LONG-SLIT REDUCTION WITH PYRAF

Rubén García-Benito

IAA-CSIC

Optical Spectroscopy School
University of Rwanda
August 24th - September 2nd, 2018

LONG-SLIT REDUCTION WITH PYRAF

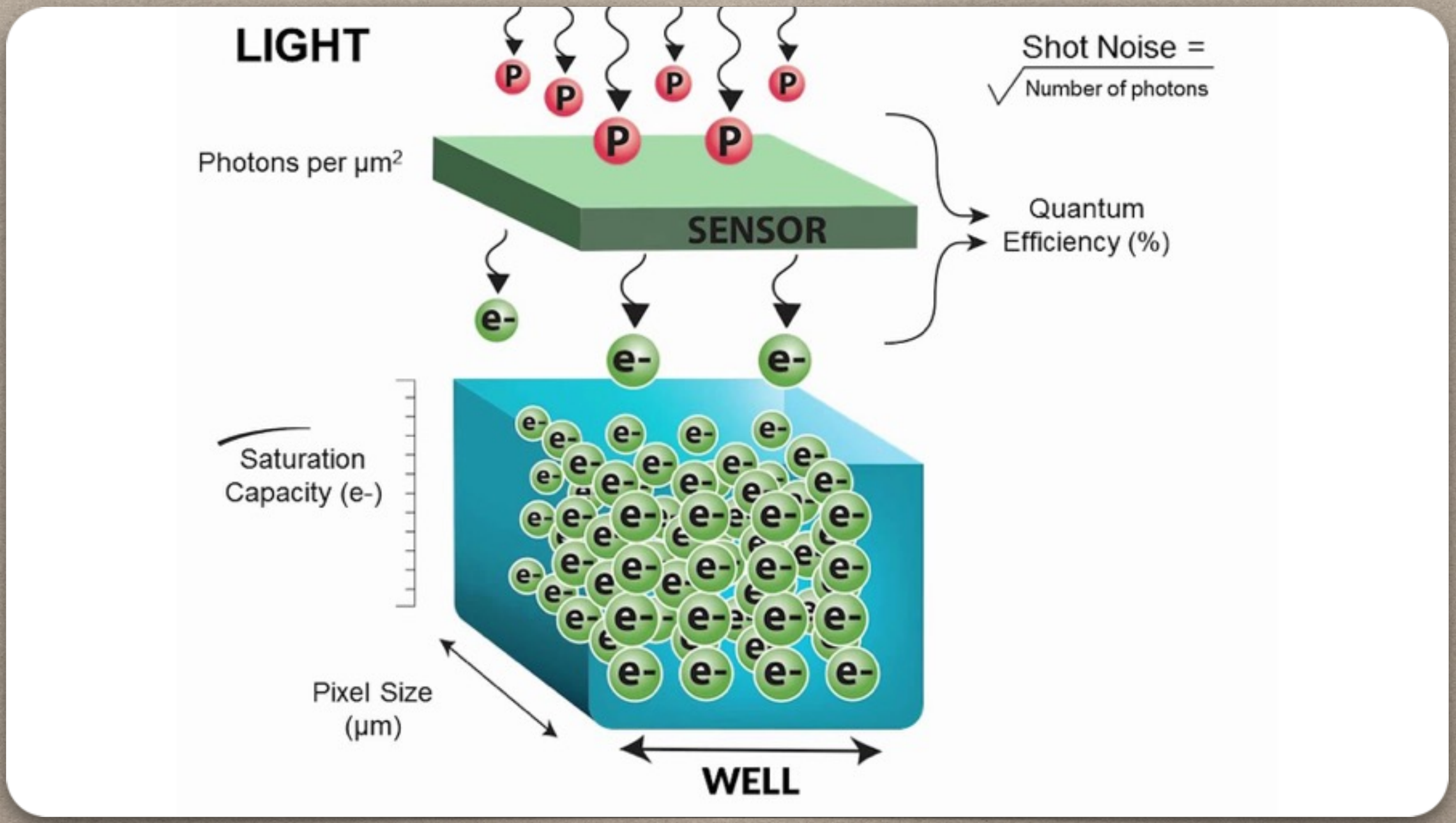
Rubén García-Benito

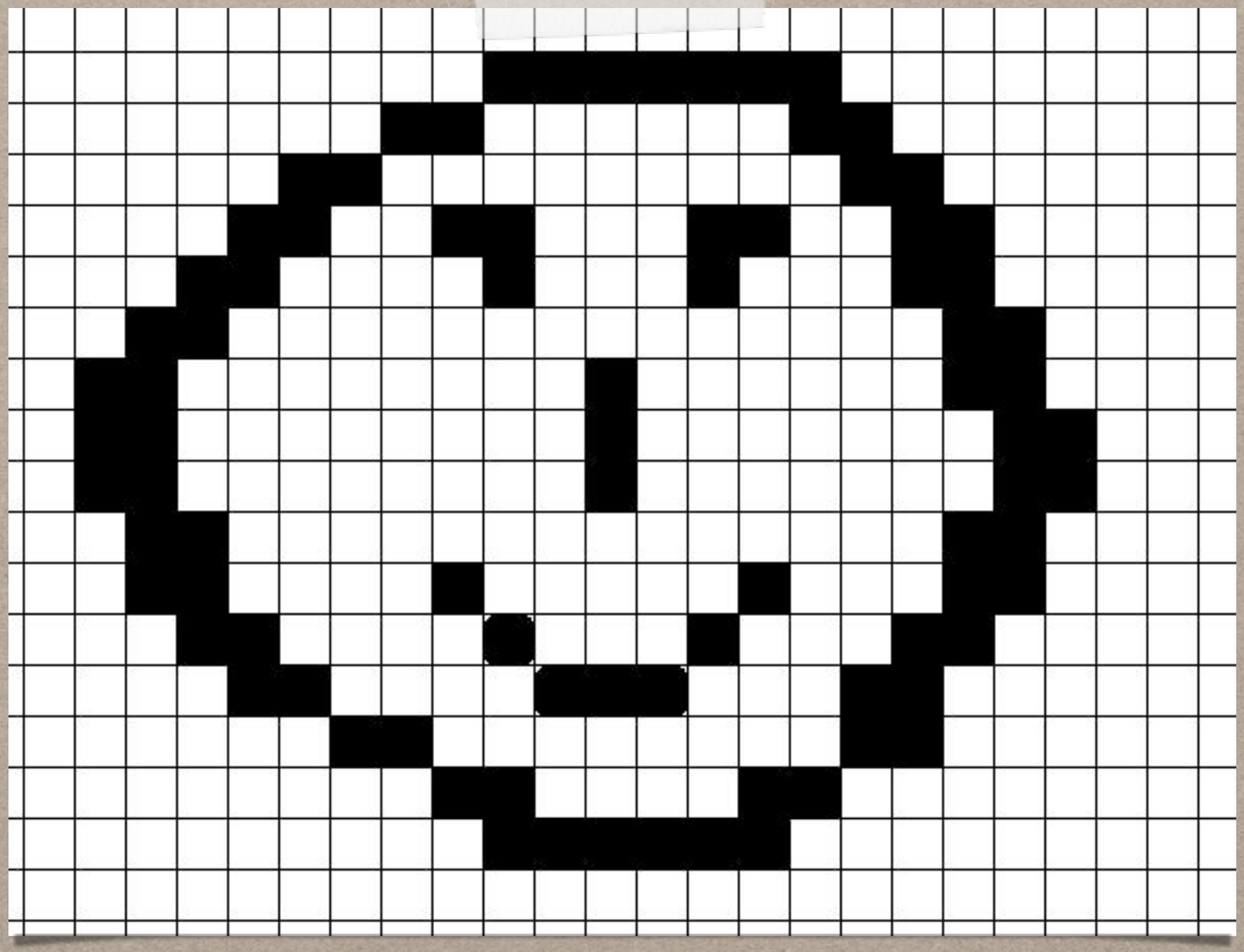
IAA-CSIC

Part I

Optical Spectroscopy School
University of Rwanda
August 24th - September 2nd, 2018

https://lco.global/spacebook/ccds/


LIGHT
Shot Noise =
√ Number of photons
Photons per μm²
SENSOR
Quantum
Efficiency (%)
Saturation
Capacity (e-)
Pixel Size
(μm)
WELL



0 - 65535

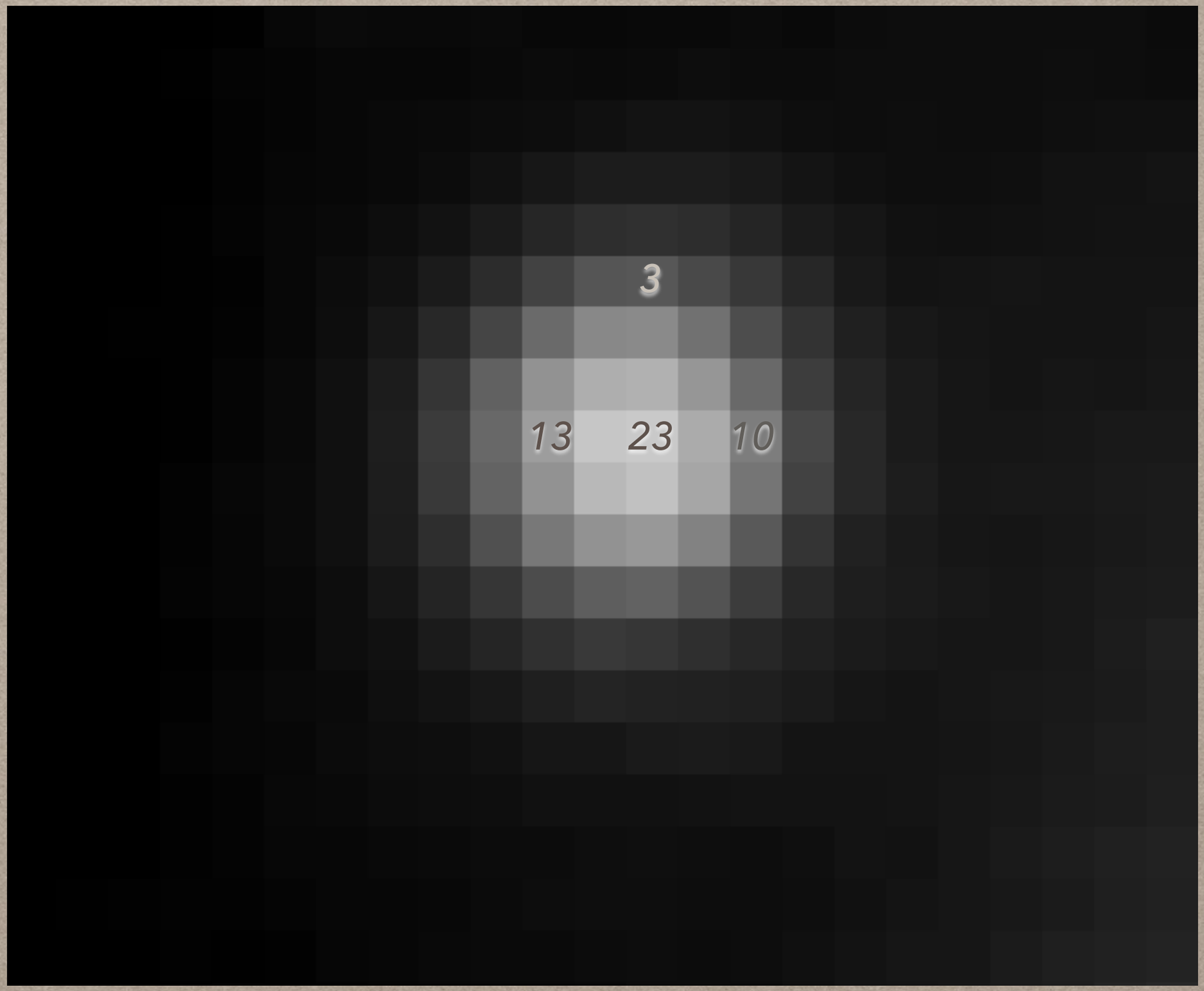
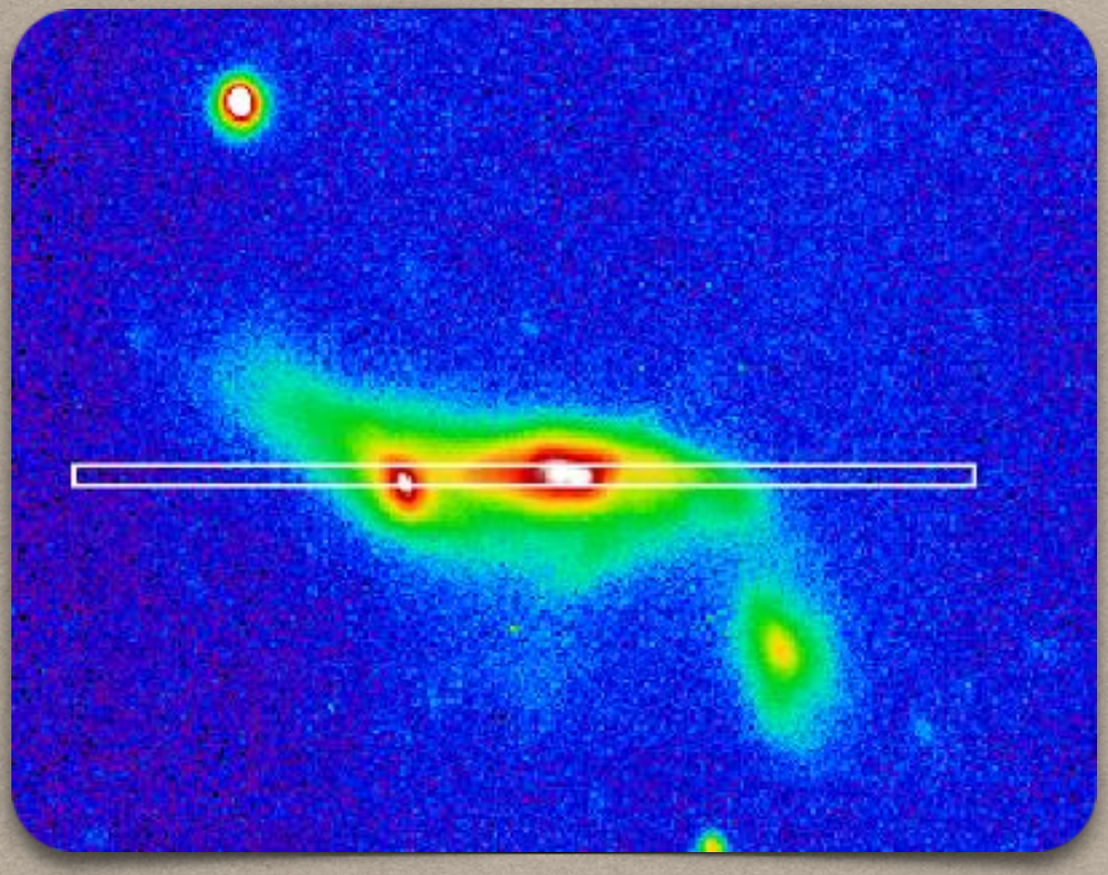

3
13
23
10

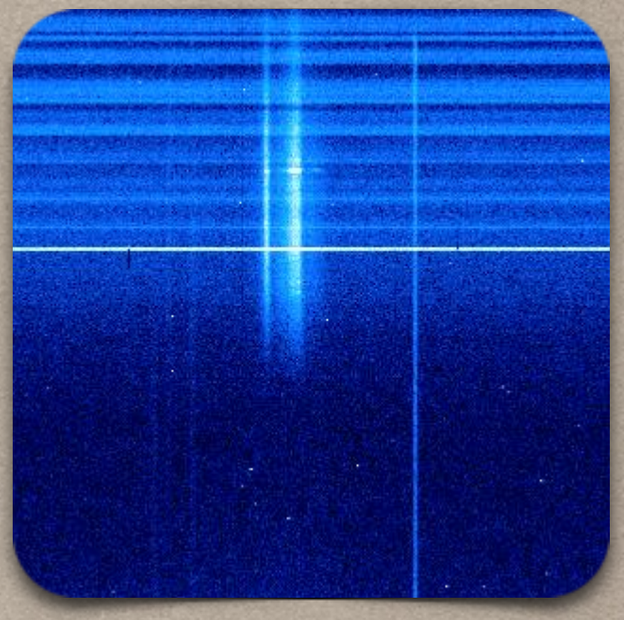
Tabla en píxeles

Fichero Editar Tamaño

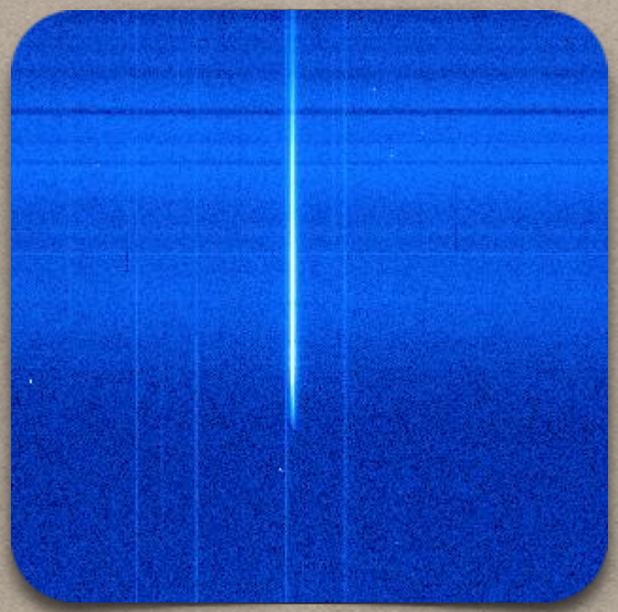
	451	452	453	454	455	456	457	458
491	3048.18	3069.08	3090.91	3122.76	3148.64	3185.13	3236.77	3228.
490	3052.89	3078.19	3123.35	3197.63	3306.04	3382.06	3380.05	3384.
489	3073.99	3132.47	3213.94	3364.13	3534.91	3666.06	3704.68	3655.
488	3112.6	3204.28	3364.43	3641.37	3988.66	4299.62	4336.77	4105.
487	3138.99	3300.5	3590.68	4054.33	4650.11	5141.98	5164.35	4750.
486	3191.78	3391.75	3787.63	4495.47	5310.92	5752.37	5798.51	5363.
485	3190.11	3413.14	3877.74	4620.39	5474.94	6141.12	6192	5705.
484	3192.07	3401.19	3845.14	4550.31	5304.58	5915.83	6067.65	5632.
483	3193.37	3379.96	3666.7	4226.84	4874.5	5304.63	5395.73	5043.
482	3144.96	3291.22	3510.33	3812.79	4158.65	4460.11	4513.44	4276.
481	3127.64	3202.4	3356.52	3525.7	3715.67	3854.68	3793.39	3702.
480	3093.12	3169.77	3233.53	3317.98	3435.2	3507.98	3483.71	3452.
479	3105.55	3135.91	3159.96	3207.85	3289.57	3290.83	3346.75	3370.

Cerrar

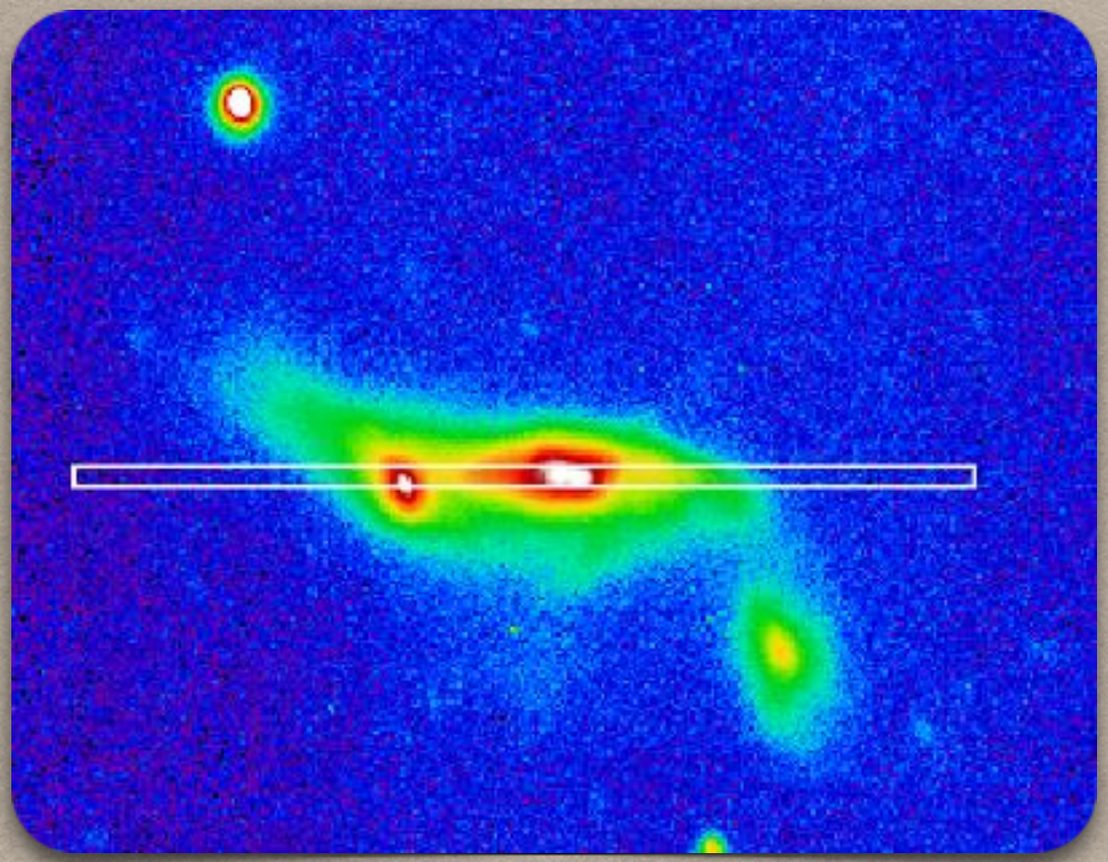




Galaxy

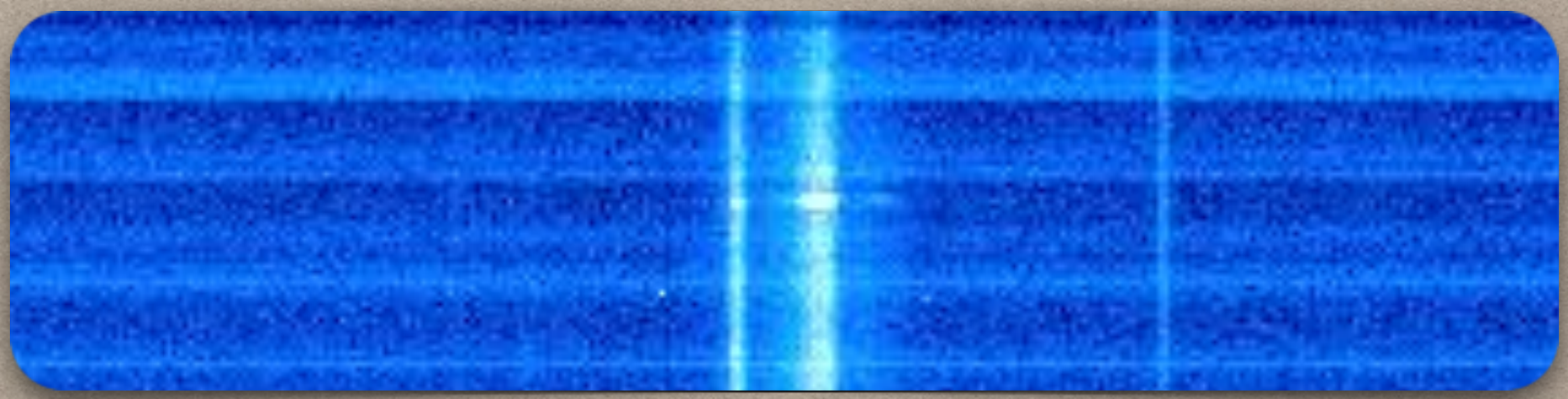


Star



Galaxy

spatial direction



λ

Reduction steps

1) *Pre-processing (correct frames):*
   * *Master Bias*
   * *Master Flat*

Frame - <bias>

<flat> - <bias>

2) *Wavelength calibration:*
   * identify
   * reidentify
   * fitcoords

3) *Flux calibration & sky subtraction:*
   * standard
   * sensfunc
   * calibrate

Reduction steps

1) **Pre-processing (correct frames):**
   * **Master Bias**
   * **Master Flat**

Frame - <bias>

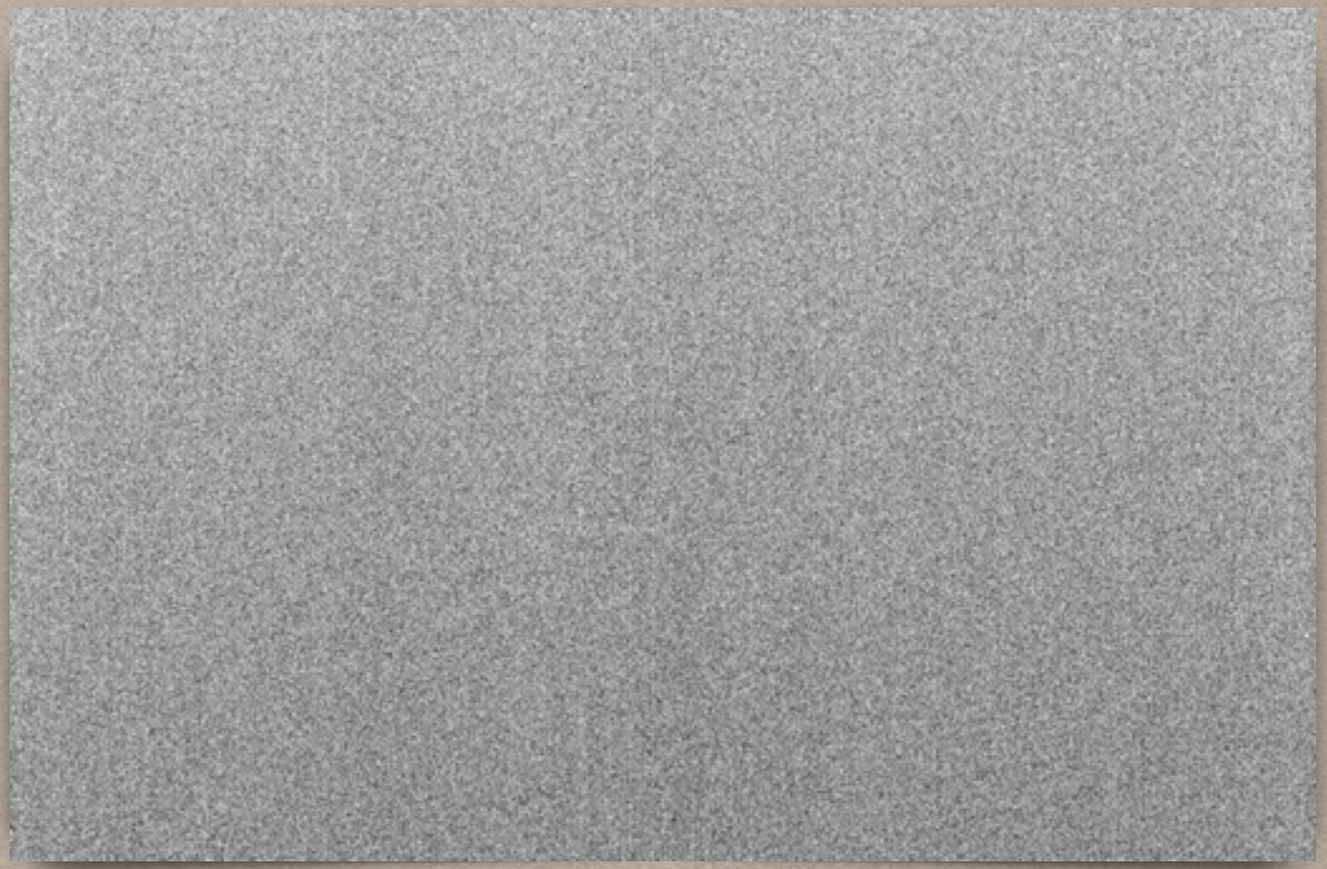
<flat> - <bias>

2) *Wavelength calibration:*
   * identify
   * reidentify
   * fitcoords

3) *Flux calibration & sky subtraction:*
   * standard
   * sensfunc
   * calibrate

BIAS

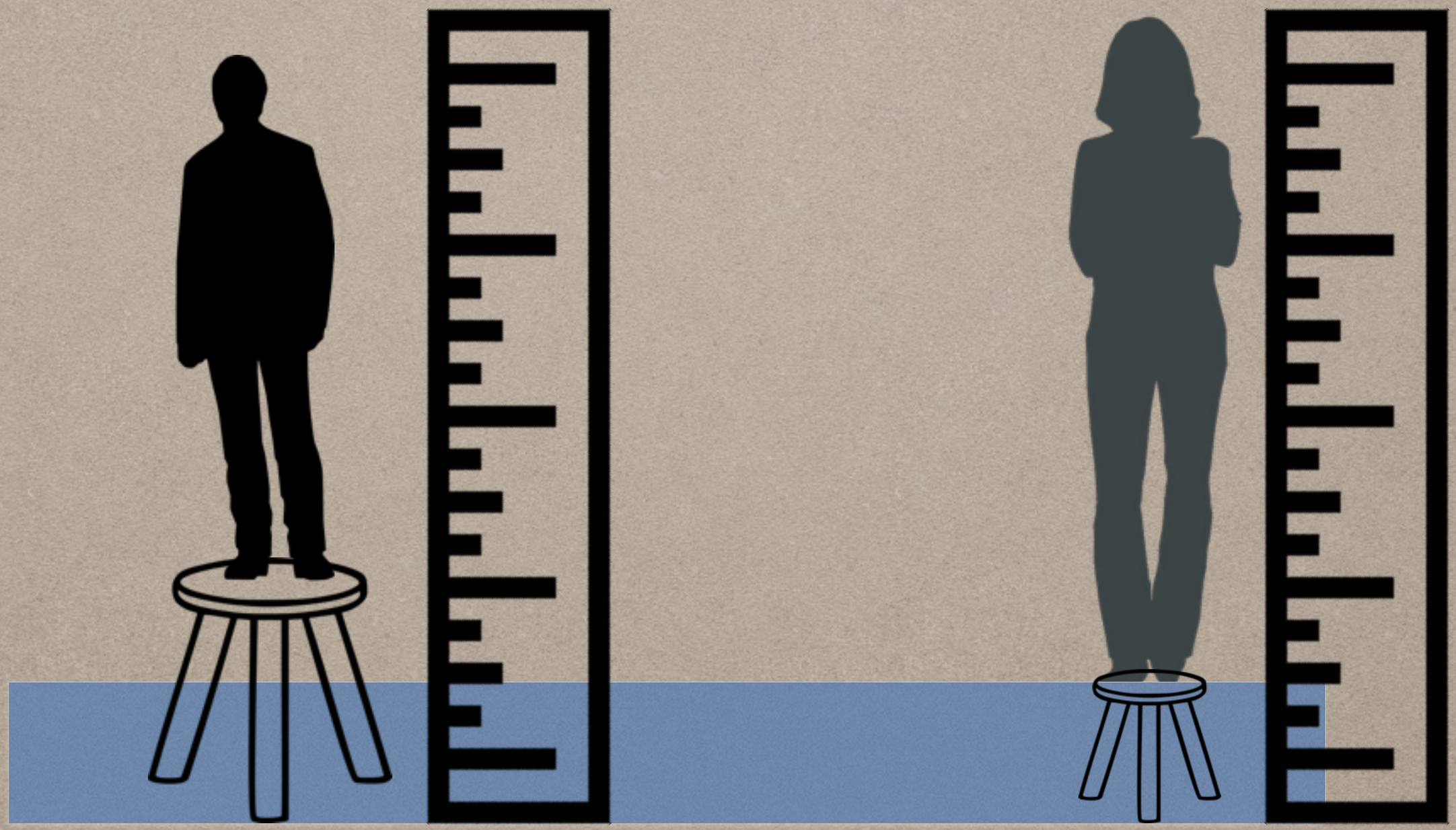
0 second images



Pixel-to-pixel structure in the the read noise

The bias is caused by a constant voltage applied to the CCD detector

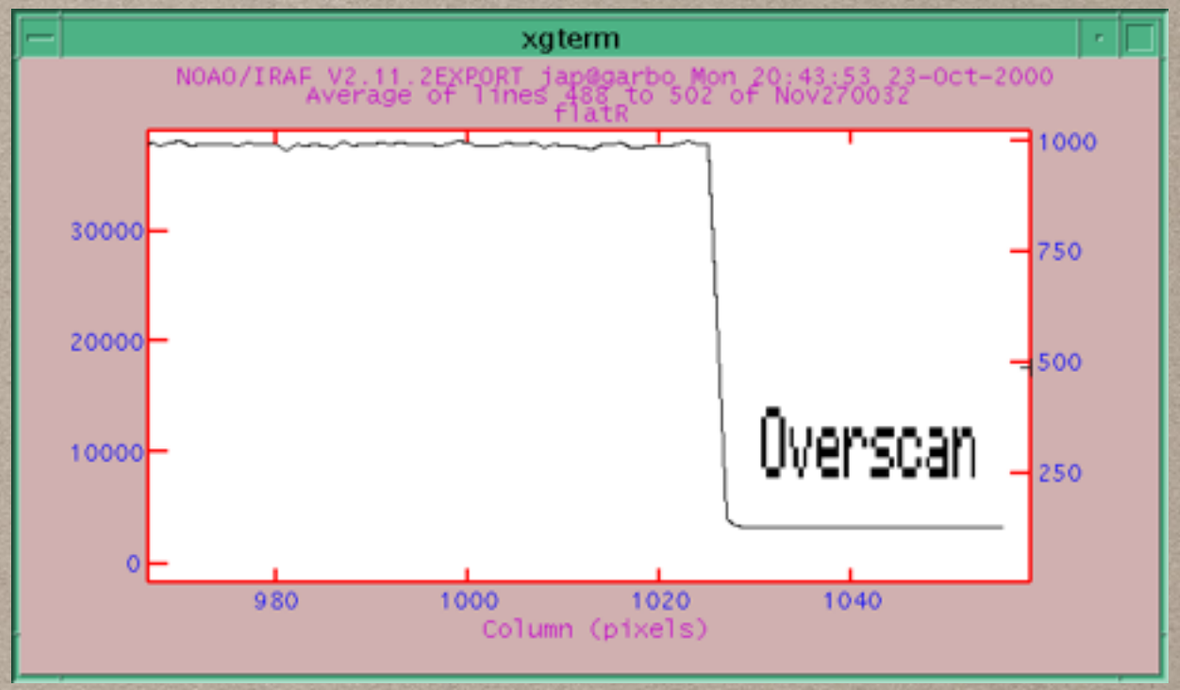
Ideally it should appear as a constant value added to the image



OVERSCAN

BIASSEC: Overscan

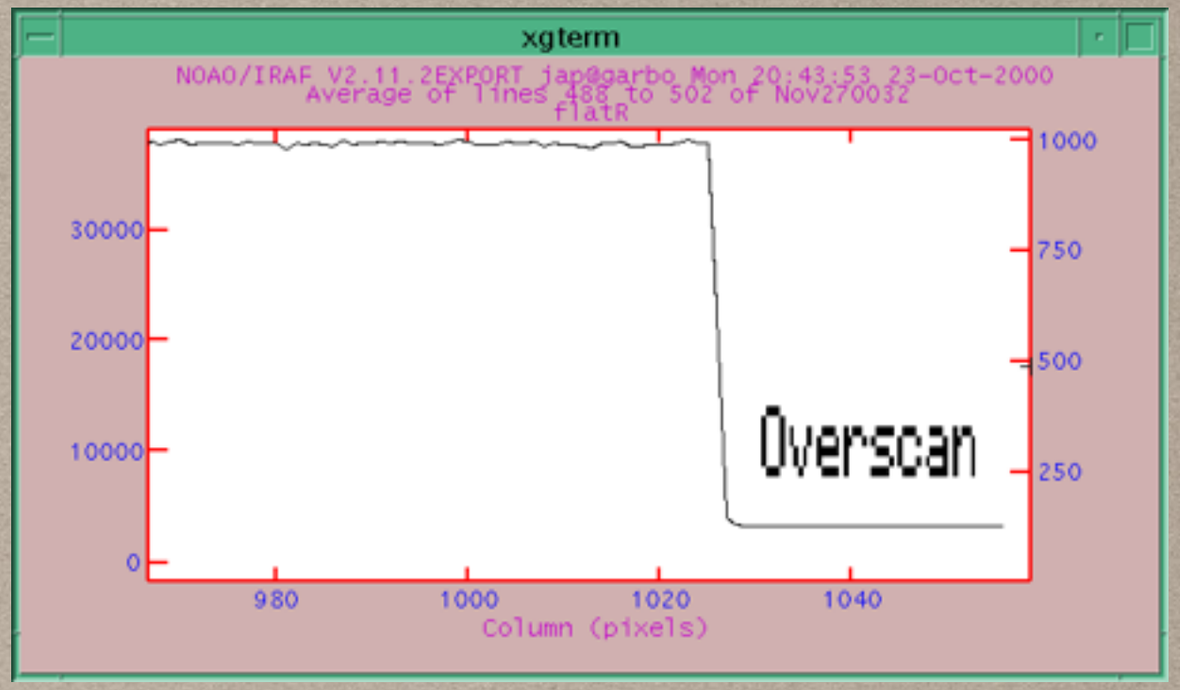
TRIMSEC: Data



Use a flat file to check

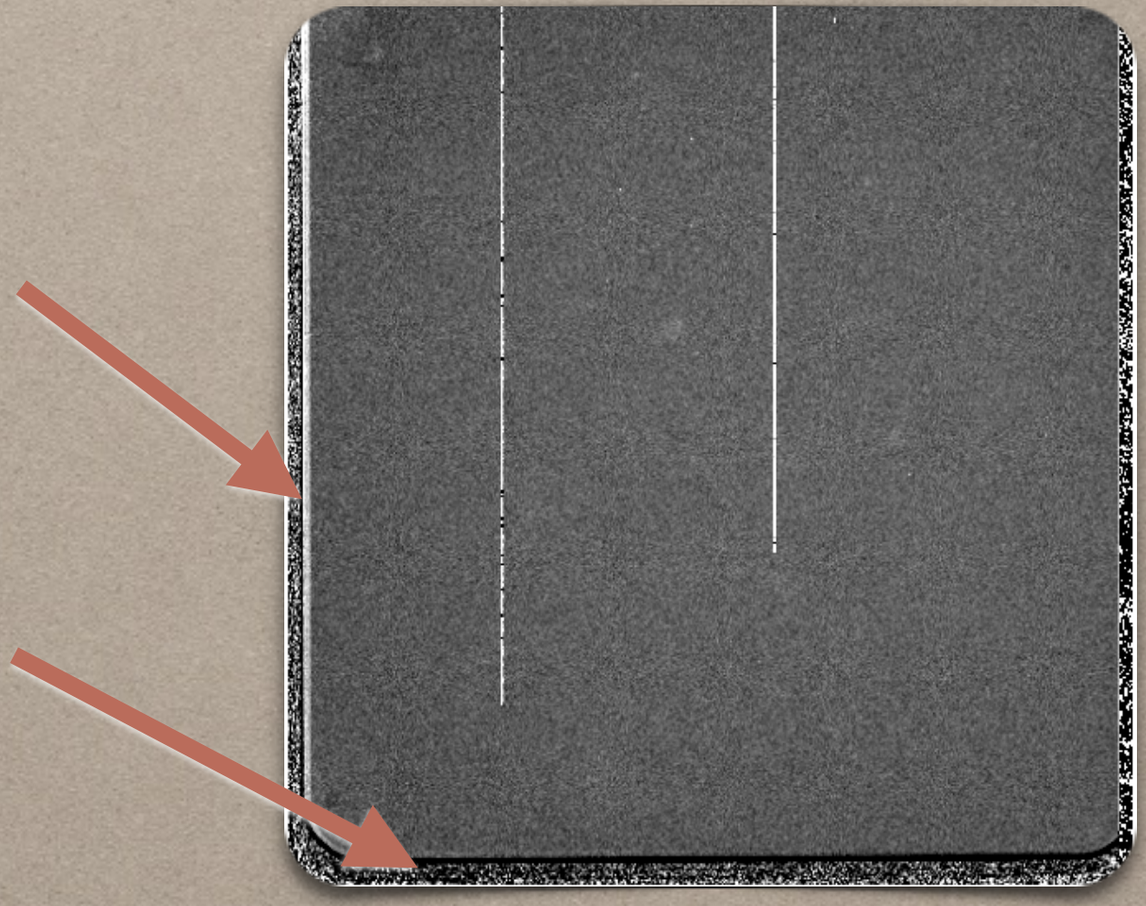
OVERSCAN

Rows/ columns not exposed to the light



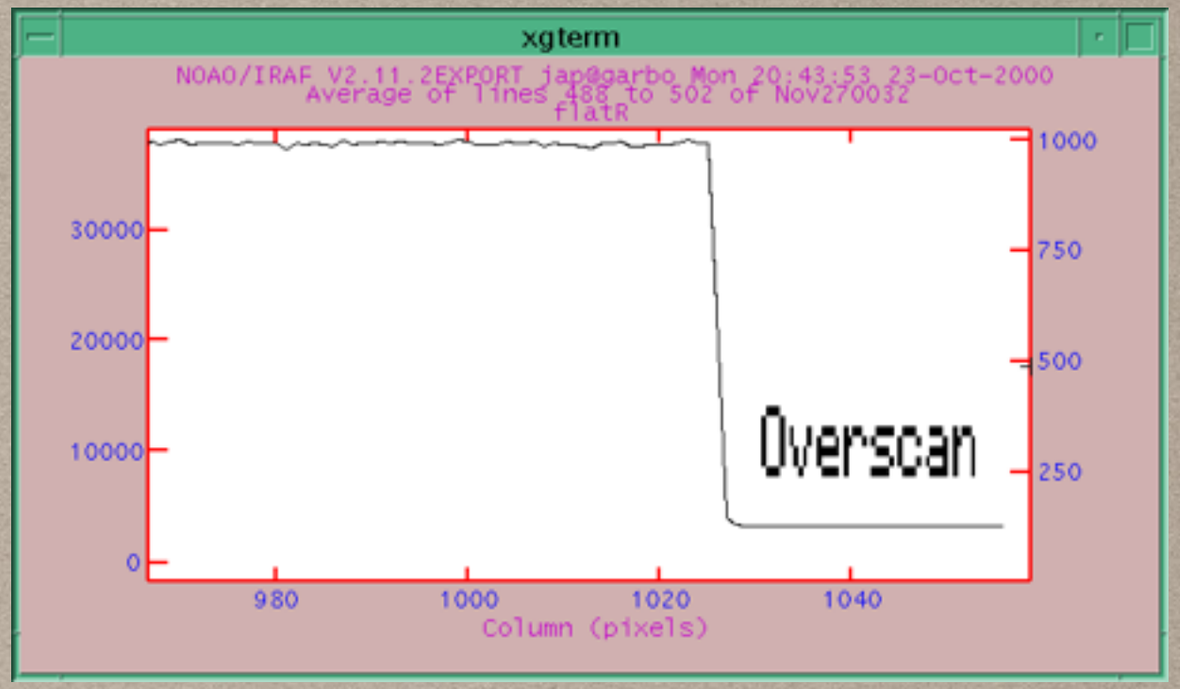
BIASSEC: Overscan

TRIMSEC: Data



OVERSCAN

Rows/ columns not exposed to the light



BIASSEC: Overscan

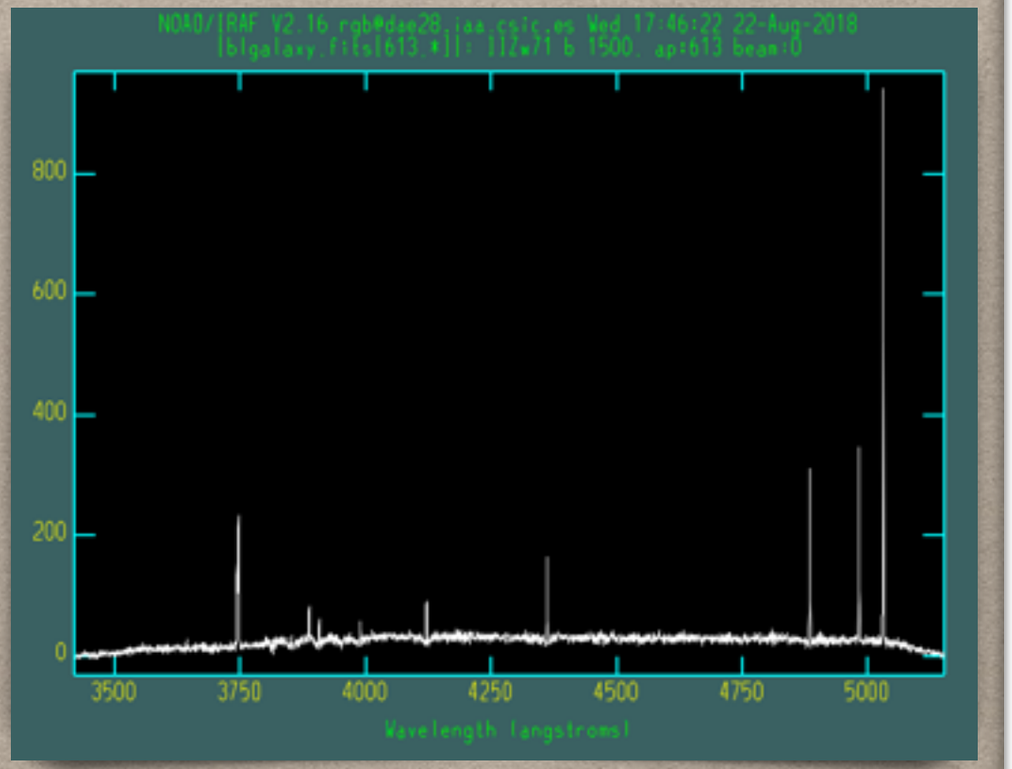
TRIMSEC: Data

IRAF AXIS NOTATION

IRAF notation: `image[x1:x2, y1:y2]`

Starts in 1

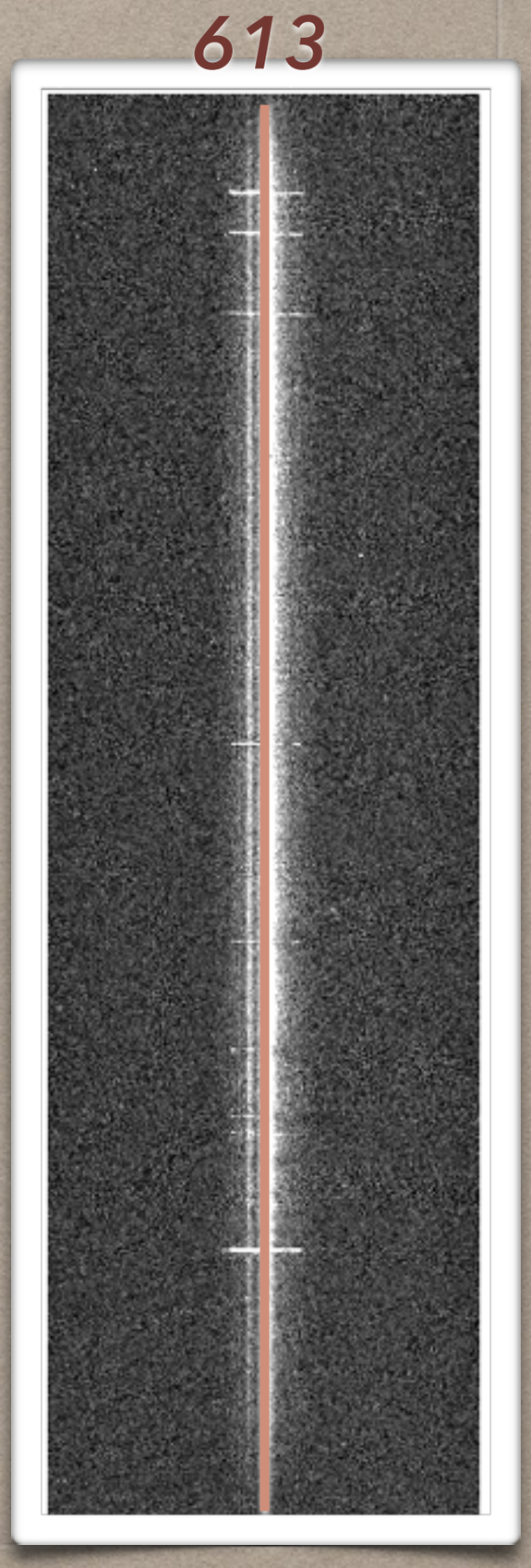
`splot file.fits[613,*]`



Python notation: `image[y1:y2, x1:x2]`

Starts in 0

`plot flux[:, 612]`



SETTING IRAF/PYRAF

```
~$ source activate iraf27
(iraf27) ~$ mkdir iraf
(iraf27) ~$ cd iraf
(iraf27) iraf$ mkiraf

-- creating a new uparm directory
Terminal types: xgterm,xterm,gterm,vt640,vt100,etc.
Enter terminal type [default xterm-256color]: xterm
                                              xgterm
```

CCD AND OBSERVATION SETTINGS

2007/07/08 - ISIS@WHT
Dispersion [A/pix] = 0.45 (B)
Spatial Resolution = 0.2 (arcsec/pixel)
Slit width = 1.''
GAIN_B = 1.16 [e-/ADU]
RDNOISE_B = 5.0 [e-]

file	run	object	RA	Dec	equinox	UT	airm	instr	texp	PA	slit	grat	cenwA
r761744.fit	761744	bias	14:01:38.81	01:32:40.1	J2000.0	20:47	1.14	ISISB	0.0	180	1.03	R600B	4302
r761746.fit	761746	bias	14:01:38.74	01:32:39.3	J2000.0	20:48	1.14	ISISB	0.0	180	1.03	R600B	4302
r761748.fit	761748	bias	0:00:00.00	00:00:00.0	J2000.0	20:48	1.05	ISISB	0.0	180	1.03	R600B	4302
r761750.fit	761750	bias	13:47:15.55	17:27:24.7	J2000.0	20:49	1.05	ISISB	0.0	180	1.03	R600B	4302
r761752.fit	761752	bias	13:47:15.62	17:27:24.7	J2000.0	20:50	1.05	ISISB	0.0	180	1.03	R600B	4302
r761787.fit	761787	bd33	15:51:58.99	32:56:51.2	J2000.0	22:11	1.0	ISISB	240.0	76	1.02	R600B	4302
r761790.fit	761790	bd33	15:51:58.99	32:56:51.2	J2000.0	22:16	1.01	ISISB	240.0	76	1.02	R600B	4302
r761805.fit	761805	arc	14:51:14.73	35:32:31.0	J2000.0	23:09	1.14	ISISB	60.0	30	1.03	R600B	4302
r761806.fit	761806	flat	14:51:14.73	35:32:31.0	J2000.0	23:12	1.15	ISISB	30.0	30	1.03	R600B	4302
r761809.fit	761809	GALAXY	14:51:14.73	35:32:31.0	J2000.0	23:15	1.16	ISISB	1500.0	30	1.03	R600B	4302
r761810.fit	761810	GALAXY	14:51:14.94	35:32:29.5	J2000.0	23:41	1.23	ISISB	1500.0	30	1.03	R600B	4302
r761813.fit	761813	arc	14:51:14.94	35:32:29.4	J2000.0	00:08	1.33	ISISB	60.0	30	1.03	R600B	4302
r761815.fit	761815	flat	14:51:14.94	35:32:29.4	J2000.0	00:11	1.34	ISISB	30.0	30	1.03	R600B	4302
r761821.fit	761821	arc	21:59:41.89	26:25:56.7	J2000.0	01:11	1.27	ISISB	60.0	283	1.03	R600B	4302
r761822.fit	761822	flat	21:59:41.89	26:25:56.8	J2000.0	01:13	1.26	ISISB	30.0	283	1.03	R600B	4302
r761824.fit	761824	bd25	21:59:41.89	26:25:56.8	J2000.0	01:16	1.25	ISISB	100.0	283	1.03	R600B	4302
r761827.fit	761827	bd25	21:59:41.89	26:25:56.7	J2000.0	01:20	1.24	ISISB	100.0	283	1.03	R600B	4302
r761881.fit	761881	bias	19:16:42.76	63:58:12.6	J2000.0	05:31	1.64	ISISB	0.0	76	1.03	R600B	4302
r761882.fit	761882	bias	19:16:42.76	63:58:12.5	J2000.0	05:31	1.65	ISISB	0.0	76	1.03	R600B	4302
r761883.fit	761883	bias	0:00:00.00	00:00:00.0	J2000.0	05:32	1.22	ISISB	0.0	76	1.03	R600B	4302
r761884.fit	761884	bias	0:00:00.00	00:00:00.0	J2000.0	05:32	1.0	ISISB	0.0	76	1.03	R600B	4302
r761885.fit	761885	bias	0:00:00.00	00:00:00.0	J2000.0	05:33	1.0	ISISB	0.0	76	1.03	R600B	4302

~

BIAS *image statistics*

1) Create file with a list of bias files

r761744.fit
r761746.fit
r761748.fit
r761750.fit
r761752.fit
r761881.fit
r761882.fit
r761883.fit
r761884.fit
r761885.fit

BIAS *image statistics*

2) Load imstatistics

```
from pyraf import iraf
iraf.noao.imred(_doprint=0)
iraf.noao.imred.ccdred(_doprint=0)

from iraf import imstatistics
```

 # IRAF packages

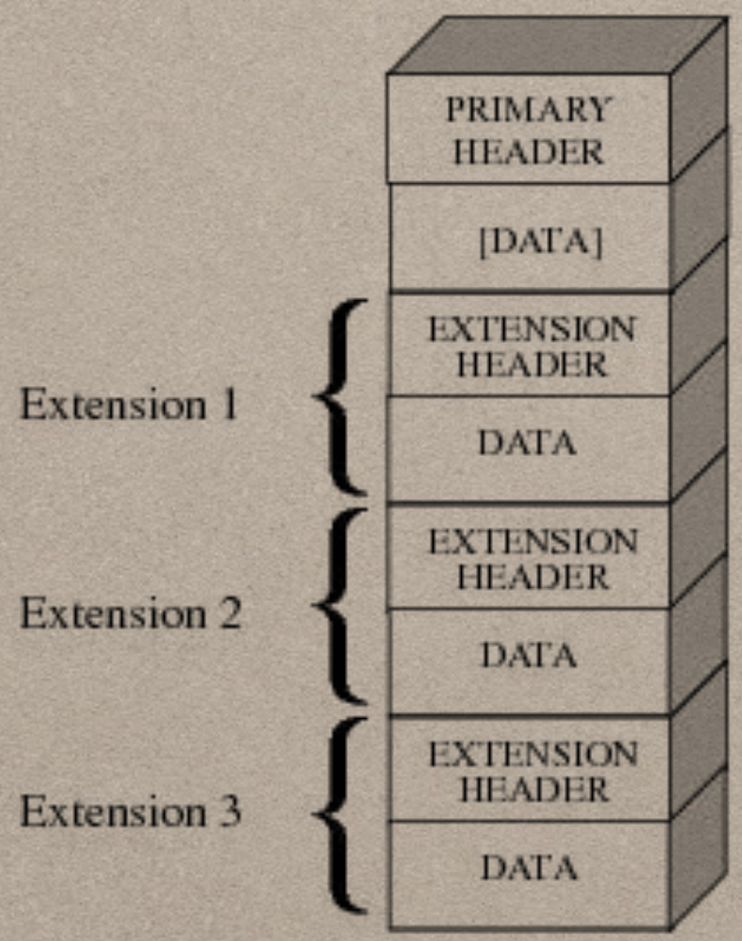
 # IRAF functions

BIAS *image statistics*

3) Command line

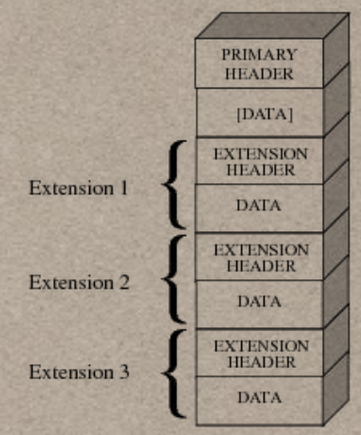
```
imstatistics images=@bias.list fields='image,mean,midpt,stddev,min,max'
```

FITS file format



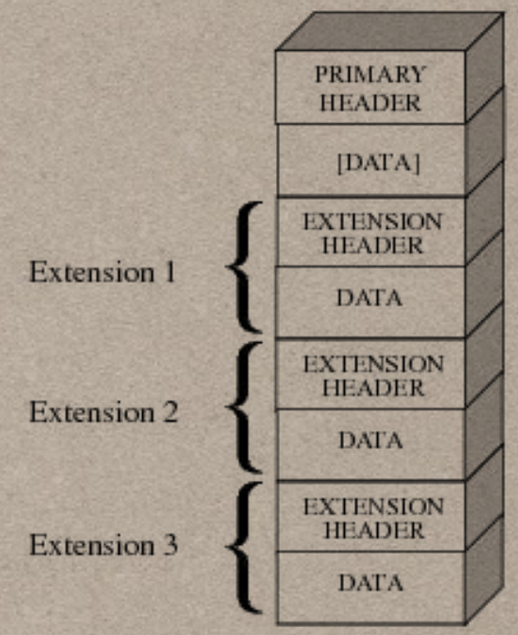

```
import astropy.io.fits as fits

f = fits.open('spec-0861-52318-0261.fits')
f.info()
f.close()
```

FITS file format




```
Filename: spec-0861-52318-0261.fits
No.   Name       Type            Cards  Dimensions  Format
0   PRIMARY    PrimaryHDU    139    ()
1   COADD      BinTableHDU   26     3821R x 8C  [E, E, E, J, J, E, E, E]
2   SPECOBJ    BinTableHDU   262    1R x 126C   [6A, 4A, 16A, 23A, 16A, 8A, E, E, E, J, E, E, J, B, B,
B, B, B, B, J, 22A, 19A, 19A, 22A, 19A, I, 3A, 3A, 1A, J, D, D, D, E, E, 19A, 8A, J, J, J, J, K, K, J, J, J, J,
J, J, K, K, K, K, I, J, J, J, J, 5J, D, D, 6A, 21A, E, E, E, J, E, 24A, 10J, J, 10E, E, E, E, E, E, E, J, E, E, E, J,
E, 5E, E, 10E, 10E, 10E, 5E, 5E, 5E, 5E, 5E, J, J, E, E, E, E, E, E, 25A, 21A, 10A, E, E, E, E, E, E, E, E, J, E,
E, J, 1A, 1A, E, E, J, J, 1A, 5E, 5E]
3   SPZLINE    BinTableHDU   48     29R x 19C   [J, J, J, 13A, D, E, E, E, E, E, E, E, E, E, E, J, J, E, E]
4   B1-00013168-00013172-00013173  BinTableHDU   146   2044R x 7C   [E, E, E, J, E, E, E]
5   B1-00013169-00013172-00013173  BinTableHDU   146   2044R x 7C   [E, E, E, J, E, E, E]
6   B1-00013170-00013172-00013173  BinTableHDU   146   2044R x 7C   [E, E, E, J, E, E, E]
7   R1-00013168-00013172-00013173  BinTableHDU   146   2047R x 7C   [E, E, E, J, E, E, E]
8   R1-00013169-00013172-00013173  BinTableHDU   146   2047R x 7C   [E, E, E, J, E, E, E]
9   R1-00013170-00013172-00013173  BinTableHDU   146   2045R x 7C   [E, E, E, J, E, E, E]
```

FITS file format




```
Filename: r761744.fit

No.   Name      Type      Cards  Dimensions   Format
0   PRIMARY    PrimaryHDU    129  ()
1                ImageHDU       50  (1201, 4200)   int16 (rescales to uint16)
```

BIAS

Statistics

```
from iraf import imstatistics

imstatistics images=@bias.list fields='image,mean,midpt,stddev,min,max'
```

r761744.fit[1]
r761746.fit[1]
r761748.fit[1]
r761750.fit[1]
r761752.fit[1]
r761881.fit[1]
r761882.fit[1]
r761883.fit[1]
r761884.fit[1]
r761885.fit[1]

} *bias.list*

BIAS

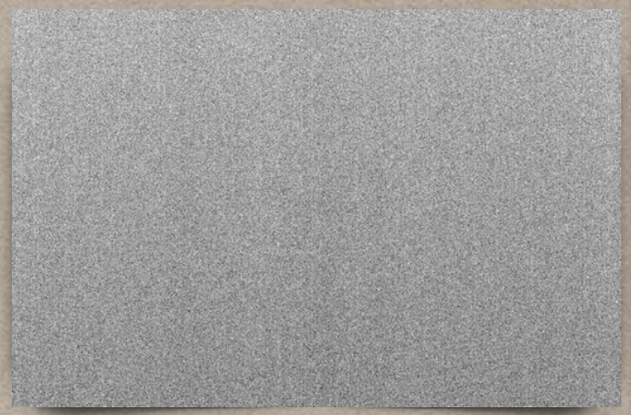
Statistics

```
from iraf import imstatistics

imstatistics images=@bias.list fields='image,mean,midpt,stddev,min,max'
```

# IMAGE	MEAN	MIDPT	STDDEV	MIN	MAX
r761744.fit[1]	1372.	1372.	4.828	1193.	2910.
r761746.fit[1]	1372.	1372.	5.412	1195.	5242.
r761748.fit[1]	1372.	1372.	4.756	1194.	2329.
r761750.fit[1]	1372.	1372.	4.855	1193.	3684.
r761752.fit[1]	1372.	1372.	4.755	1194.	2705.
r761881.fit[1]	1372.	1371.	4.566	1195.	2562.
r761882.fit[1]	1372.	1372.	4.83	1196.	3025.
r761883.fit[1]	1374.	1374.	5.184	1196.	4481.
r761884.fit[1]	1375.	1375.	4.359	1197.	2395.
r761885.fit[1]	1375.	1375.	4.291	1195.	2154.

BIAS *command line & epar*

zercombine

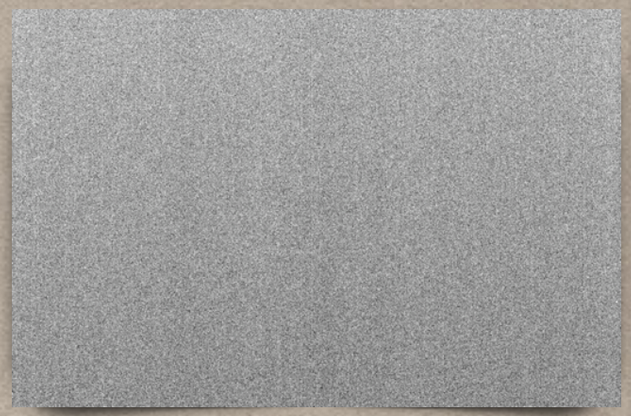


1) Create file with a list of bias files

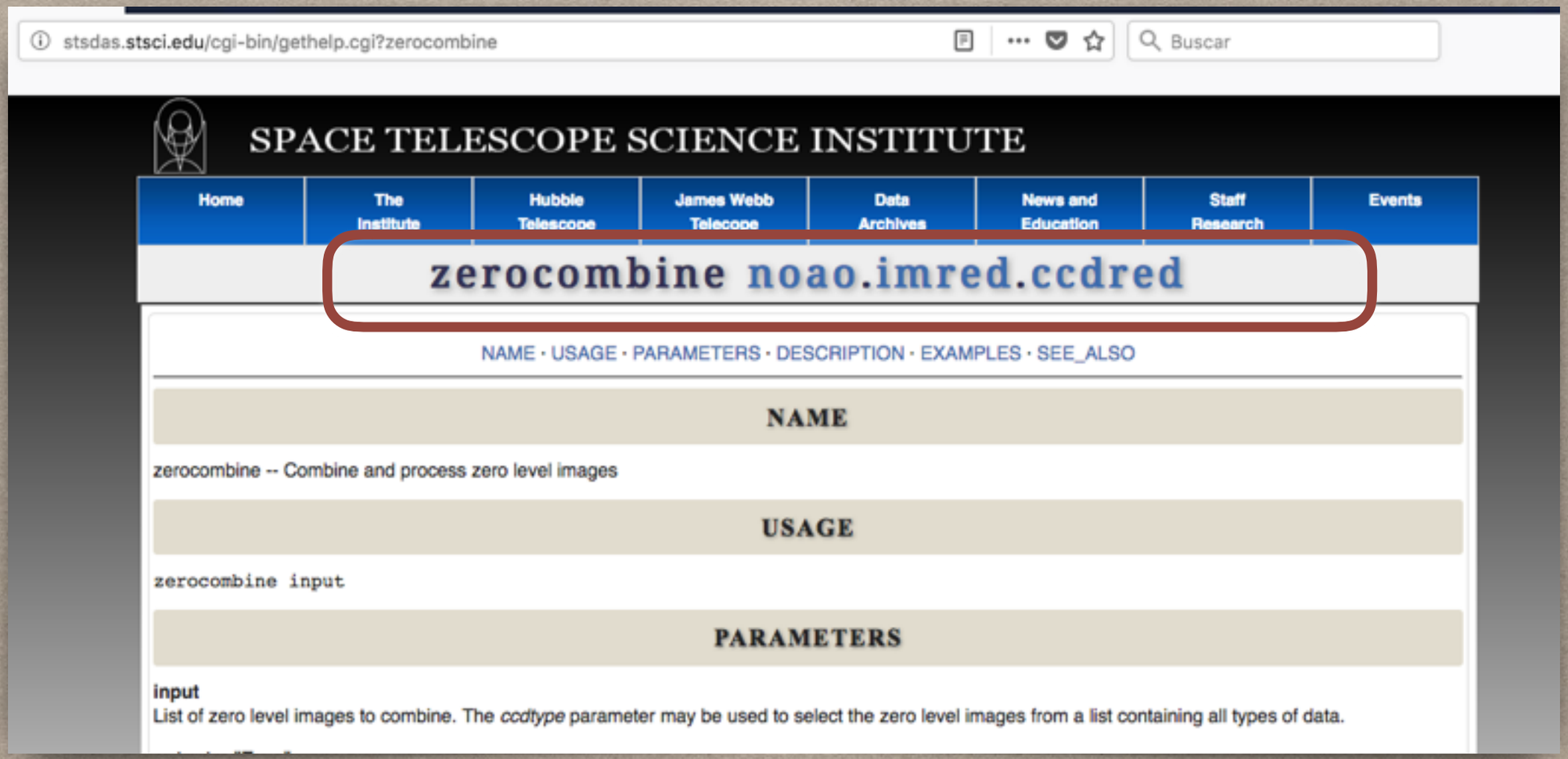
r761744.fit[1]
r761746.fit[1]
r761748.fit[1]
r761750.fit[1]
r761752.fit[1]
r761881.fit[1]
r761882.fit[1]
r761883.fit[1]
r761884.fit[1]
r761885.fit[1]

BIAS *command line & epar*

zercombine



2) Load zerocombine



stsdas.stsci.edu/cgi-bin/gethelp.cgi?zerocombine

Buscar

SPACE TELESCOPE SCIENCE INSTITUTE

Home | The Institute | Hubble Telescope | James Webb Telecope | Data Archives | News and Education | Staff Research | Events

zerocombine noao.imred.ccdred

NAME · USAGE · PARAMETERS · DESCRIPTION · EXAMPLES · SEE_ALSO

NAME

zerocombine -- Combine and process zero level images

USAGE

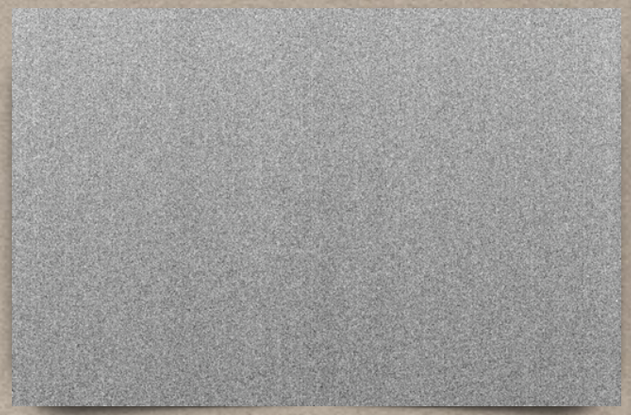
```
zerocombine input
```

PARAMETERS

input
List of zero level images to combine. The *ccdtype* parameter may be used to select the zero level images from a list containing all types of data.

BIAS *command line & epar*

zercombine



2) Load zerocombine

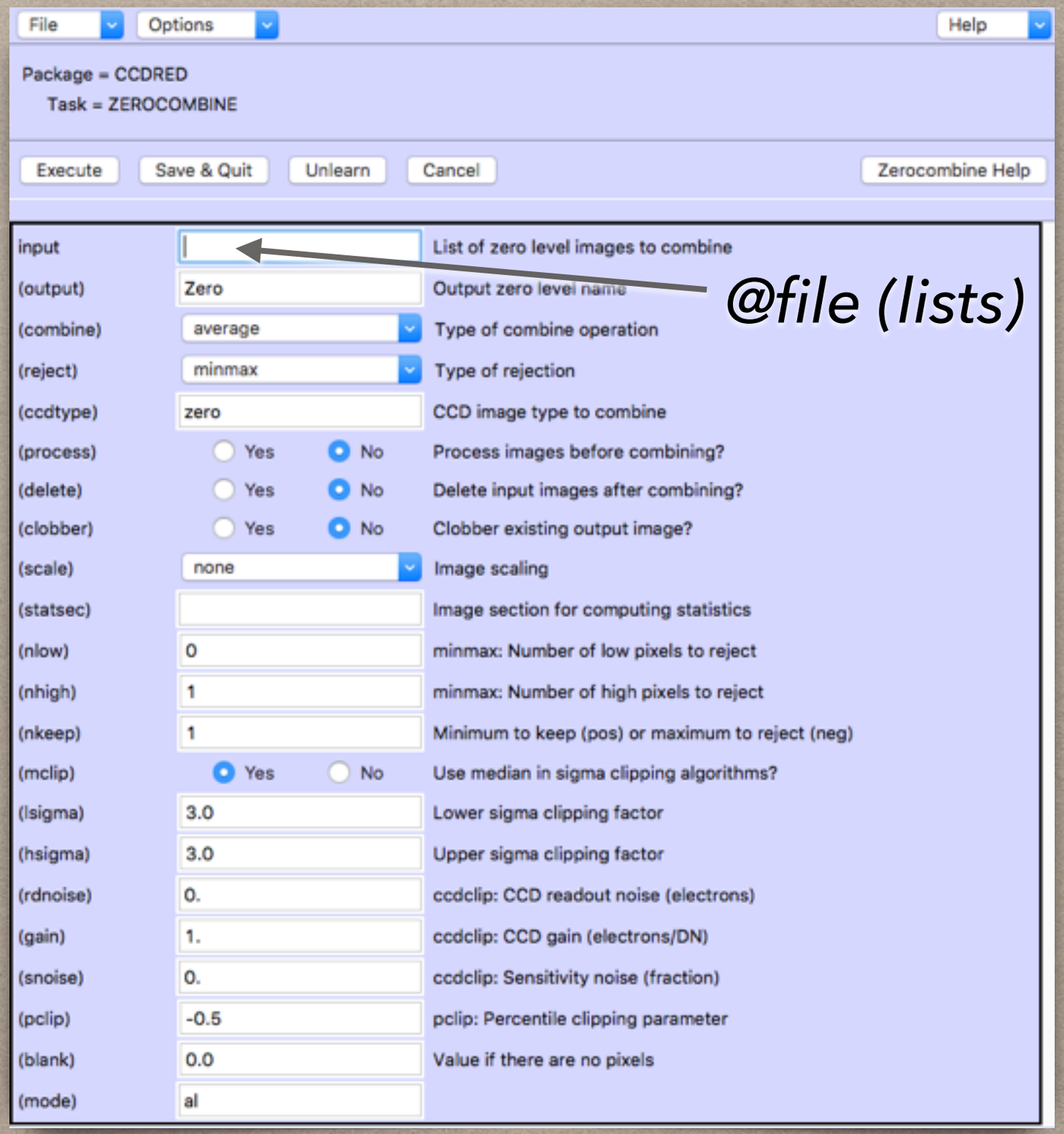
```
from pyraf import iraf
iraf.noao.imred(_doprint=0)
iraf.noao.imred.ccdred(_doprint=0)

from iraf import zerocombine
```

 # IRAF packages

 # IRAF functions

BIAS *command line & epar*

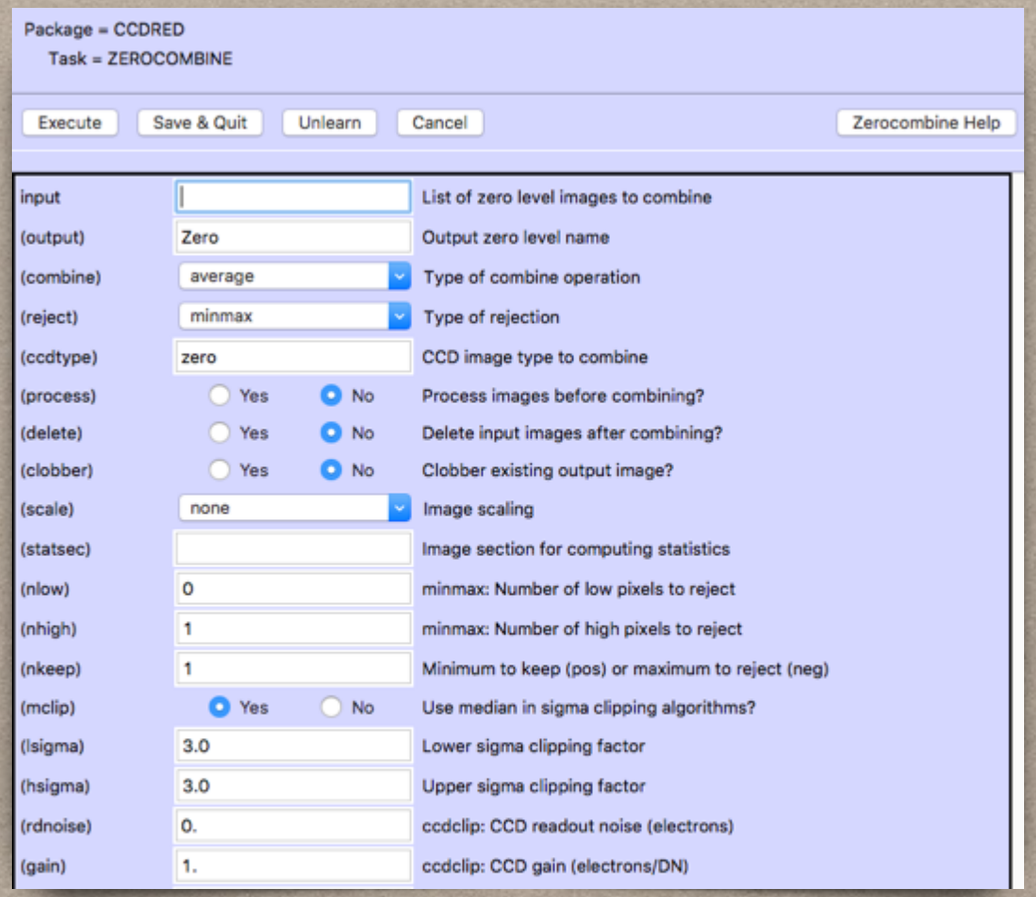
3) epar zerocombine



BIAS *command line & epar*

3) Command line*

zerocombine input=@bias.list output=biasf.fits
combine=average rdnoise=5. gain=1.16



Long sequences

Flat fielding:

1) *Create list of flat files*
2) *Create a master flat with* `flatcombine`
3) *Crop and remove bias and overscan from flat with* `ccdproc`
4) *Normalize master flat with* `response`

Scripting in PyRAF

```
# IRAF packages
from pyraf import iraf
iraf.noao.imred(_doprint=0)
iraf.noao.twodspec(_doprint=0)
iraf.noao.onedspec(_doprint=0)
iraf.noao.imred.ccdred(_doprint=0)
iraf.noao.twodspec.longslit(_doprint=0)
iraf.noao.twodspec.apextract(_doprint=0)

# IRAF functions
from iraf import imcombine, background, apall
from iraf import ccdproc, zerocombine, imstatistics
from iraf import flatcombine, response, hedit, unlearn
from iraf import identify, reidentify, fitcoords, transform
from iraf import standard, sensfunc, calibrate, setairmass

# Python modules
from collections import OrderedDict
from astropy.table import Table
from glob import glob
import numpy as np
import astropy
import os
```

Scripting in PyRAF

Options:

1) Move relspyraf.py *in the same directory as your reduction script (*long_slit_reduction_with_PyRAF.py*)*

2) Move relspyraf.py *to your $PYTHONPATH directory*

~anaconda/lib/python2.7/site-packages
~anaconda/envs/astroconda/lib/python2.7/site-packages
~$HOME/.local/lib/python2.7/site-packages

Scripting in PyRAF

Options:

3) Move relspyraf.py *to your personal python directory (*my_python_directory*) and add to your* $HOME/.basrhrc *(or* $HOME/.bash_profile*)*

```
export PYTHONPATH=my_python_directory:$PYTHONPATH
```

Scripting in PyRAF

```
from collections import OrderedDict
from astropy.table import Table
from relspyraf import *
```

Zerocombine (BIAS)

```
-->
--> lpar zerocombine
       input = "@bias.list"    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 = 3.0)            Lower sigma clipping factor
     (hsigma = 3.0)            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 = 0.0)            Value if there are no pixels
       (mode = "al")
```

*Default parameters (**unlearn**)*

```
ltask = [‘ccdproc','zerocombine','imstatistics','flatcombine','response',
        ‘standard','sensfunc','calibrate','setairmass',
        'identify','reidentify','fitcoords','transform',
        'imcombine','background','apall']


for task in ltask:
    unlearn(task)
```

*Default parameters (**unlearn**)*

```
ltask = ['ccdproc','zerocombine','imstatistics','flatcombine','response',
        'identify','reidentify','fitcoords','transform','illumination',
        'imcombine','lacos_spec','background','apall','dispcor','lacos_im',
        'standard','sensfunc','calibrate','wspectext','setairmass','imcalc']


for task in ltask:
    unlearn(task)
```

```
# Unlearn list of functions ------------------------------------
def unlearn_list(ltask):
    ltask = cklist(ltask)
    for task in ltask:
        unlearn(task)
```

Useful functions

```
# Check if object is list ---------------------

def cklist(obj):
    if not isinstance(obj, (list, tuple)):
        obj = [obj]
    return obj
```

```
# Delete list of files -------------------------

def delfiles(files):
    files = cklist(files)
    for nfile in files:
        if nfile is not None and os.path.exists(nfile):
            os.remove(nfile)
```

Warning: Operation would overwrite existing image (bias.fits)

Useful functions

```
# Change name of file with a given format ----------------------

def fmt_file(fmt, name):
    dirname  = os.path.dirname(name)
    rname    = os.path.basename(name)
    new_name = os.path.join(dirname, fmt % rname)
    return new_name
```

fmt_file('changed_%s', '/user/local/me/myfile.fits ')

'/user/local/me/changed_myfile.fits '

Read observations LOG table

```
# Python modules
from astropy.table import Table

# Read LOG table
tlog = Table.read('nightlog_2005-07-08.txt', format='ascii')
```

file	run	object	RA	Dec	equinox	UT	airm	instr	texp	PA	slit	grat	cenwA
r761744.fit	761744	bias	14:01:38.81	01:32:40.1	J2000.0	20:47	1.14	ISISB	0.0	180	1.03	R600B	4302
r761746.fit	761746	bias	14:01:38.74	01:32:39.3	J2000.0	20:48	1.14	ISISB	0.0	180	1.03	R600B	4302
r761748.fit	761748	bias	0:00:00.00	00:00:00.0	J2000.0	20:48	1.05	ISISB	0.0	180	1.03	R600B	4302
r761750.fit	761750	bias	13:47:15.55	17:27:24.7	J2000.0	20:49	1.05	ISISB	0.0	180	1.03	R600B	4302
r761752.fit	761752	bias	13:47:15.62	17:27:24.7	J2000.0	20:50	1.05	ISISB	0.0	180	1.03	R600B	4302
r761787.fit	761787	bd33	15:51:58.99	32:56:51.2	J2000.0	22:11	1.0	ISISB	240.0	76	1.02	R600B	4302
r761790.fit	761790	bd33	15:51:58.99	32:56:51.2	J2000.0	22:16	1.01	ISISB	240.0	76	1.02	R600B	4302
r761805.fit	761805	arc	14:51:14.73	35:32:31.0	J2000.0	23:09	1.14	ISISB	60.0	30	1.03	R600B	4302
r761806.fit	761806	flat	14:51:14.73	35:32:31.0	J2000.0	23:12	1.15	ISISB	30.0	30	1.03	R600B	4302
r761809.fit	761809	GALAXY	14:51:14.73	35:32:31.0	J2000.0	23:15	1.16	ISISB	1500.0	30	1.03	R600B	4302
r761810.fit	761810	GALAXY	14:51:14.94	35:32:29.5	J2000.0	23:41	1.23	ISISB	1500.0	30	1.03	R600B	4302
r761813.fit	761813	arc	14:51:14.94	35:32:29.4	J2000.0	00:08	1.33	ISISB	60.0	30	1.03	R600B	4302
r761815.fit	761815	flat	14:51:14.94	35:32:29.4	J2000.0	00:11	1.34	ISISB	30.0	30	1.03	R600B	4302
r761821.fit	761821	arc	21:59:41.89	26:25:56.7	J2000.0	01:11	1.27	ISISB	60.0	283	1.03	R600B	4302
r761822.fit	761822	flat	21:59:41.89	26:25:56.8	J2000.0	01:13	1.26	ISISB	30.0	283	1.03	R600B	4302
r761824.fit	761824	bd25	21:59:41.89	26:25:56.8	J2000.0	01:16	1.25	ISISB	100.0	283	1.03	R600B	4302
r761827.fit	761827	bd25	21:59:41.89	26:25:56.7	J2000.0	01:20	1.24	ISISB	100.0	283	1.03	R600B	4302
r761881.fit	761881	bias	19:16:42.76	63:58:12.6	J2000.0	05:31	1.64	ISISB	0.0	76	1.03	R600B	4302
r761882.fit	761882	bias	19:16:42.76	63:58:12.5	J2000.0	05:31	1.65	ISISB	0.0	76	1.03	R600B	4302
r761883.fit	761883	bias	0:00:00.00	00:00:00.0	J2000.0	05:32	1.22	ISISB	0.0	76	1.03	R600B	4302
r761884.fit	761884	bias	0:00:00.00	00:00:00.0	J2000.0	05:32	1.0	ISISB	0.0	76	1.03	R600B	4302
r761885.fit	761885	bias	0:00:00.00	00:00:00.0	J2000.0	05:33	1.0	ISISB	0.0	76	1.03	R600B	4302

~

BIAS

```
fbias = 'bias.list'

idbias = (tlog['object'] == 'bias')

wfile(fbias, tlog[idbias]['file'], suffix='s1')
```

```
$ cat bias.list

r761744.fit[1]
r761746.fit[1]
r761748.fit[1]
r761750.fit[1]
r761752.fit[1]
r761881.fit[1]
r761882.fit[1]
r761883.fit[1]
r761884.fit[1]
r761885.fit[1]
```

file	run	object	
r761744.fit	761744	bias	14:0
r761746.fit	761746	bias	14:0
r761748.fit	761748	bias	0:0
r761750.fit	761750	bias	13:4
r761752.fit	761752	bias	13:4
r761787.fit	761787	bd33	15:5
r761790.fit	761790	bd33	15:5
r761805.fit	761805	arc	14:5
r761806.fit	761806	flat	14:5
r761809.fit	761809	GALAXY	14:5
r761810.fit	761810	GALAXY	14:5
r761813.fit	761813	arc	14:5
r761815.fit	761815	flat	14:5
r761821.fit	761821	arc	21:5
r761822.fit	761822	flat	21:5
r761824.fit	761824	bd25	21:5
r761827.fit	761827	bd25	21:5
r761881.fit	761881	bias	19:1
r761882.fit	761882	bias	19:1
r761883.fit	761883	bias	0:0
r761884.fit	761884	bias	0:0
r761885.fit	761885	bias	0:0

~

BIAS

```
# Check if there is something wrong with an image
imstatistics('@'+fbias, fields='image,mean,midpt,stddev,min,max')
```

# IMAGE	MEAN	MIDPT	STDDEV	MIN	MAX
r761744.fit[1]	1372.	1372.	4.828	1193.	2910.
r761746.fit[1]	1372.	1372.	5.412	1195.	5242.
r761748.fit[1]	1372.	1372.	4.756	1194.	2329.
r761750.fit[1]	1372.	1372.	4.855	1193.	3684.
r761752.fit[1]	1372.	1372.	4.755	1194.	2705.
r761881.fit[1]	1372.	1371.	4.566	1195.	2562.
r761882.fit[1]	1372.	1372.	4.83	1196.	3025.
r761883.fit[1]	1374.	1374.	5.184	1196.	4481.
r761884.fit[1]	1375.	1375.	4.359	1197.	2395.
r761885.fit[1]	1375.	1375.	4.291	1195.	2154.

Zerocombine (BIAS)

```
-->
--> lpar zerocombine
       input = "@bias.list"    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 = 3.0)            Lower sigma clipping factor
     (hsigma = 3.0)            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 = 0.0)            Value if there are no pixels
       (mode = "al")
```

BIAS

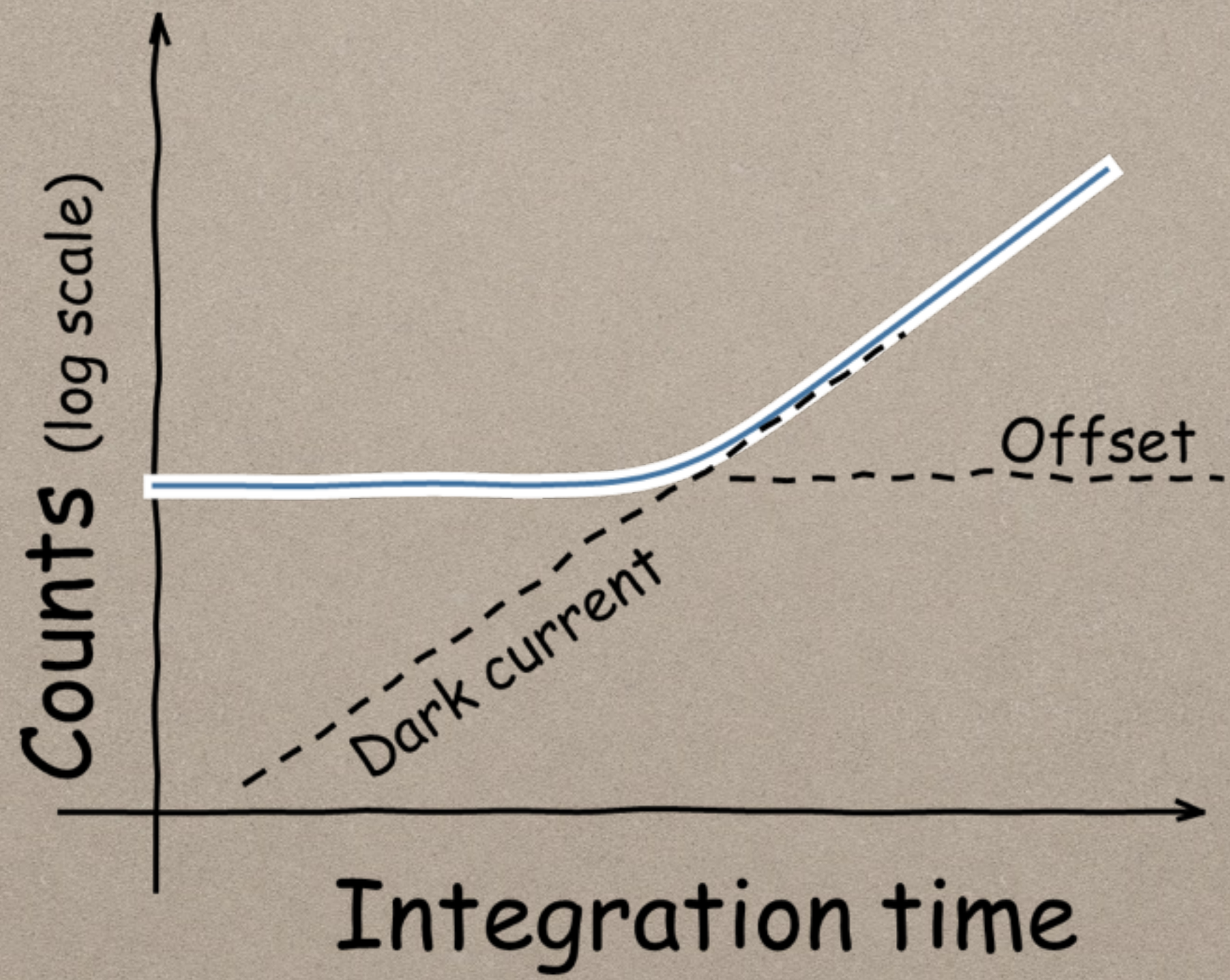
```
# Parameters ----------------------------------------------------------
rdnoise = 5.0
gain    = 1.16

# Name master bias ----------------------------------------------------
bias    = 'bias.fits'

# Delete output file if exists ----------------------------------------
delfiles(bias)

# Create master bias  -------------------------------------------------
zerocombine('@'+fbias, output=bias, combine='average',
            ccdtype="", rdnoise=rdnoise, gain=gain)
```

DARK FRAME

Bias frame

Flat-Field

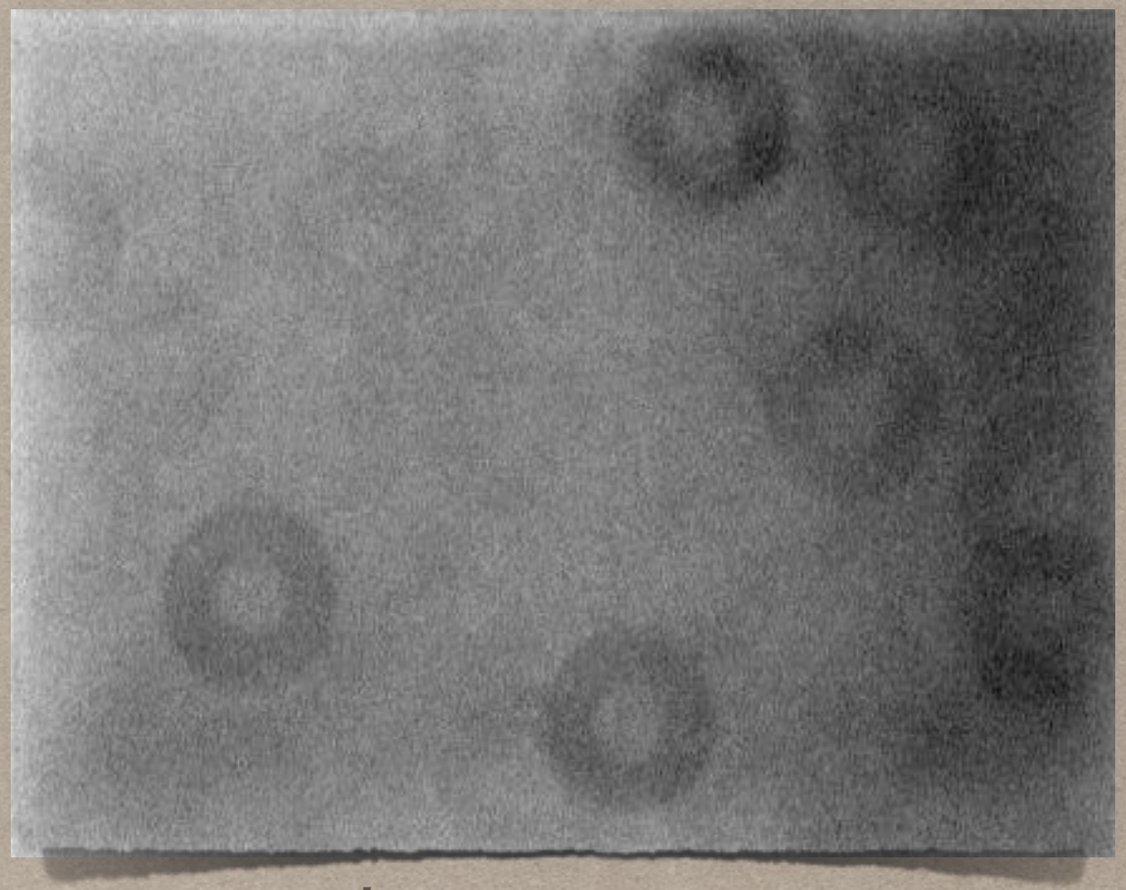
[Dark frame]



FLAT

Frame - <bias>

<flat> - <bias>



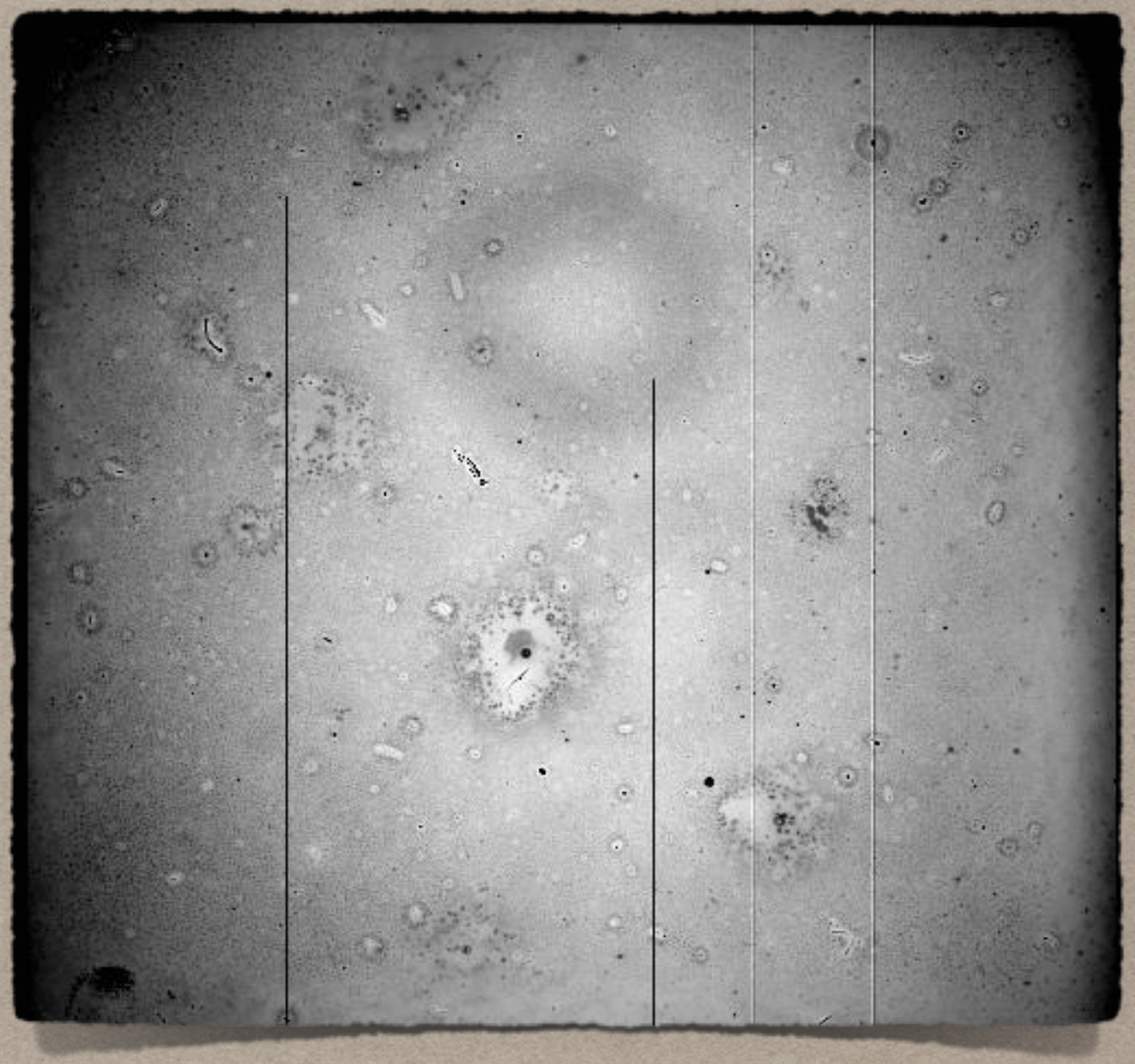
Photometry

Flat-Field is used to correct the different quantum efficiency of different pixel and also correct the non-uniformity from some optics (such as filter, etc)

FLAT

Frame - <bias>

<flat> - <bias>

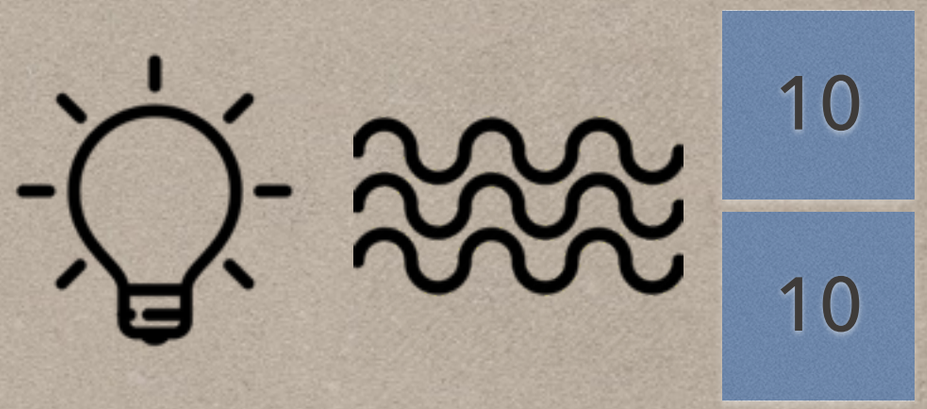


Flat-Field is used to correct the different quantum efficiency of different pixel and also correct the non-uniformity from some optics (such as filter, etc)

FLAT

Frame - <bias>

<flat> - <bias>

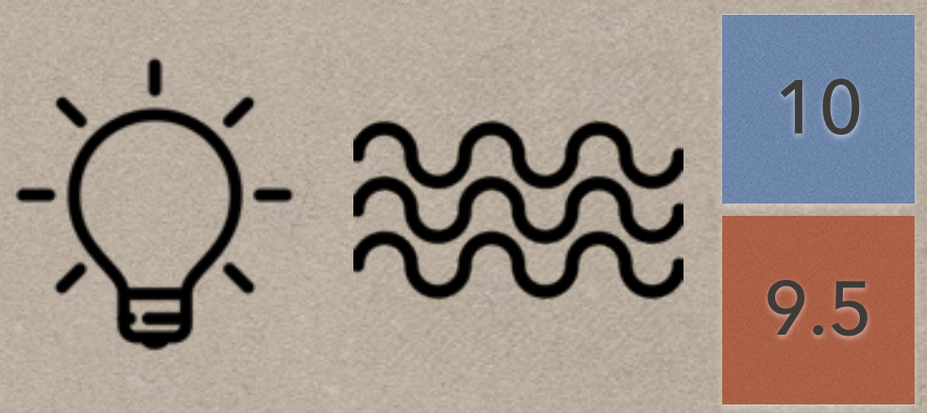


Flat-Field is used to correct the different quantum efficiency of different pixel and also correct the non-uniformity from some optics (such as filter, etc)

FLAT

Frame - <bias>

<flat> - <bias>

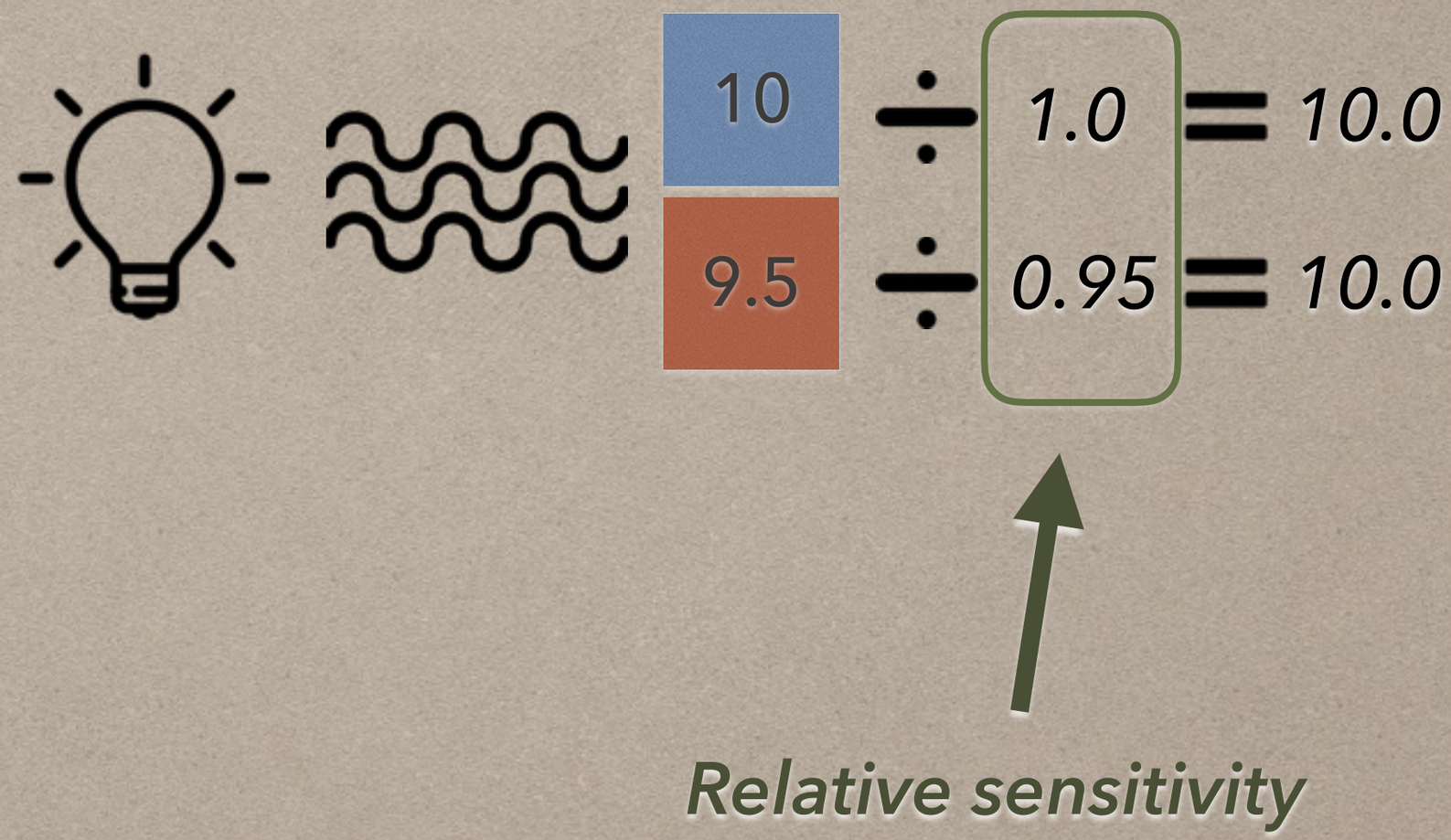


Flat-Field is used to correct the different quantum efficiency of different pixel and also correct the non-uniformity from some optics (such as filter, etc)

FLAT

Frame - <bias>

<flat> - <bias>



Flat-Field is used to correct the different quantum efficiency of different pixel and also correct the non-uniformity from some optics (such as filter, etc)

FLAT

flatcombine, ccdproc & response

```
# Write flat field file list

fflat = 'flat.list'

idflat = (tlog['object'] == 'flat')

wfile(fflat, tlog[idflat]['file'], suffix='s1')
```

```
$ cat flat.list

r761778.fit[1]
r761786.fit[1]
r761806.fit[1]
r761815.fit[1]
r761822.fit[1]
r761830.fit[1]
```

Flat-Field: combine same filter (photometry) or same wavelength coverage (spectroscopy)

FLAT

flatcombine, ccdproc & response

```
def flatccdres(flat_files, flat='flat.fits', nflat='nflat.fits', cflat='cflat.fits', iccd='no', iresponse='yes',
      combine='average', reject='avsigclip', ccdtype="", process='no', fixpix='no', overscan='yes',
      zerocor='yes', trim='yes', darkcor='no', flatcor='no', illumco='no', fringec='no', function='legendre',
      corder=5, rorder=18, readaxi='column', biassec=biassec, zero=bias, trimsec=trimsec, low_reject=3,
      high_reject=3, scale='mode', lflats=None, suffix=None):

   unlearn('ccdproc')
   unlearn('flatcombine')

   delfiles(flat)
   if lflats is not None:
      wfile(flat_files, lflats, suffix=suffix)

   flatcombine('@'+flat_files, output=flat, combine=combine, reject=reject, ccdtype=ccdtype,
                  scale=scale, process=process, rdnoise=rdnoise, gain=gain)

   delfiles(cflat)
   ccdproc(images=flat, output=cflat, ccdtype=ccdtype, fixpix=fixpix, overscan=overscan, trim=trim,
            zerocor=zerocor, darkcor=darkcor, flatcor=flatcor, illumco=illumco, fringec=fringec,
            zero=zero, function=function, order=corder, interactive=iccd, readaxi=readaxi,
            trimsec=trimsec, biassec=biassec)

    delfiles(nflat)
   response(cflat, cflat, nflat, interac=iresponse, order=rorder, low_reject=low_reject, high_reject=high_reject)

   unlearn('ccdproc')
   unlearn('flatcombine')
```

FLAT

flatcombine

```
def flatccdres(flat_files, flat='flat.fits', nflat='nflat.fits', cflat='cflat.fits', iccd='no', iresponse='yes',
      combine='average', reject='avsigclip', ccdtype="", process='no', fixpix='no', overscan='yes',
      zerocor='yes', trim='yes', darkcor='no', flatcor='no', illumco='no', fringec='no', function='legendre',
      corder=5, rorder=18, readaxi='column', biassec=biassec, zero=bias, trimsec=trimsec, low_reject=3,
      high_reject=3, scale='mode', lflats=None, suffix=None):

   unlearn('ccdproc')
   unlearn('flatcombine')

   delfiles(flat)
   if lflats is not None:
      wfile(flat_files, lflats, suffix=suffix)

   flatcombine('@'+flat_files, output=flat, combine=combine, reject=reject, ccdtype=ccdtype,
                  scale=scale, process=process, rdnoise=rdnoise, gain=gain)

   ...........
```

```
->
-> lpar flatcombine
    input =                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 = "flat")       CCD image type to combine
  (process = yes)          Process images before combining?
  (subsets = yes)          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 = 3.0)          Lower sigma clipping factor
   (hsigma = 3.0)          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 = 1.0)          Value if there are no pixels
     (mode = "al")
```

FLAT

ccdproc

```
def flatccdres(lflats, flat='flat.fits', nflat='nflat.fits', cflat='cflat.fits', iccd='no', iresponse='yes', combine='average', reject='avsigclip', ccdtype="", process='no',
    fixpix='no', overscan='yes', zerocor='yes', trim='yes', darkcor='no', flatcor='no', illumco='no', fringec='no', function='legendre', corder=5, rorder=18,
    readaxi='column', biassec=biassec, zero=bias, trimsec=trimsec, low_reject=3, high_reject=3, scale='mode'):

    ...................

  delfiles(cflat)
  ccdproc(images=flat, output=cflat, ccdtype=ccdtype, fixpix=fixpix, overscan=overscan, trim=trim,  zerocor=zerocor, darkcor=darkcor, flatcor=flatcor,
        illumco=illumco, fringec=fringec, zero=zero, function=function, order=corder, interactive=iccd, readaxi=readaxi, trimsec=trimsec, biassec=biassec)

    ...................
```

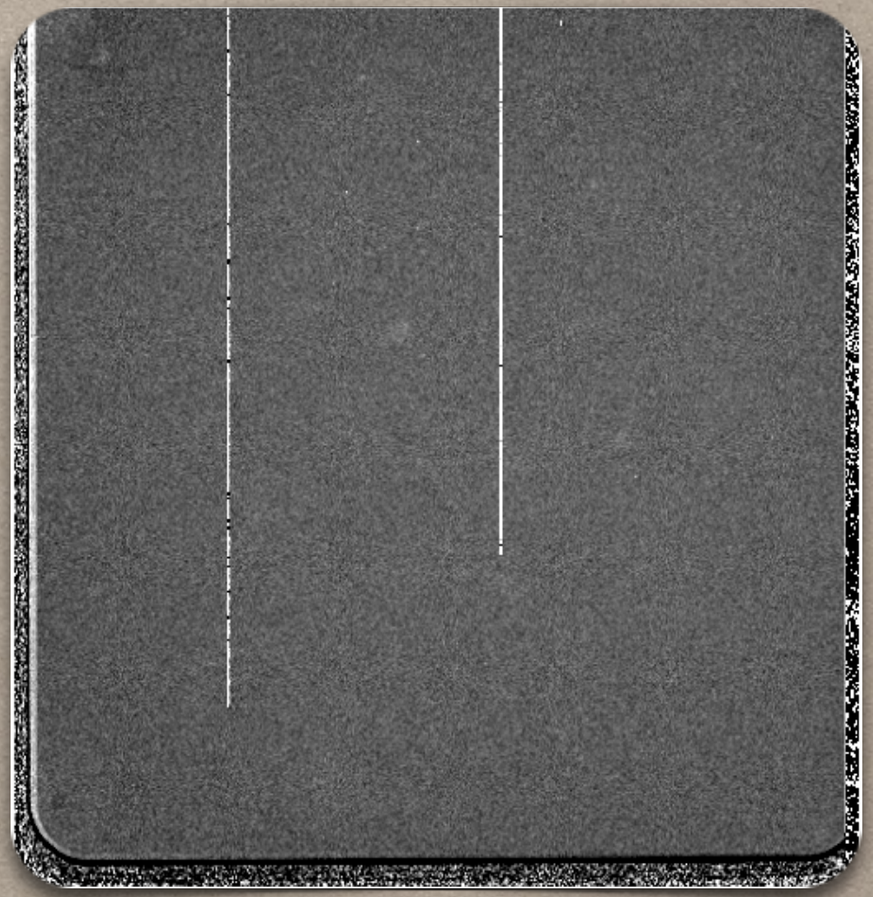
```
-> lpar ccdproc
      images = "@allimages.list" List of CCD images to correct
    (output = "c//@allimages.list") List of output CCD images
   (ccdtype = "object")       CCD image type to correct
 (max_cache = 0)              Maximum image caching memory (in Mbytes)
    (noproc = no)             List processing steps only?

    (fixpix = yes)            Fix bad CCD lines and columns?
  (overscan = yes)            Apply overscan strip correction?
       (trim = yes)           Trim the image?
   (zerocor = yes)            Apply zero level correction?
   (darkcor = yes)            Apply dark count correction?
   (flatcor = yes)            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?

  (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 = 1.0)            Minimum flat field value
  (scantype = "shortscan")    Scan type (shortscan|longscan)
     (nscan = 1)              Number of short scan lines
```

FLAT


```
biassec = '[316:1201,4105:4190]'

# From header
# trimsec = '[1:1201,1:4099]'
trimsec = '[1:1201,50:4000]'
```

BIASSEC: Overscan

TRIMSEC: Data



FLAT


```
biassec = '[316:1201,4105:4190]'

# From header
# trimsec = '[1:1201,1:4099]'
trimsec = '[1:1201,50:4000]'
```

BIASSEC: Overscan

TRIMSEC: Data

```
trimsec = fits.getval('r761783.fit', 'TRIMSEC', ext=1)
```

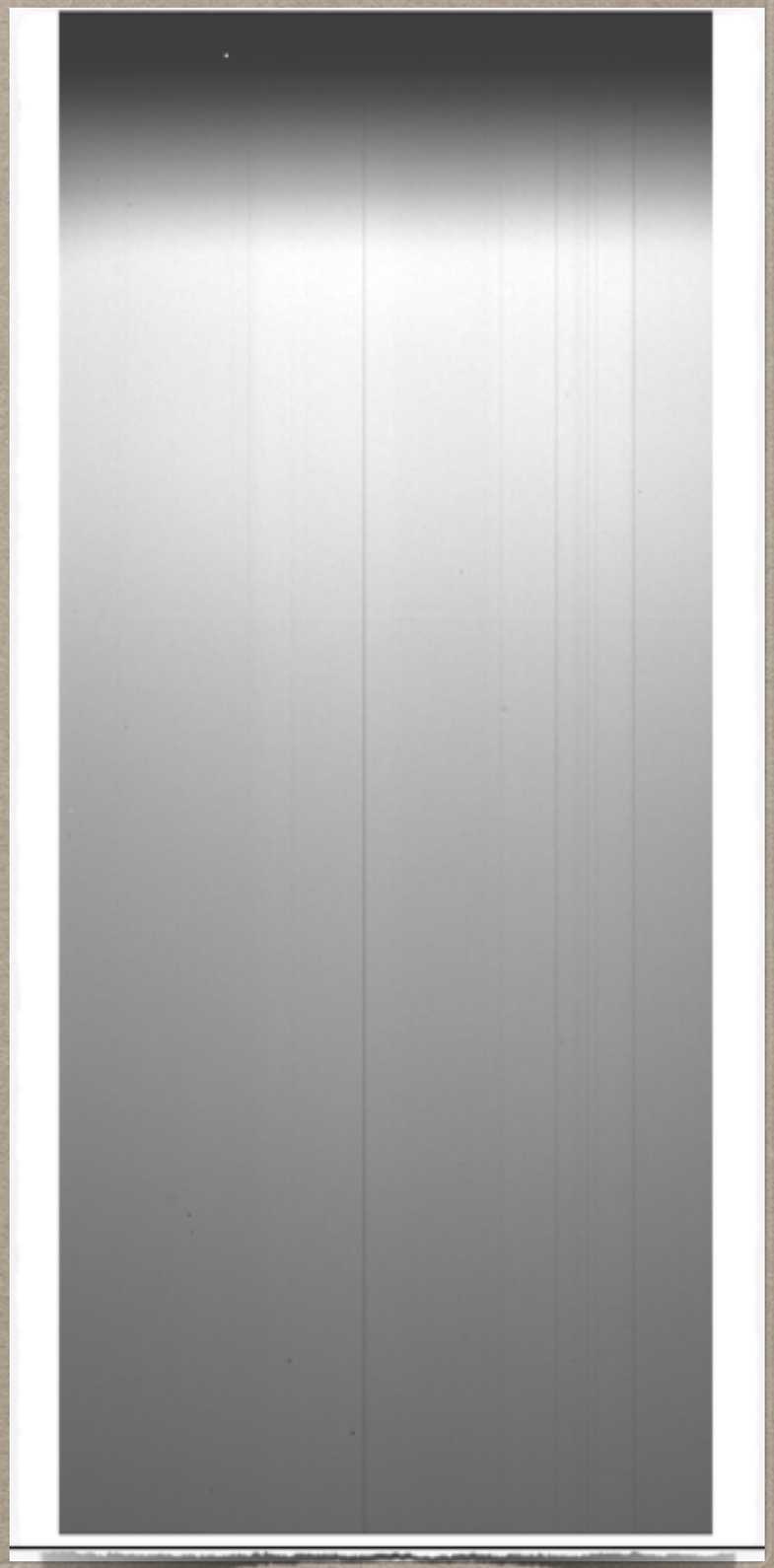
FLAT *spectroscopy*

Different structure in spatial and spectroscopic direction

Correct variations from pixel to pixel

Fit a smooth function to the average of all rows/columns in the spatial direction

response



FLAT *spectroscopy*

response

```
def flatccdres(lflats, flat='flat.fits', nflat='nflat.fits', cflat='cflat.fits', iccd='no', iresponse='yes',
    combine='average', reject='avsigclip', ccdtype="", process='no', fixpix='no', overscan='yes',
    zerocor='yes', trim='yes', darkcor='no', flatcor='no', illumco='no', fringec='no', function='legendre',
    corder=5, rorder=18, readaxi='column', biassec=biassec, zero=bias, trimsec=trimsec, low_reject=3,
    high_reject=3, scale='mode'):

    ...........

    delfiles(nflat)
    response(cflat, cflat, nflat, interac=iresponse, order=rorder, low_reject=low_reject,
        high_reject=high_reject)

    unlearn('ccdproc')
    unlearn('flatcombine')
```

response ***flat*** ***flat*** ***nflat***

calibration = normalization *response (output)*

FLAT *spectroscopy*

response

```
# Write flat field file list

fflat = 'flat.list'

idflat = (tlog['object'] == 'flat')

wfile(fflat, tlog[idflat]['file'], suffix='s1')
```

flatccdres(fflat)

Fit the normalization spectrum for cflat.fits interactively (yes):

ICFIT

:function [legendre|chebyshev|spline3|spline1]

:order n- order number of terms (polynomial, n +1)

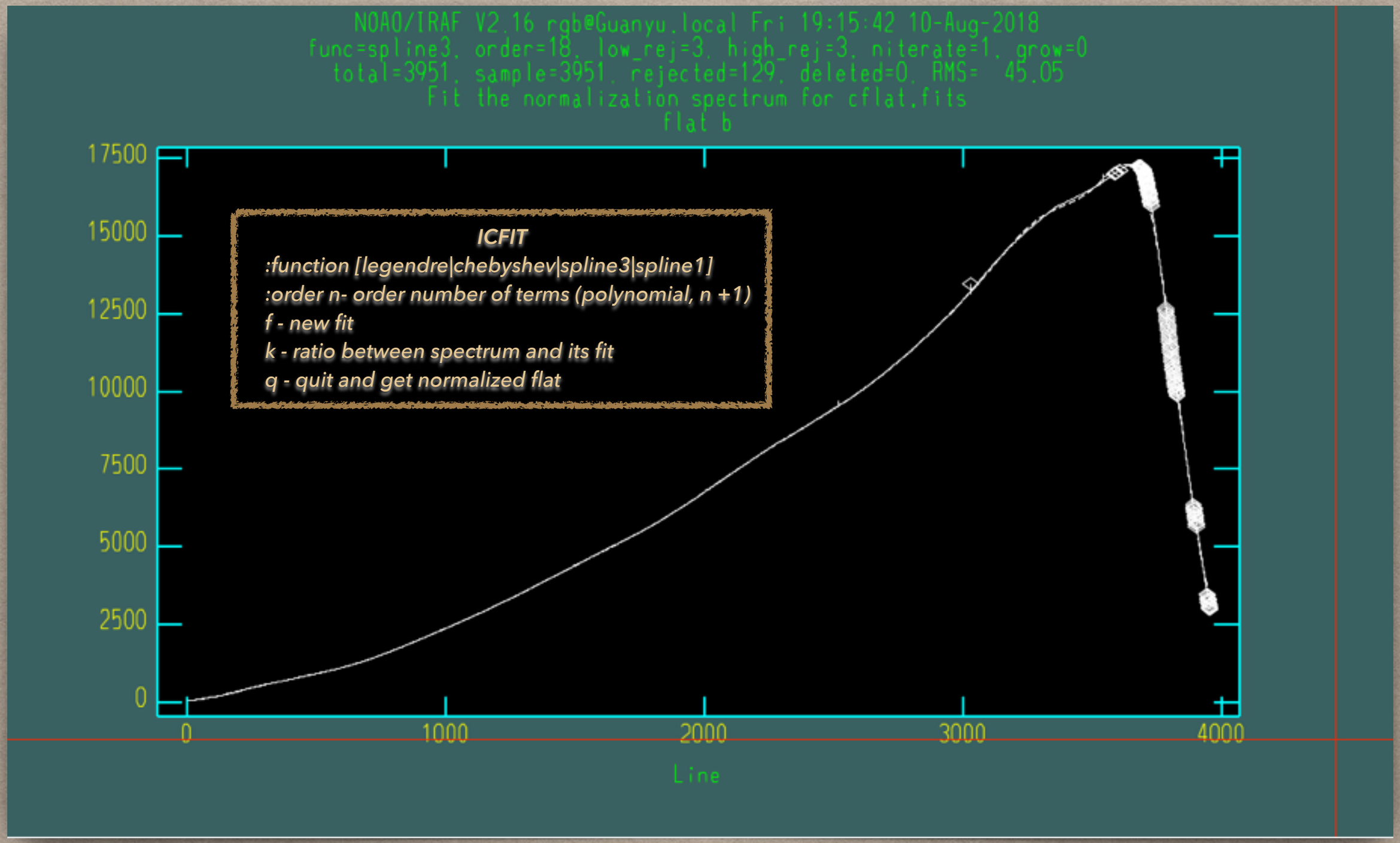
f - new fit

k - ratio between spectrum and its fit

q - quit and get normalized flat

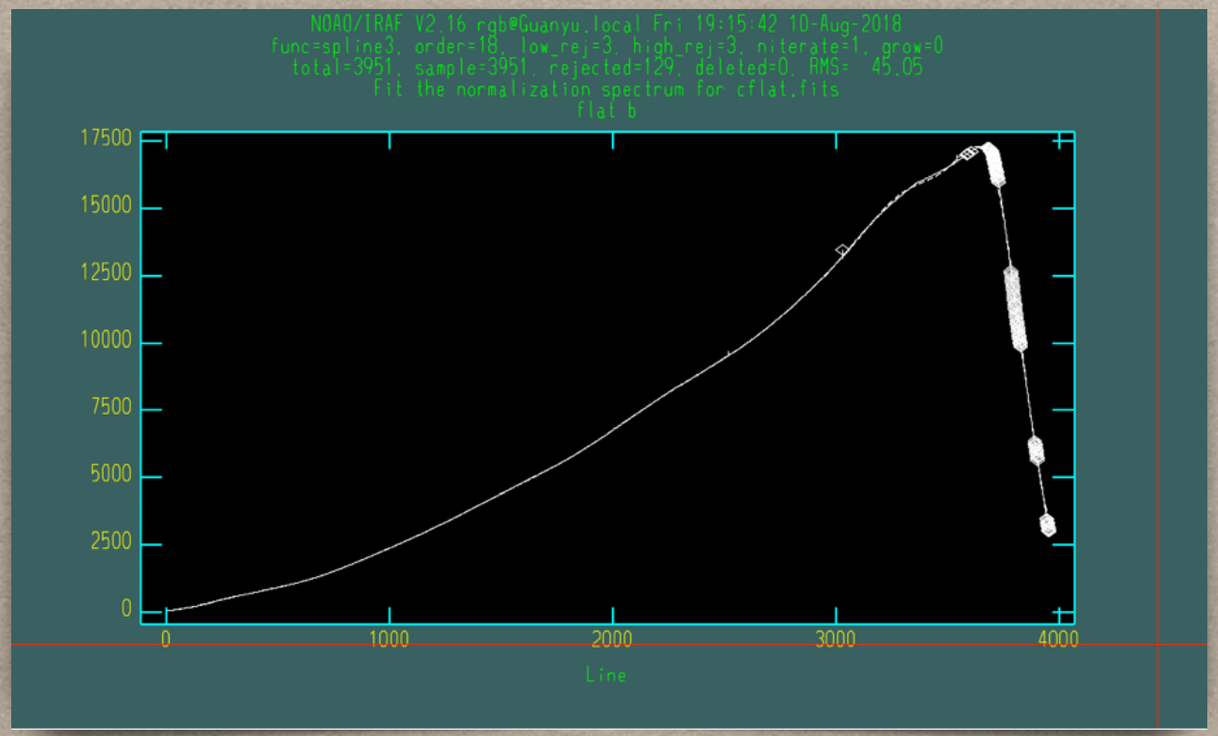
FLAT *spectroscopy*

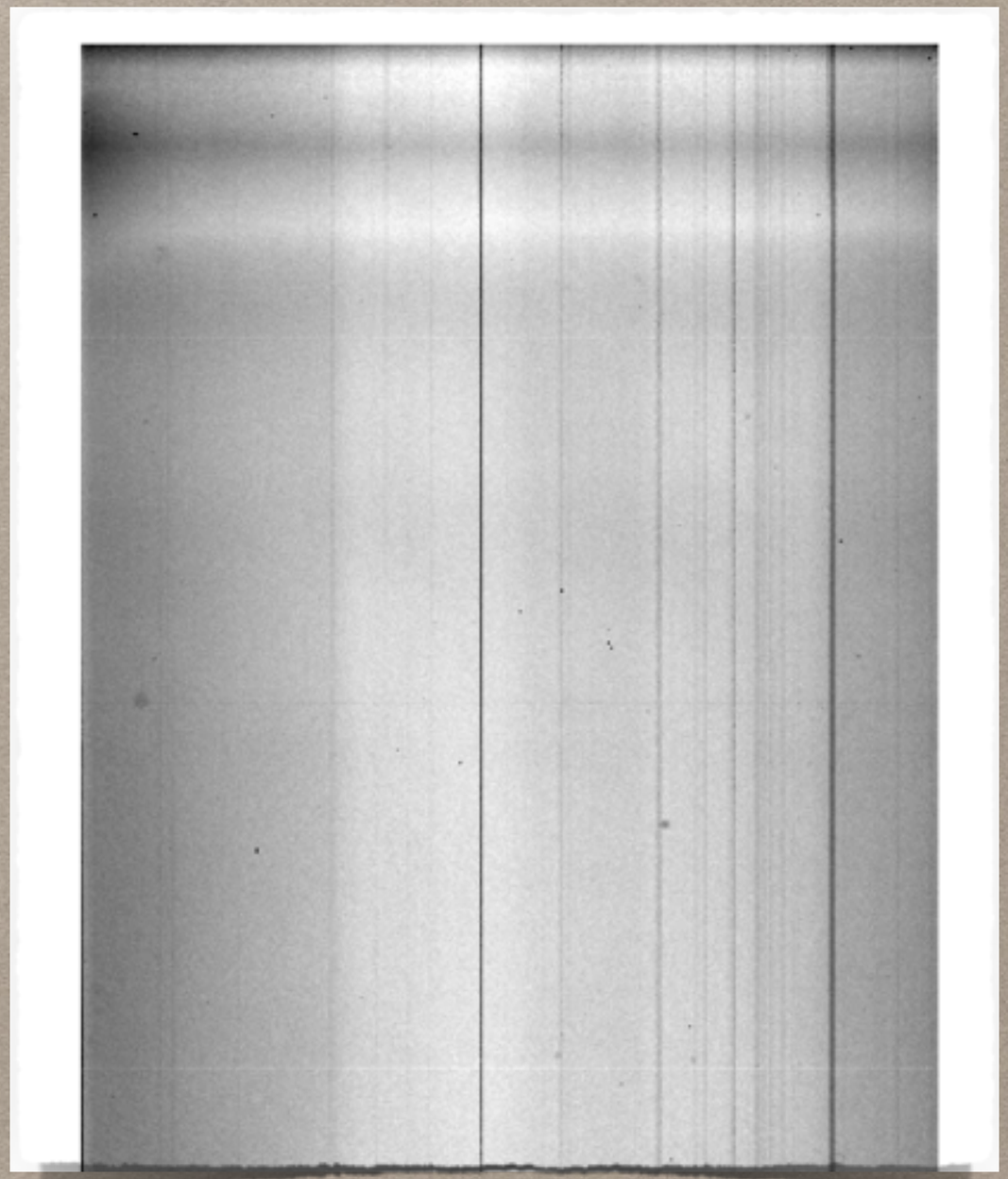
response



FLAT *spectroscopy*

response





Values around 1

Correct all images

ccdproc

Science, standard stars, …

```
def ccdp(files, suffix=None, input_files='tmp_input.list', output_files='tmp_output.list', **kwargs):

  # Dictionary with default parameters for ccdproc
  dpar = dict(ccdtype="", fixpix=no, overscan=yes, trim=yes, zerocor=yes, flatcor=no, darkcor=no,
          illumco=no, fringec=no, zero="", illum="", function='legendre', order=4,
          interactive=no, readaxi='column', trimsec=None, biassec=None, flat="")

  # Update parameters dictionary for ccdproc if new are provided (kwargs)
  dpar.update(kwargs)

  # Make "files" a list, in case there is only one file
  files = cklist(files)

  # Delete the temporary file lists
  delfiles([input_files, output_files])

  # Write the list of input files to a temporary file
  wfile(input_files, files, suffix=suffix)

  # Create a list of names with the output name after ccdproc is applied
  new_files_ccdproc = ['c%s' % item for item in files]
  # Write the new files list to a file
  wfile(output_files, new_files_ccdproc)

  # Delete, if exits, the new files to be created
  delfiles(new_files_ccdproc)

  # Apply ccdproc
  unlearn('ccdproc')
  ccdproc(images='@'+input_files, output='@'+output_files, **dpar)
  unlearn('ccdproc')

  # Delete the temporary lists
  delfiles([tmp_list, tmp_nlist])
```

Frame - <bias>

<flat> - <bias>

LONG-SLIT REDUCTION WITH PYRAF

Rubén García-Benito

IAA-CSIC

Part I

Optical Spectroscopy School
University of Rwanda
August 24th - September 2nd, 2018