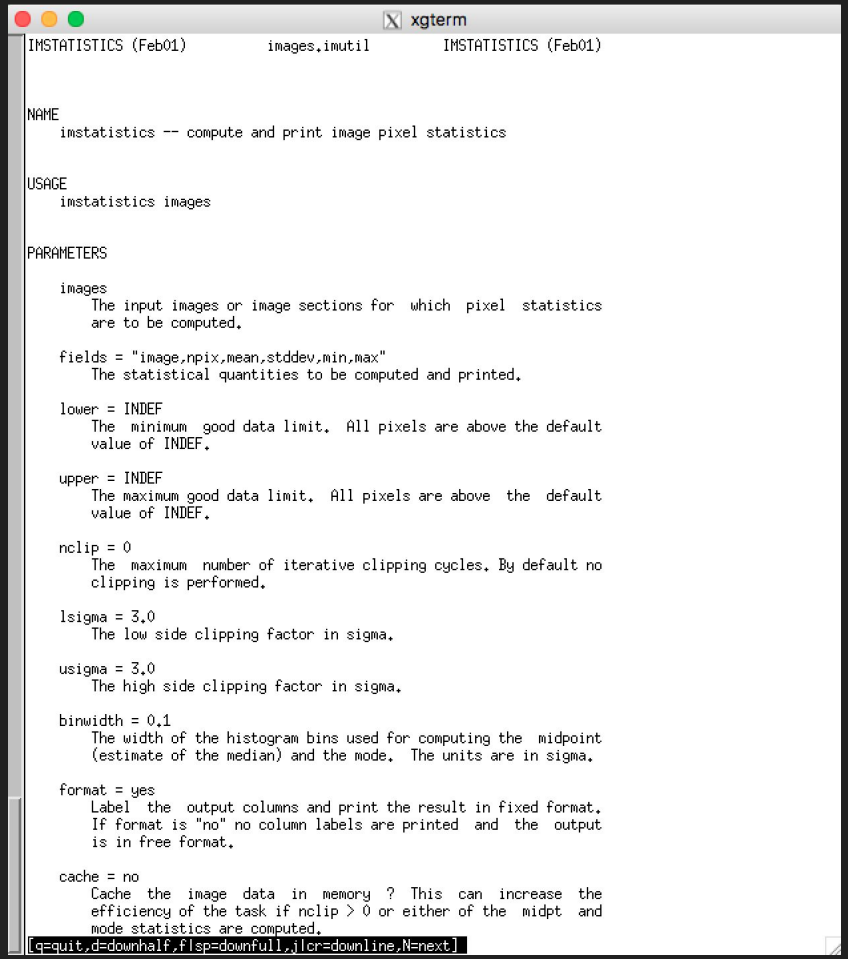
```
ecl> lpar imstat
       images = "*fits"          List of input images
     (fields = "image,npix,mean,stddev,min,max") Fields to be printed
      (lower = INDEF)            Lower limit for pixel values
      (upper = INDEF)            Upper limit for pixel values
      (nclip = 0)                Number of clipping iterations
     (lsigma = 3.)               Lower side clipping factor in sigma
     (usigma = 3.)               Upper side clipping factor in sigma
   (binwidth = 0.1)              Bin width of histogram in sigma
     (format = yes)              Format output and print column labels ?
      (cache = no)               Cache image in memory ?
       (mode = "ql")
```

Help!

`help imstat`

Will give you a more complete insight on the command

(needless to say, this applies to any IRAF command)



Last on imstat

I hate that “NPIX” and I think mode and median are useful.

Here’s something I find more useful:

```
ecl> imstat *fits fields="image,mean,mode,midpt,stddev,min,max"
#                IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
           bias1.fits     219.6     219.5     219.5     11.23        0.      3190.
           bias2.fits     222.6     222.6     222.7     13.06        0.      5094.
           bias3.fits     223.5      222.     223.3      9.46        0.       660.
         flat_R_1.fits    10315.    10517.    10516.    1488.        0.     30248.
         flat_R_2.fits    37472.    38171.    38182.    5403.        0.     65535.
         flat_R_3.fits    37207.    37945.    37959.    5365.        0.     65535.
           sci2_R.fits     230.2     231.9     230.4     23.8        0.      6920.
```

display

To display an image in ds9 from IRAF, we use the command “display”

Ds9 has several “frames” we can use (we can show several images at the same time)

I also like to “fill” the frame (so I see the full image)

```
ecl> lpar display
        image = "bias1.fits"    image to be displayed
        frame = 1               frame to be written into
      (bpmask = "BPM")          bad pixel mask
   (bpdisplay = "none")         bad pixel display (none|overlay|interpolate)
    (bpcolors = "red")          bad pixel colors
     (overlay = "")             overlay mask
     (ocolors = "green")        overlay colors
       (erase = yes)            erase frame
(border_erase = no)             erase unfilled area of window
(select_frame = yes)            display frame being loaded
      (repeat = no)             repeat previous display parameters
        (fill = no)             scale image to fit display window
      (zscale = yes)            display range of greylevels near median
    (contrast = 0.25)           contrast adjustment for zscale algorithm
     (zrange = yes)             display full image intensity range
      (zmask = "")              sample mask
     (nsample = 1000)           maximum number of sample pixels to use
     (xcenter = 0.5)            display window horizontal center
     (ycenter = 0.5)            display window vertical center
       (xsize = 1.)             display window horizontal size
       (ysize = 1.)             display window vertical size
        (xmag = 1.)             display window horizontal magnification
        (ymag = 1.)             display window vertical magnification
       (order = 0)              spatial interpolator order (0=replicate, 1=linear)
          (z1 = )               minimum greylevel to be displayed
          (z2 = )               maximum greylevel to be displayed
     (ztrans = "linear")        greylevel transformation (linear|log|none|user)
     (lutfile = "")             file containing user defined look up table
        (mode = "ql")
```

display

The “complete” way to use display is:

```
ecl> display bias1.fits 1 fill=yes
z1=184. z2=246.
```

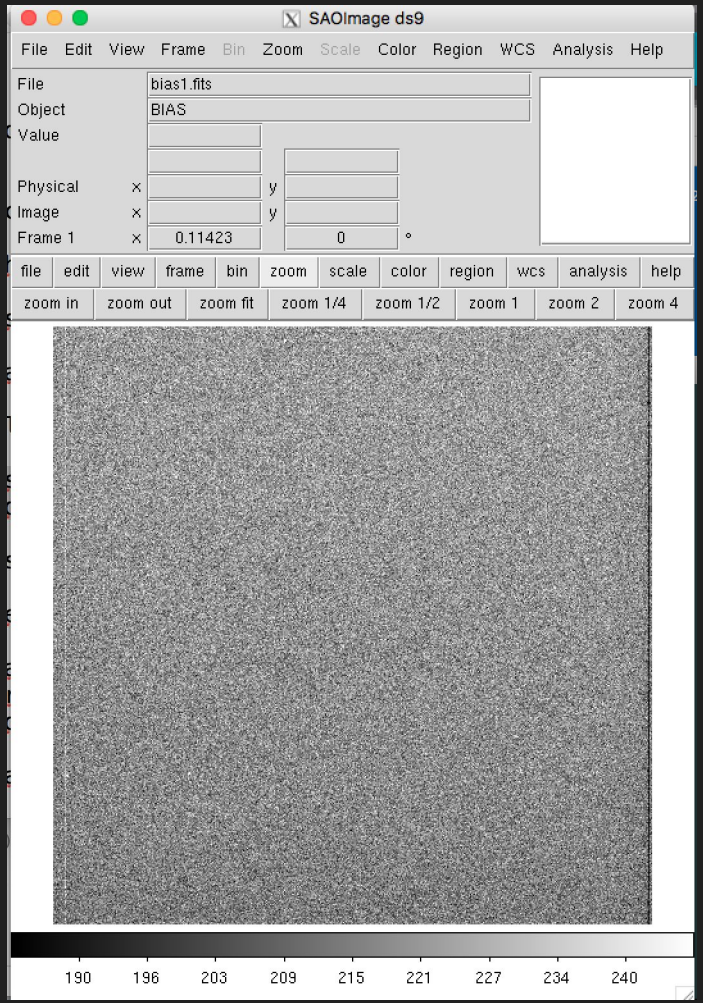
The numbers underneath are the minimum and maximum which are shown in the image

Friends invoke display like:

```
ecl> displ bias1.fits 1 fi+
z1=184. z2=246.
```

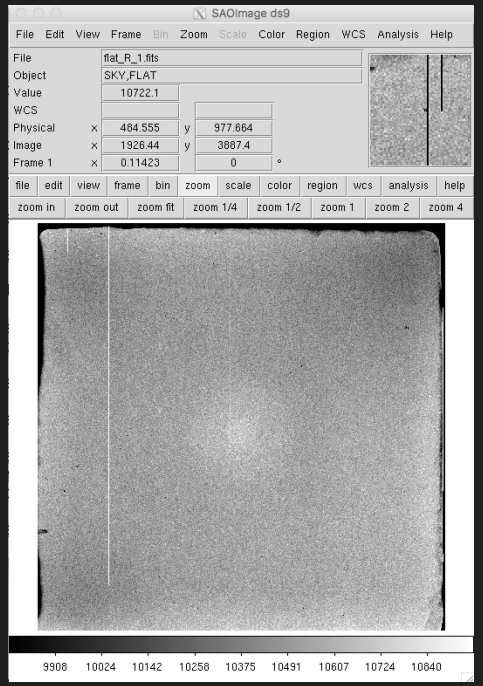
“=yes” can be replaced by “+”

You can replace complete names with abbreviations if it’s unambiguous

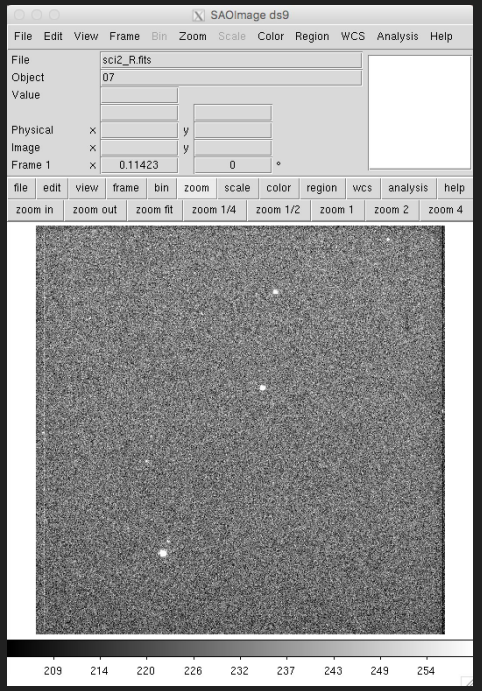


First time, it is a good practice to look at everything!

```
ecl> displ flat_R_1.fits 1 fi+
z1=9792.322 z2=10956.
```




```
ecl> displ sci2_R.fits 1 fi+
z1=203. z2=260.
```



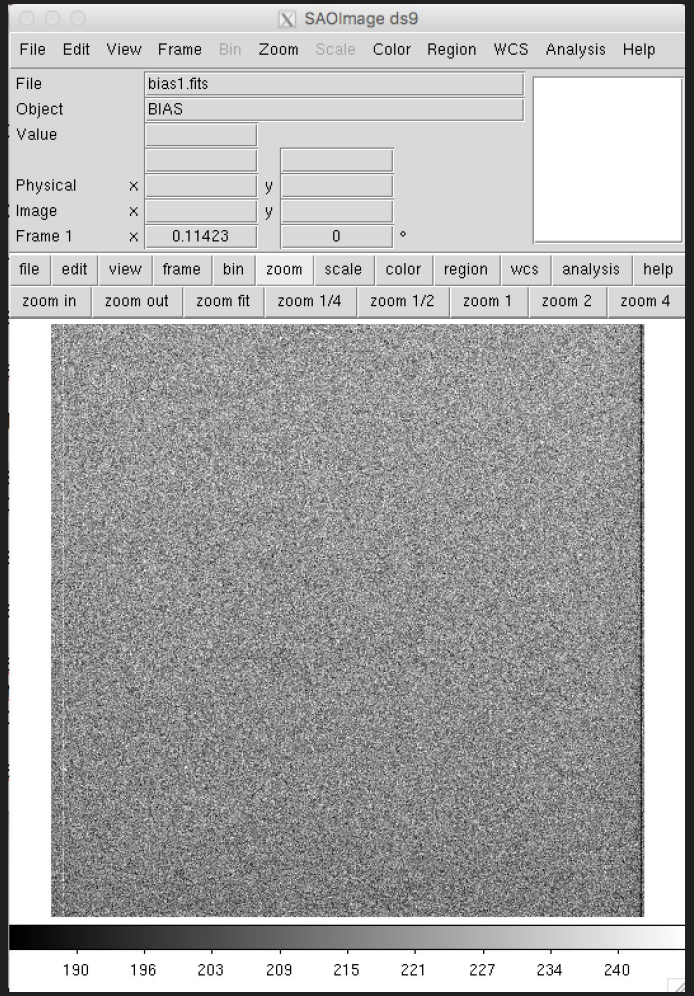
Background Noise

The “bias”

What happens if you take a “0 seconds exposure”?

Wait a second! What do you mean by “0 seconds”?

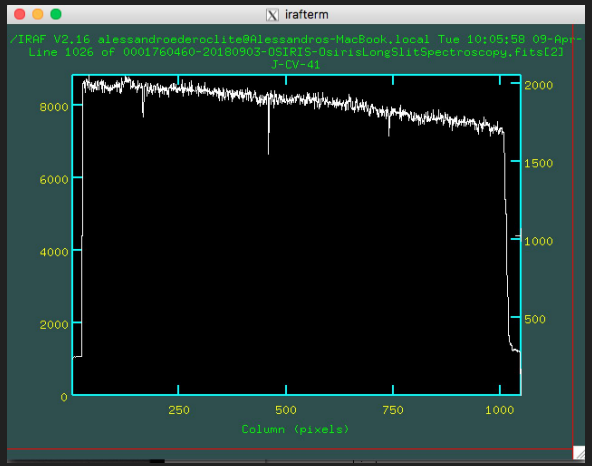
A “bias” is the readout of a CCD without collecting photons.

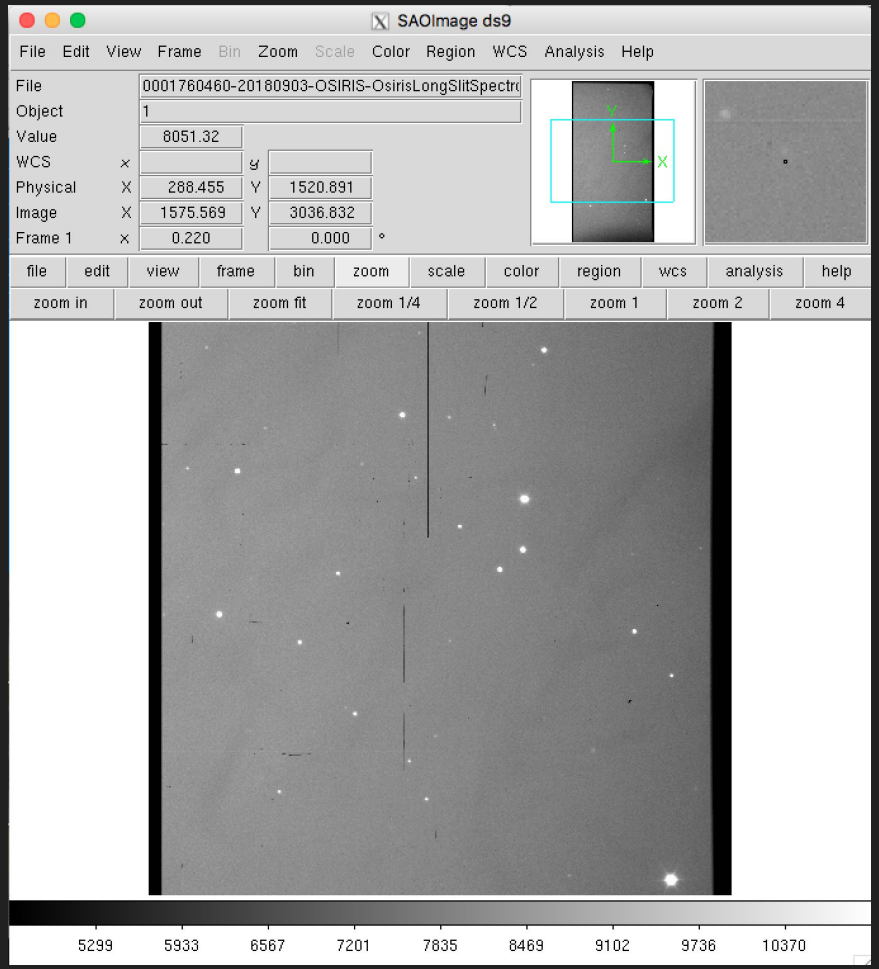


The “overscan”

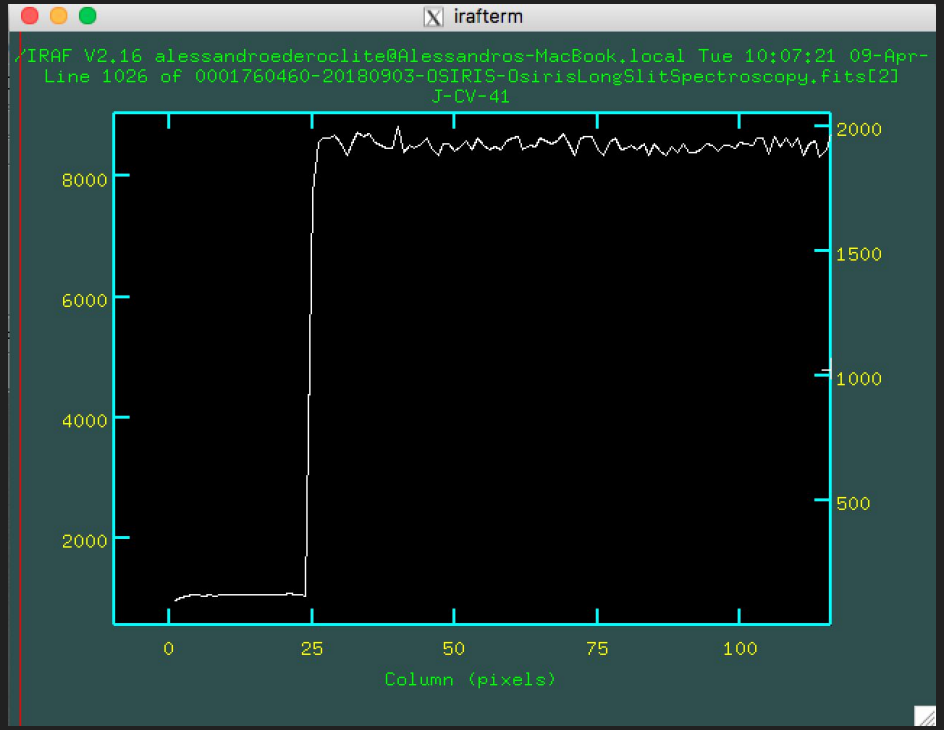
A region “on the side” of the image, where “empty” readouts of the electronics are made.

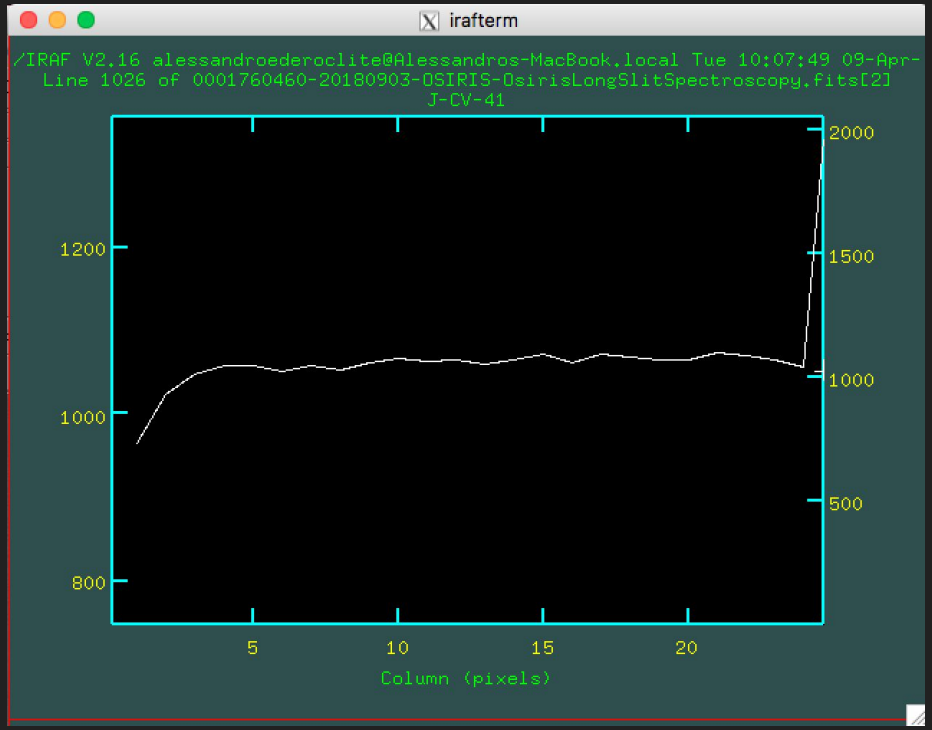
(these are cuts along lines)





The “overscan”





Bias vs Overscan

- Bias takes into account the 2D variations of noise
- Overscan measures the noise of your image

Not all CCDs have overscan (or prescan) regions.

Bias vs Overscan

	Pros	Cons
Bias	2D mapping of the noise of the CCD	Not simultaneous with observations
Overscan	Obtained together with observations	It is a 1D approximation to the noise of the CCD. Not all CCDs allow for an overscan region.

Make a “masterbias”

To get a good measure of the bias, we take several bias frames and we combine them.

For this, we need to load the imred package (if you see a “.”, it’s a sub-package; if you see an “@”, it’s a parameter file; if you see nothing else, it’s a command)

```
ecl> noao
      artdata.     astutil.     imred.       nproto.      onedspec.    twodspec.
      astcat.      digiphot.    mtlocal.     observatory  rv.
      astrometry.  focas.       nobsolete.   obsutil.     surfphot.

noao> imred
      argus.       crutil.      echelle.     iids.        kpnocoude.   specred.
      bias.        ctioslit.    generic.     irred.       kpnoslit.    vtel.
      ccdred.      dtoi.        hydra.       irs.         quadred.

imred> ccdred
      badpiximage    ccdinstrument  ccdproc        darkcombine    mkillumcor     mkskyflat
      ccdgroups      ccdlist        ccdtest.       flatcombine    mkillumflat    setinstrument
      ccdhedit       ccdmask        combine        mkfringecor    mkskycor       zerocombine
```

We want to combine bias frames

In IRAF, a bias is called “zero” (guess why?)

Hence we want to use the zerocombine command.

Let’s get to know it:

```
ccdred> lpar zerocombine
       input = "bias*fits"      List of zero level images to combine
      (output = "Zero")         Output zero level name
     (combine = "average")      Type of combine operation
      (reject = "minmax")       Type of rejection
     (ccdtype = "zero")         CCD image type to combine
     (process = no)             Process images before combining?
      (delete = no)             Delete input images after combining?
     (clobber = no)             Clobber existing output image?
       (scale = "none")         Image scaling
     (statsec = "")             Image section for computing statistics
        (nlow = 0)              minmax: Number of low pixels to reject
       (nhigh = 1)              minmax: Number of high pixels to reject
       (nkeep = 1)              Minimum to keep (pos) or maximum to reject (neg)
       (mclip = yes)            Use median in sigma clipping algorithms?
      (lsigma = 3.)             Lower sigma clipping factor
      (hsigma = 3.)             Upper sigma clipping factor
     (rdnoise = "0.")           ccdclip: CCD readout noise (electrons)
        (gain = "1.")           ccdclip: CCD gain (electrons/DN)
      (snoise = "0.")           ccdclip: Sensitivity noise (fraction)
       (pclip = -0.5)           pclip: Percentile clipping parameter
       (blank = 0.)             Value if there are no pixels
       (mode = "ql")
```

Let’s run zerocombine

```
ccdred> zerocombine bias*fits output=masterbias.fits ccdtype=""
ccdred>
```

(the `ccdtype = ""` is a sad story)

It looks pretty dull, eh?

Let’s see what it did:

```
ccdred> imstat *ias* fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
          bias1.fits     219.6     219.5     219.5     11.23        0.      3190.
          bias2.fits     222.6     222.6     222.7     13.06        0.      5094.
          bias3.fits     223.5      222.     223.3      9.46        0.       660.
      masterbias.fits     217.8     218.4     217.7     6.196        0.       247.
ccdred>
```

The flat field

It is related with the difference of response across the field of view.

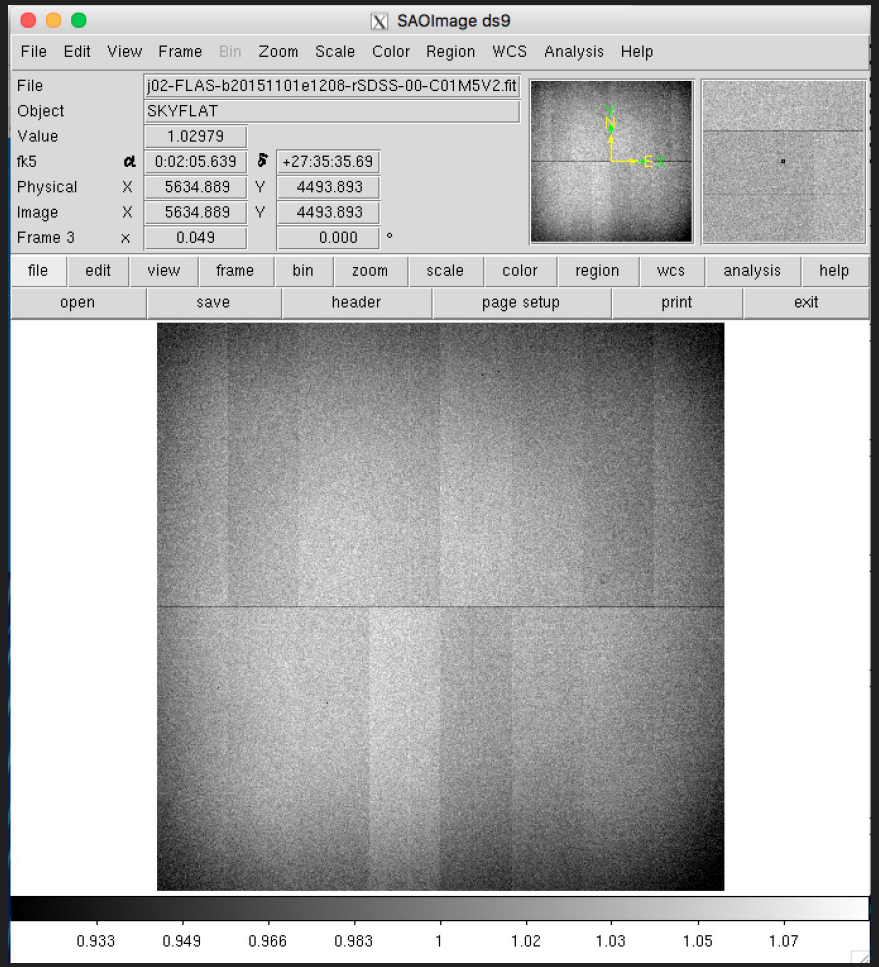
Famous example: dust particles on filters.

Corrected by observing a “flat field” (hence the name): sky vs. dome.

It is a multiplicative effect.

Mind the division by zero!

(happens often at the borders)



Make a masterflat

Check the flats:

```
ccdred> imstat *lat* fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
        flat_R_1.fits    10315.    10517.    10516.     1488.        0.    30248.
        flat_R_2.fits    37472.    38171.    38182.     5403.        0.    65535.
        flat_R_3.fits    37207.    37945.    37959.     5365.        0.    65535.
```

You normally want a flat to have counts of about the half well (~35,000); in this case, only 2 fulfill this criteria. Let's make a list (the "@" lets you access a list of files.

```
ccdred> ls flat_R_2.fits flat_R_3.fits > flats.lis
ccdred> imstat @flats.lis fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
        flat_R_2.fits    37472.    38171.    38182.     5403.        0.    65535.
        flat_R_3.fits    37207.    37945.    37959.     5365.        0.    65535.
```

Subtract the bias

Mind you: a **flat** has a bias

You want to subtract it before combining the flats.

We use the almighty ccdproc

(the longer the parameter list, the more powerful the task)

```
ccdred> lpar ccdproc
      images = "sci2_R.fits"   List of CCD images to correct
      (output = "")            List of output CCD images
     (ccdtype = "")            CCD image type to correct
   (max_cache = 0)             Maximum image caching memory (in Mbytes)
     (noproc = no)             List processing steps only?\n
     (fixpix = no)             Fix bad CCD lines and columns?
    (overscan = no)            Apply overscan strip correction?
       (trim = no)             Trim the image?
     (zerocor = no)            Apply zero level correction?
     (darkcor = no)            Apply dark count correction?
     (flatcor = no)            Apply flat field correction?
    (illumcor = no)            Apply illumination correction?
   (fringecor = no)            Apply fringe correction?
     (readcor = no)            Convert zero level image to readout correction?
     (scancor = no)            Convert flat field image to scan correction?\n
    (readaxis = "line")        Read out axis (column|line)
     (fixfile = "")            File describing the bad lines and columns
     (biassec = "")            Overscan strip image section
     (trimsec = "")            Trim data section
        (zero = "")            Zero level calibration image
        (dark = "")            Dark count calibration image
        (flat = "")            Flat field images
       (illum = "")            Illumination correction images
      (fringe = "")            Fringe correction images
  (minreplace = 1.)            Minimum flat field value
    (scantype = "shortscan")   Scan type (shortscan|longscan)
       (nscan = 1)             Number of short scan lines\n
 (interactive = no)            Fit overscan interactively?
    (function = "legendre")    Fitting function
       (order = 1)             Number of polynomial terms or spline pieces
      (sample = "*")           Sample points to fit
    (naverage = 1)             Number of sample points to combine
    (niterate = 1)             Number of rejection iterations
  (low_reject = 3.)            Low sigma rejection factor
 (high_reject = 3.)            High sigma rejection factor
        (grow = 0.)            Rejection growing radius
        (mode = "ql")
```

```
ccdred> ccdproc @flats.lis output=b_//@flats.lis zerocor+ zero=masterbias.fits
```

The “//” allows you to add some text before the name of the files in the list.

Let’s check what happened:

```
ccdred> imstat *lat* fields="image,mean,mode,midpt,stddev,min,max"
#                IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
      b_flat_R_2.fits    37255.    37926.    37933.     5403.    -229.5     65364.
      b_flat_R_3.fits    36989.    37695.    37708.     5366.    -229.5     65364.
        flat_R_1.fits    10315.    10517.    10516.     1488.        0.     30248.
        flat_R_2.fits    37472.    38171.    38182.     5403.        0.     65535.
        flat_R_3.fits    37207.    37945.    37959.     5365.        0.     65535.
Error reading image flats.lis ...
```

Let’s combine flats

You can probably guess, by now, that we will need a command called `flatcombine`

Here are the parameters ->

```
ccdred> lpar flatcombine
       input = "b_//@flats.lis" List of flat field images to combine
      (output = "Flat")         Output flat field root name
     (combine = "average")      Type of combine operation
      (reject = "avsigclip")    Type of rejection
     (ccdtype = "")             CCD image type to combine
     (process = no)             Process images before combining?
     (subsets = no)             Combine images by subset parameter?
      (delete = no)             Delete input images after combining?
     (clobber = no)             Clobber existing output image?
       (scale = "mode")         Image scaling
     (statsec = "")             Image section for computing statistics
        (nlow = 1)              minmax: Number of low pixels to reject
       (nhigh = 1)              minmax: Number of high pixels to reject
       (nkeep = 1)              Minimum to keep (pos) or maximum to reject (neg)
       (mclip = yes)            Use median in sigma clipping algorithms?
      (lsigma = 3.)             Lower sigma clipping factor
      (hsigma = 3.)             Upper sigma clipping factor
     (rdnoise = "0.")           ccdclip: CCD readout noise (electrons)
        (gain = "1.")           ccdclip: CCD gain (electrons/DN)
      (snoise = "0.")           ccdclip: Sensitivity noise (fraction)
       (pclip = -0.5)           pclip: Percentile clipping parameter
       (blank = 1.)             Value if there are no pixels
        (mode = "ql")
```

Let’s run it!

```
ccdred> flatcombine b_//@flats.lis output=masterflat_R.fits
```

As usual… let’s check

```
ccdred> imstat *lat* fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
      b_flat_R_2.fits     37255.     37926.     37933.      5403.    -229.5     65364.
      b_flat_R_3.fits     36989.     37695.     37708.      5366.    -229.5     65364.
        flat_R_1.fits     10315.     10517.     10516.      1488.        0.     30248.
        flat_R_2.fits     37472.     38171.     38182.      5403.        0.     65535.
        flat_R_3.fits     37207.     37945.     37959.      5365.        0.     65535.
Error reading image flats.lis ...
    masterflat_R.fits     37122.     37710.     37688.      5383.    -229.5     65365.
```

Of course, `flats.lis` is not an image, hence you get an error

We have few flats, so we don’t get a very very good flat.

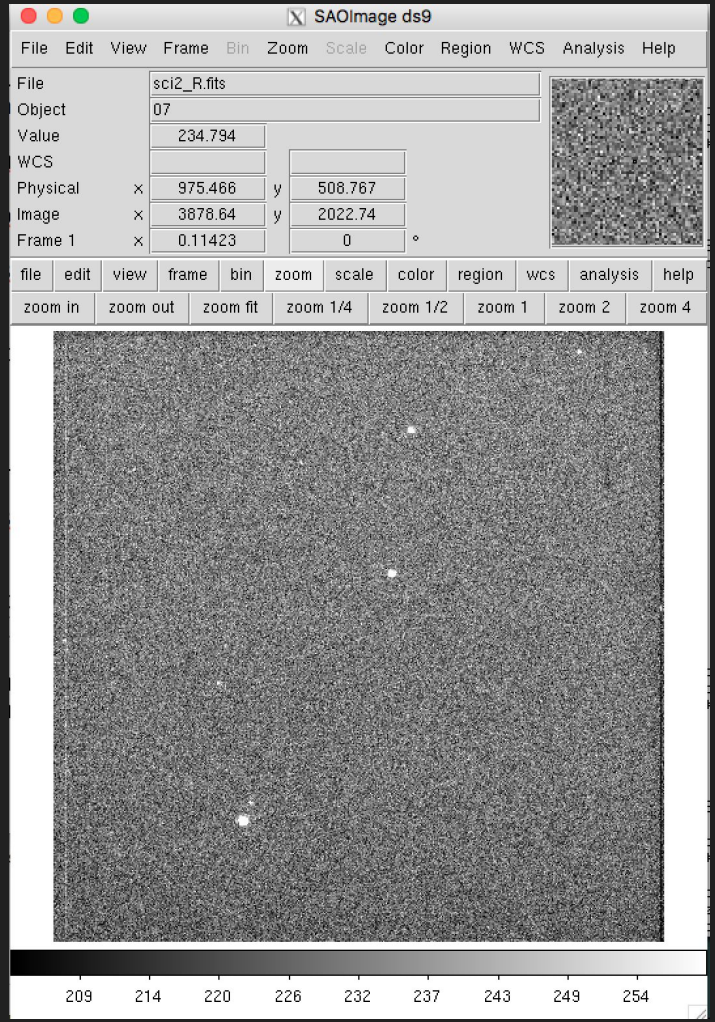
Reduce the science frame!

We are finally there!

Let’s check our science frame (it’s actually the field of a photometric standard star; we will see what these are soon)

```
ccdred> imstat sci2_R.fits fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
         sci2_R.fits     230.2     231.9     230.4      23.8        0.      6920.
ccdred> displ sci2_R.fits 1 fi+
z1=203. z2=260.
```

We will use our old friend `ccdproc`



```
ccdred> ccdproc sci2_R.fits output=r_sci2_R.fits zerocor+ flatcor+ \
>>> zero=masterbias.fits flat=masterflat_R.fits
ERROR: floating point invalid operation
```

What do you mean “error”? Do you know who I am?

2 things:

1) The “\” is used to go to a new line
2) Remember I said that you may get a zero at the edges of the image? It is exactly what happened! We need to “trim” the image.

First delete the reduced image that IRAF tried to create.

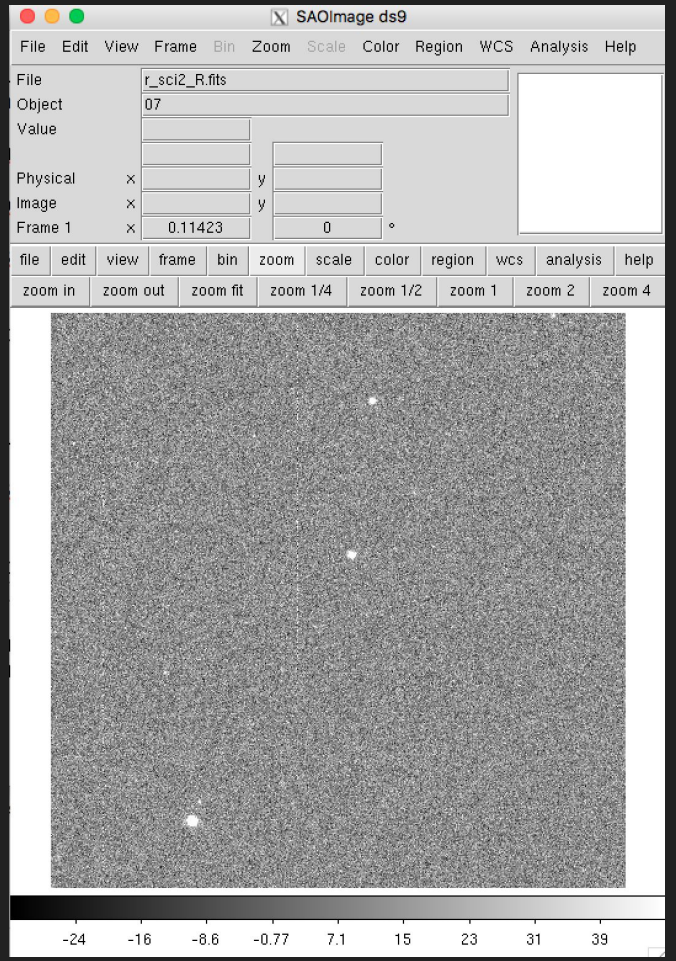
```
ccdred> del r_sci2_R.fits
ccdred> ccdproc sci2_R.fits output=r_sci2_R.fits zerocor+ flatcor+ \
zero=masterbias.fits flat=masterflat_R.fits trim+ trimsec=[100:1000,100:1000]
```

Then do the right thing and say that you want to trim the image and define the "trim section":

```
[x of lower left corner : x lower right corner , y upper left corner : y upper right corner]
```

Let's see our masterpiece

```
ccdred> imstat *.fits fields="image,mean,mode,midpt,stddev,min,max"
#               IMAGE      MEAN      MODE     MIDPT    STDDEV       MIN       MAX
      b_flat_R_2.fits     37255.     37926.     37933.     5403.    -229.5     65364.
      b_flat_R_3.fits     36989.     37695.     37708.     5366.    -229.5     65364.
           bias1.fits      219.6      219.5      219.5     11.23        0.      3190.
           bias2.fits      222.6      222.6      222.7     13.06        0.      5094.
           bias3.fits      223.5       222.      223.3      9.46        0.       660.
         flat_R_1.fits     10315.     10517.     10516.     1488.        0.     30248.
         flat_R_2.fits     37472.     38171.     38182.     5403.        0.     65535.
         flat_R_3.fits     37207.     37945.     37959.     5365.        0.     65535.
       masterbias.fits      218.3      217.9       218.    6.003      175.      244.5
    masterflat_R.fits     37910.     37931.     37927.     1059.     15881.     58180.
        r_sci2_R.fits      12.58      14.31      12.37     26.27    -81.83     6688.
          sci2_R.fits      230.2      231.9      230.4      23.8        0.     6920.
ccdred> displ r_sci2_R.fits 1 fi+
z1=-32.14126 z2=46.33546
```



Congratulations! You have reduced your first image!

Some more steps

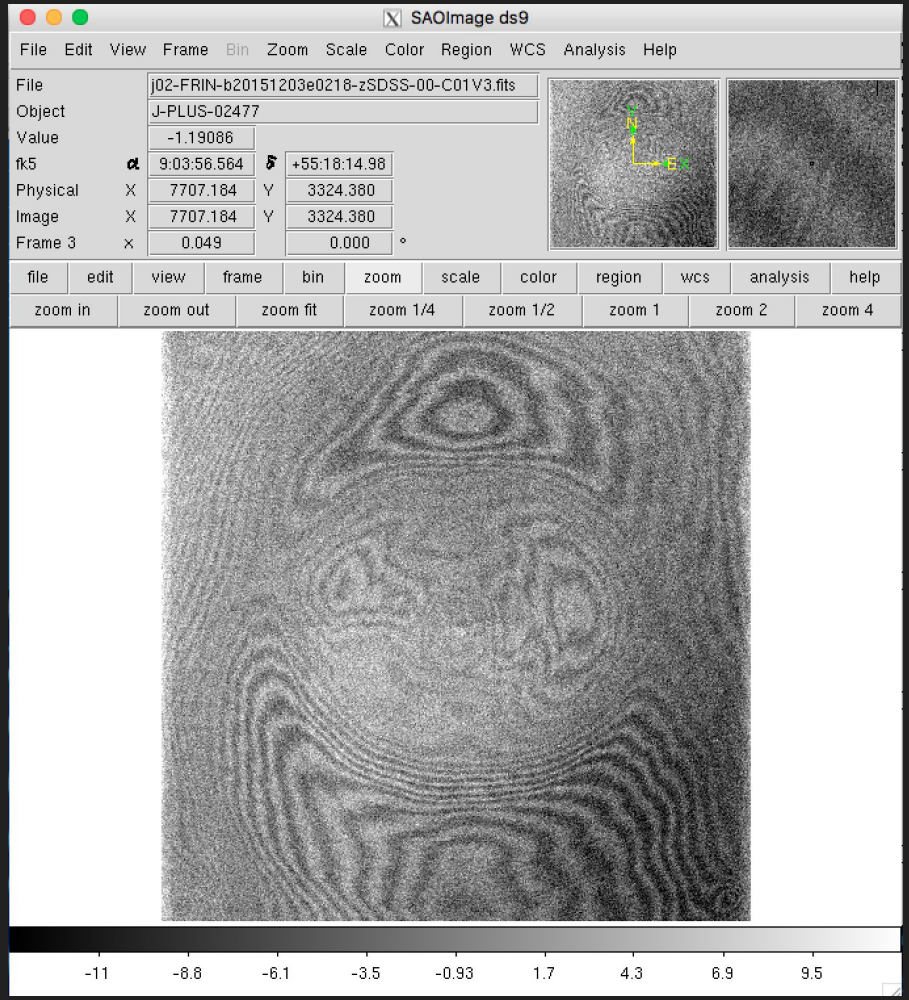
Fringing

This is due to the interaction of light with the coating of the CCD.

It only affects the reddest filters (and not always).

It may vary across the night.

It is (yet) another additive effect.



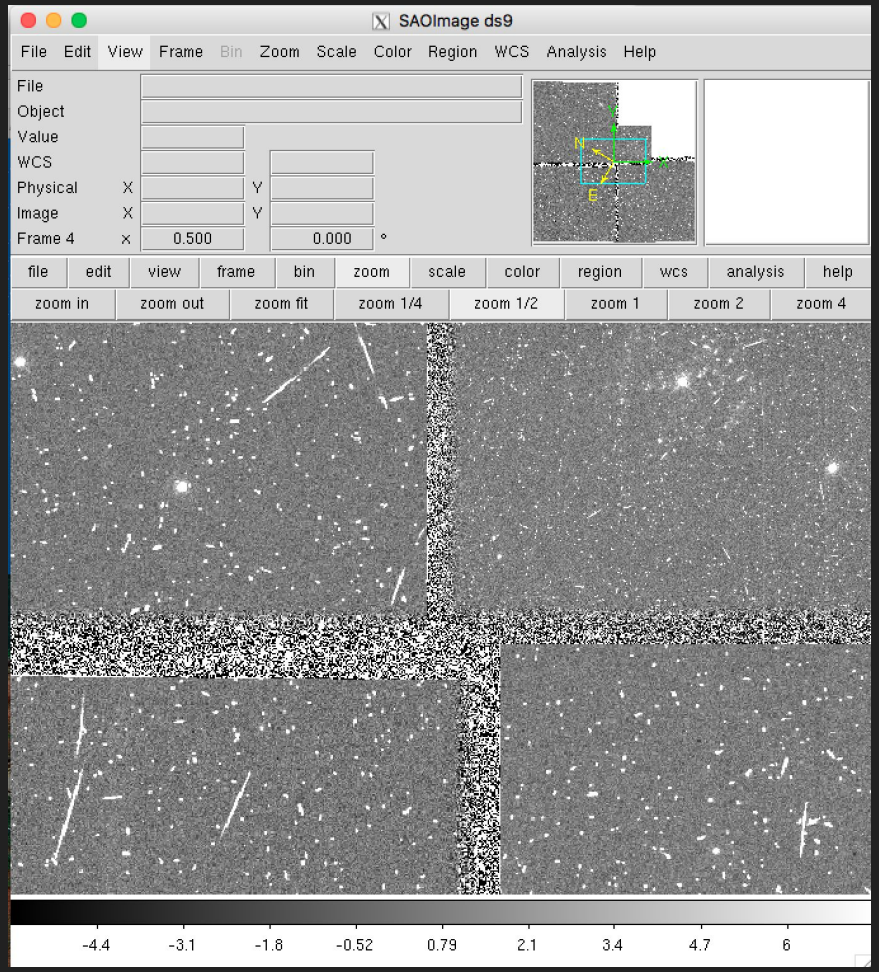
Cosmic Rays

These are high energy particles which interact with our silicon.

It gets worse out of the atmosphere (e.g. HST).

They are normally removed by “averaging” several images.

There are some programs (e.g. LACos) which use statistics to get rid of them.



Bad pixels / columns

Some pixels do not respond linearly -> hot/cold pixels.

Some pixels actually block the movement of charge -> bad columns

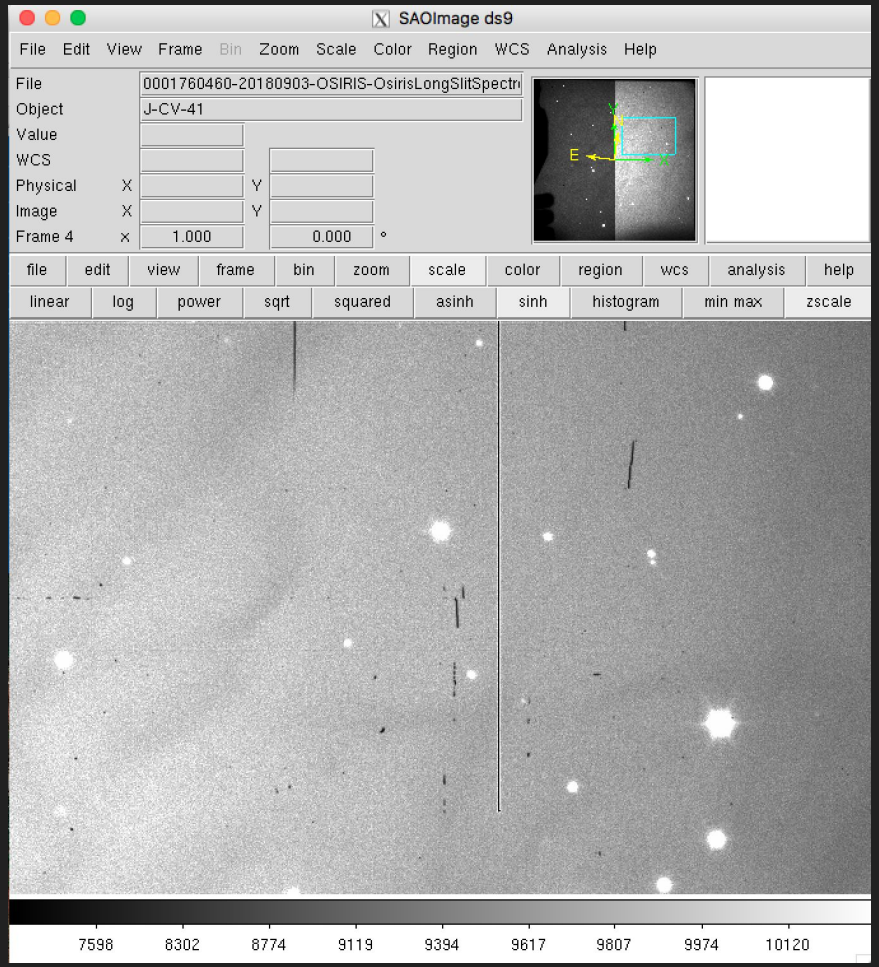
Fix it by

-) “dithering” the telescope

-) align the images

-) combine

It is also what the IRAF fixpix in ccdproc promises.



That’s all for today

Can you do the same for the other filters?

OR

Can you do it in Python?

Next time: we will see how to measure the characteristics of a CCD (gain and readout noise) and we will learn to do some basic photometry (finally speaking about noise)