The FITS Image Format and Image Display with DS9

Guillermo Damke (TA)
Steve Majewski (Course instructor)

ASTR 5110 – Fall 2011
University of Virginia

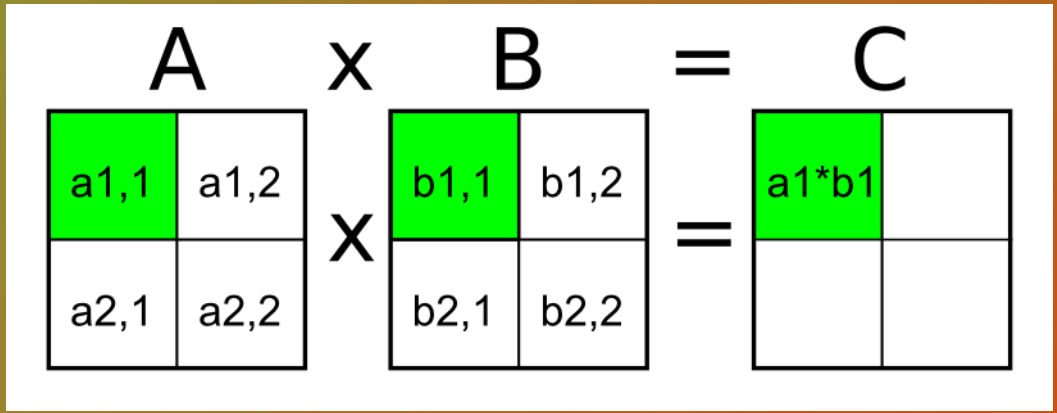
Part I. FITS Files

- FITS is the *standard* and *most used* data format in astronomy.
- FITS: Flexible Image Transport System.
- It is used for:
  - transporting,
  - analazing,
  - and archiving data.
- Primarily designed to store multi-dimensional arrays (like matrices) of data (images), or 2-D data tables (rows/columns) of information.
- Remember that array operations are different than matrix operations; the former work on an element-to-element basis.

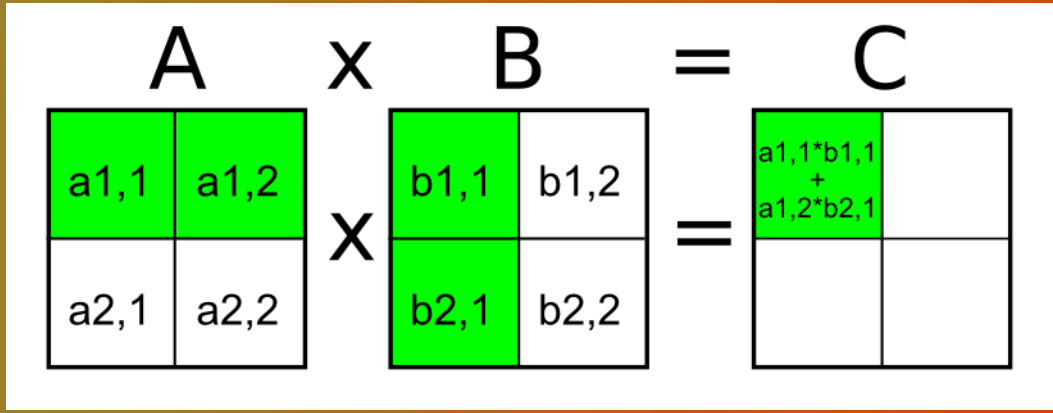
  The difference is most evident when you multiply an array by an array versus a matrix by a matrix.

FITS Files

Example: array multiplication vs. matrix multiplication.



2-d Array multiplication.
Size = m×n.
m = 2 (rows), n = 2(columns).
A and B must have the same size.
C will have the size of A and B.



2-d Matrix multiplication.
Size = m×n.
m = 2 (rows), n = 2(columns).
Size(A) may be m×n,
and Size(B) may be n×p.
Then, C will have the size m×p

FITS basic structure

- A FITS file is comprised of *Header Data Units* (HDUs).
  - The first HDU is called the *Primary HDU* or *Primary Array*.

    It always is present in a FITS file.

  - An arbitrary number (which may be zero) of HDUs may follow the Primary HDU.

    These HDUs are called *Extensions*.

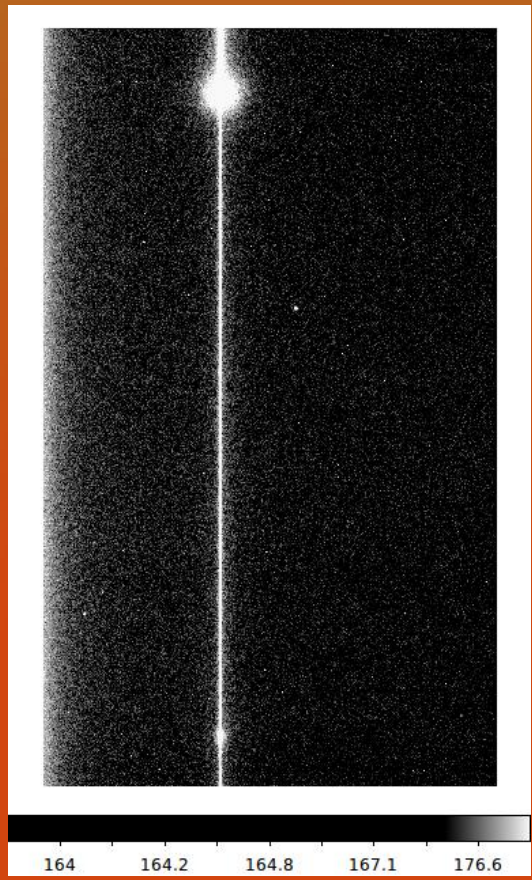
HDU structure

- A *Header Data Unit* consists of:
  - A *Header Unit, and*
  - A *Data Unit.*
- The Header Unit is composed of ASCII entries, and
- The Data Unit follows the Header, is optional, and contains your data as an array of binary data, or as data tables.

HDU structure

A Header Unit and a Data Unit (2-d) from the Swope telescope (Las Campanas Obs.)
This FITS file contains only a Primary HDU.

```
[gjd3r@SIM sim]$ imhead imaging/dec01/ccd171.fits
SIMPLE  =                    T                  / FITS STANDARD
BITPIX  =                   16                  / FITS BITS/PIXEL
NAXIS   =                    2                  / FITS NUMBER OF AXES
NAXIS1  =                 2064                  / FITS PIX PER ROW
NAXIS2  =                 3456                  / FITS PIX PER COL
CHIP    = 'SITe3'                              / DETECTOR NAME
TEL     = 'LCO-40'                             / TELESCOPE NAME
OBJECT  = 'G1003+1652'                         / OBJECT NAME
COMMENT = ''                                   / COMMENT
UTSTART = '07 25 16'                           / UT OF START FROM PC
UTEND   = '07 25 46'                           / UT OF END FROM PC
FILTERP =                    3                 / FILTER POSITION
FILTER  = '51'                                 / FILTER NAME
CCDPICNO=                  171                 / FRAME NUMBER OF IMAGE
EXPTIME =               30.030                 / TRUE EXP-TIME IN SEC.
GAIN    =                    3                 / GAIN NOT NEC. E/DN
LOOP    =                    1                 / LOOP SIZE
LOOPCTR =                    1                 / LOOP COUNTER
DATE-OBS= '19Dec01'                            / UT DATE AT START OF FRAME
RA      = '10:03:22.11'                        / RA FROM TEL
DEC     = ' 16:52:26.1'                        / DEC FROM TEL
HRA     = 'E 01 31 27'                         / HR-ANGLE FROM C40
EQUINOX =             2001.964                 / UNKNOWN
AIRMASS =                 1.58                 / AIRMASS FROM C40
IMAGETYP= 'object'                             / IMAGE TYPE
DISPAXIS=                    1                 / DISPERSION AXIS
END
[gjd3r@SIM sim]$
```



Header Unit

Data Unit

Header structure

- A Header contains an arbitrary number of entries in ASCII.
  They describe the corresponding Data Unit.
- Each entry is 80 characters long.
  These entries are called ``card images''. The usual format is:
  - *KEYWORD = value / comment string*
- *KEYWORD* name may only contain [AZ09_-], 8 characters long.
- *Value* may be an integer, float, complex (2 floats), boolean (T or F).
- A *Comment string* is not mandatory, but it is good practice to describe your data well!
- Furthermore, you can add *COMMENT* and/or *HISTORY* keywords. These don't have the equal sign, and are ASCII entries.
- Data reduction packages may store the steps an image has gone through in *HISTORY* keywords.

Header mandatory reserved keywords

- A Header describes the Data Unit data. A Header will start with the keywords:
  - SIMPLE = T or F / file conforms to FITS standard
  - BITPIX = 8,16,32,-32,-64 / number of bits per data pixel.
  - NAXIS = 0-999 / number of data axes.
  - NAXISn = m / length of data axis n. n depends on NAXIS.
  - EXTEND = T or F / File may contain extensions.
  - ...
  - END / last keyword in a header, it has no value nor comments.

Header reserved keywords

- It may happen that any image data type does not span the range of your data values. In this case, the data values can be linearly transformed and re-scaled to another values, or a physical value for example.
- These three reserved keywords do this for you: *BSCALE*, *BZERO* and *BUNITS*.

*physical_values = BZERO + BSCALE * array_values*

BUNITS is an ASCII string that contains the physical units after the linear transformation.

Usual optional keywords

- It is usual to find keywords describing the HDU creation and your observations:
  - DATE: date when the FITS file was generated. It is NOT when the image was obtained!
  - ORIGIN: The generator of the FITS file.
  - DATE-OBS: When your data was observed.
  - RA: Object Right Ascension, from Telescope Control System (TCS).
  - DEC: Object Declination, from TCS.
  - EQUINOX: RA and DEC equinox.
  - EPOCH: RA and DEC Epoch.
  - AIRMASS: Object airmass at moment of observation. From TCS.
  - TELESCOP: Telescope name.
  - INSTRUME: Instrument name.
  - OBJECT: Object name.
  - EXPTIME: Exposure time.
  - GAIN: Detector gain.
  - RDNOISE: Detector read noise.
  - OBSERVER: Observer name(s).
  - … etc.
  - These keywords depends on the Telescope/Instrument you use, but the ones above are typically standard.

Data structure

- A Data Unit follows a Header immediately.

  However, they are not required and we may have a Header Unit with no Data Unit.

- A Data Unit contains our data (e.g., spectrum, image, data-cube).

- There are 3 types of data:

  1. Image extension:

     - 8 (unsigned), 16 and 32 (signed) bit integers.
     - 32 bit (single) and 64 bit (double precision) floating point numbers.
     - 16 and 32 bit unsigned integers by adding a constant described in the header to the 16 and 32 bit signed integers.
     - Each element in the Data Array represents a *pixel* in our array (image), which is the basic element of an image.
     - Raw telescope images usually are in the form of integer data,
       but YOU MUST work with FLOAT data (more precision) when you reduce your data!

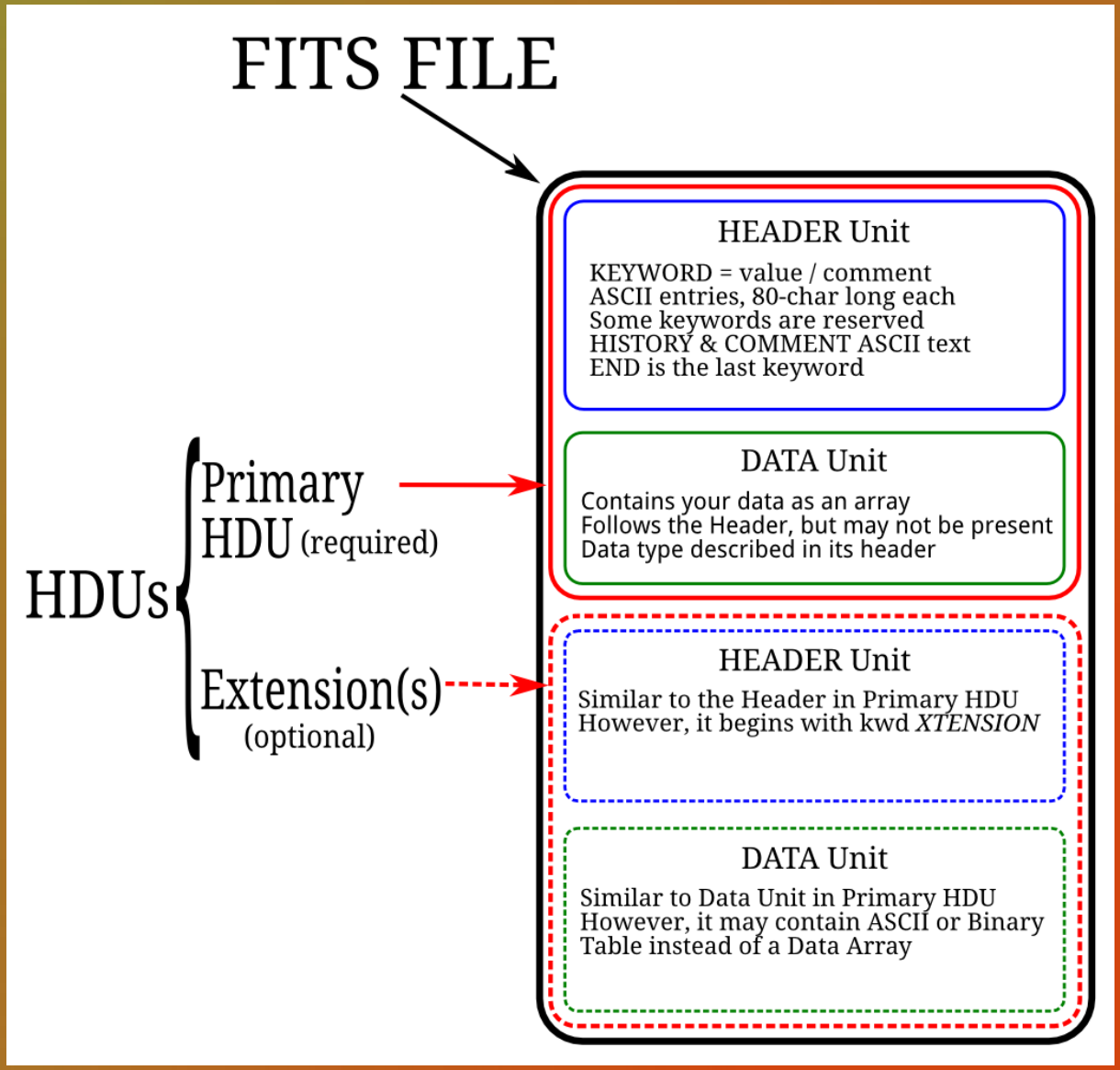
       The IRAF ccdproc routine will convert them for you (if set up properly).

Data structure

- 2. ASCII tables:
  - Allow data in tabular form (rows/columns)
  - The data type in each columns has to be constant.
- 3. Binary tables:
  - Similar to ASCII tables, but they use machine readable binary entries, which is faster to read and write.
  - Allows n-dimensional data tables.
- These data types in 2. and 3. are allowed in Extensions, not in the Primary HDU.

  Extensions will start with Header keyword *XTENSION*

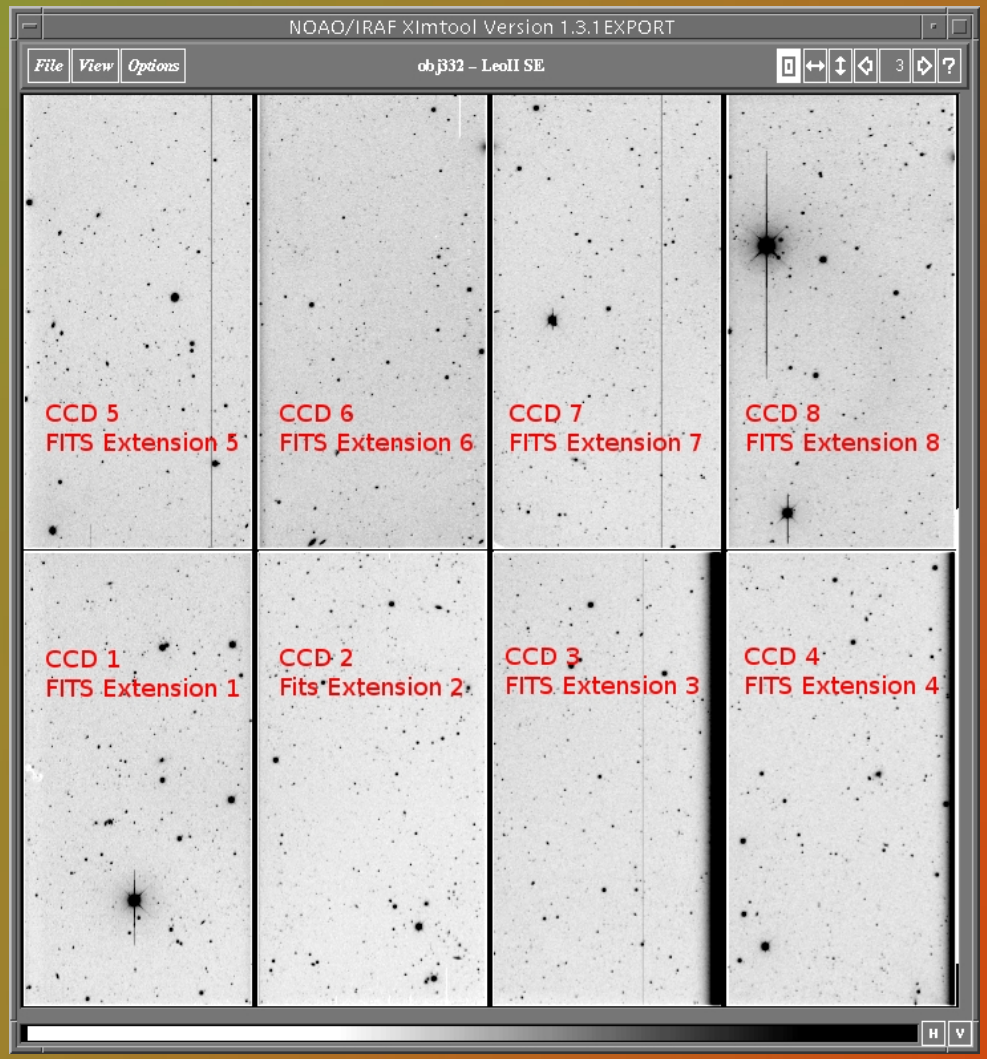
FITS summary



A multiple HDU example

- Some detectors, such as the MOSAIC 1 in the KPNO 4m, are an array of CCDs spacialy arranged in a mosaic.
  - This increases the usable field of view of the telescope.
  - Since we generally consider all individual CCDs in the mosaic to be contributing to the same image, it is useful to always operate on the data from each array identically and together, and so treat it all in one FITS file.
  - In this case, data from each individual CCD (each which has some different characteristics, such as readnoise and gain) are stored into an extension in the single FITS file.
  - The multiple extension FITS image keep the data from each detector separated for appropriate data reduction (i.e., keeping track of the differences in each array),
  - and still keeps data from a single exposure in a single FITS file.
  - We will see the Header for a MOSAIC 1 FITS file later.

A multiple HDU example



The FITS file that contains the data from KPNO MOSAIC 1 CCD camera contains 8 extensions, one per CCD.

We will have a look at the header in the next slide.

(Image: Steven Majewski)

A multiple HDU example

```
SIMPLE  =                    T / Fits standard
BITPIX  =                   16 / Bits per pixel
NAXIS   =                    0 / Number of axes
EXTEND  =                    T / File may contain extensions
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 2003' / FITS file originator
DATE    = '2007-06-13T22:17:36' / Date FITS file was generated
IRAF-TLM= '18:17:40 (13/06/2007)' / Time of last modification
NEXTEND =                    8 / Number of extensions
FILENAME= 'obj3052 '           / Original host filename
OBJECT  = 'And VII '           / Observation title
OBSTYPE = 'object  '           / Observation type
EXPTIME =              900.000 / Exposure time (sec)
RADECSYS= 'FK5     '           / Default coordinate system
RADECEQ =               2000. / Default equinox
RA      = '23:25:55.54'        / RA of observation (hr)
DEC     = '50:46:00.00'        / DEC of observation (deg)
DATE-OBS= '2006-09-17T03:52:10.0' / Date of observation start (UTC approximate)
TIME-OBS= '03:52:10.0'         / Time of observation start
MJD-OBS =       53995.16122685 / MJD of observation start
OBSERVAT= 'KPNO    '           / Observatory
TELESCOP= 'KPNO 4.0 meter telescope' / Telescope
TELRADEC= 'FK5     '           / Telescope coordinate system
TELEQUIN=              2000.0 / Equinox of tel coords
TELRA   = '23:25:55.55       ' / RA of telescope (hr)
TELDEC  = '50:46:00.0        ' / DEC of telescope (deg)
AIRMASS =               1.314 / Airmass
DETECTOR= 'CCDMosaThin1'       / Detector
DETSIZE = '[1:8192,1:8192]'    / Detector size for DETSEC
NCCDS   =                    8 / Number of CCDs
FILTER  = 'M Washington k1007' / Filter name(s)
TV1FOC  =  9.2000000000000E-02 / North TV Camera focus position
TV2FOC  = -2.6040000000000E+00 / South Camera focus position
ENVTEM  =  1.7200001000000E+01 / Ambient temperature (C)
CCDTEM  = -9.8000000000000E+01 / CCD temperature  (C)
OBSERVER= 'Patterson, Beaton'  / Observer(s)
END
```

The Primary Header (FITS file from Rachael Beaton)

This Primary Header does not have any Data Unit.

Note the NAXIS keyword.

However, this FITS file contains 8 extensions, one per CCD in the MOSAIC 1.

Note that the header cards here are for things that affect all of the CCDs in the image.

A multiple HDU example

```
XTENSION= 'IMAGE   '           / Image extension
BITPIX  =                   -32 / Bits per pixel
NAXIS   =                     2 / Number of axes
NAXIS1  =                  2048 / Axis length
NAXIS2  =                  4096 / Axis length
PCOUNT  =                     0 / No 'random' parameters
GCOUNT  =                     1 / Only one group
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 2003' / FITS file originator
EXTNAME = 'im1     '           / Extension name
IRAF-TLM= '18:17:40 (13/06/2007)' / Time of last modification
INHERIT =                     T / Inherits global header
DATE    = '2007-06-13T22:17:39' / Date FITS file was generated
OBJECT  = 'And VII '           / Name of the object observed
IMAGEID =                     1 / Image identification
EXPTIME =               900.000 / Exposure time in secs
CCDNAME = 'SITe #7298FBR06-01 (NOAO 14)' / CCD name
GAIN    =                   3.1 / gain expected for amp 112 (e-/ADU)
RDNOISE =                   7.0 / read noise expected for amp 112 (e-)
SATURATE=                 36000 / Maximum good data value (ADU)
WCSASTRM= 'kp4m.19981010T033325 (Tr 37 V) by L. Davis 19981010' / WCS Source
EQUINOX =                2000. / Equinox of WCS
WCSDIM  =                     2 / WCS dimensionality
CTYPE1  = 'RA---TNX'           / Coordinate type
CTYPE2  = 'DEC--TNX'           / Coordinate type
CRVAL1  =            351.48142 / Coordinate reference value
CRVAL2  =            50.766667 / Coordinate reference value
CRPIX1  =            4213.8307 / Coordinate reference pixel
CRPIX2  =            4149.4003 / Coordinate reference pixel
CD1_1   =        5.8820606e-07 / Coordinate matrix
CD2_1   =       -7.1352875e-05 / Coordinate matrix
CD1_2   =        -7.168726e-05 / Coordinate matrix
CD2_2   =        4.3587026e-07 / Coordinate matrix
TRIM    = 'Jan  2 12:36 Trim is [65:2112,1:4096]'
FIXPIX  = 'Jan  2 12:36 Fix +sat+bleed'
OVERSCAN= 'Jan  2 12:36 Overscan is [1:50,1:4096], mean 1334.229'
ZEROCOR = 'Jan  2 12:36 Zero is Zero.n3.fits[im1]'
FLATCOR = 'Jan  2 12:36 Flat is DFlatM.fits[im1], scale 12821.46'
CCDPROC = 'Jun 13 18:17 CCD processing done'
PROCID  = 'kp4m.20060917T035210V2'
OBJMASK = 'om3052/im1.pl'
SFLATCOR= 'Jun 13 18:17 Sky flat is SFlatM.fits[im1], scale 2047.65'
END
```

This Header belongs to the first extension, i.e. CCD 1 in the MOSAIC 1.

Note the first keyword.

Note the NAXIS keyword.

Note that many of the header cards here are for things that are particular to THIS individual CCD in this array, like gain and readnoise.

The Header of the first extension (FITS file from Rachael Beaton)

The WCS

- WCS: World Coordinate System
- The WCS describes a coordinate system as a layer projected over the usual 1-d, 2-d or 3-d pixel data.

  You need to reduce your data and solve for a WCS using, e.g.:

  - a calibration lamp spectrum (1-d),
  - an astrometric catalog (2-d),
  - astrometry + velocity (frequency or wavelength) calibration (3-d data cube, e.g., Integral Field Spectroscopy or radio data).

- The WCS is a 1-1 transformation from (x), (x,y) or (x,y,z) (image coordinates, in pixels), to, e.g., wavelength, RA-DEC, RA-DEC-Velocity, respectively.
- The WCS has become more sophisticated over time:

  Now it can take distortions into account, e.g., distortions in the focal plane.

- The WCS is stored in the header unit by using specific *KEYWORDS*.

Accesing the header

- We will usually access the header keywords through tasks in our favorite data reduction software (IRAF, Python, IDL, CASA, etc.).

  We will have an IRAF practical in the coming weeks!

  However, there are simple command-line tasks already installed on your UVa computer (and most astronomical centers) for simple interaction with the header, WCS and data.

- These tasks are the WCS Tools Poograms. You can install them in your personal computer from this website:

  http://tdc-www.harvard.edu/wcstools/

- You are recommended to have a look at this webpage and read the description of these tasks -- they may save you a lot of time!

Accesing the header

- The WCS tools contain groups of utilities under categories:
  - Images and WCS,
  - Source catalogs and images with WCS,
  - Image header utility programs,
  - Image utility programs,
  - Catalog utility programs, and
  - Utility programs.

Accesing the header

- For learning purposes, we will see a few tasks in WCS tools.

  In your Linux terminal, you can simply type (try it!):

  - % imhead image.fits
    # Returns the header.
  - % gethead image.fits kw1 kw2 …
    # Returns the values of kw1, kw2, etc.
  - % sethead image.fits kw1=val1 [/comment] kw2=val2 ...
    # Set a new KEYWORD with val1.
  - % keyhead image.fits kw1=newkw1 …
    # Rename a keywords in the header.
  - Also try *sky2xy* and *xy2sky*.

Part II. FITS data visualization

- The two principal tools for data visualization that we will use are DS9 and Ximtool.
- We will focus on DS9 for this practical.

  Download and install DS9 on your laptop!

  http://hea-www.harvard.edu/RD/ds9/
- Both programs can interact with IRAF, and they are GUI based.
- Mostly used for 2-D data display.

  - IRAF provides better tasks for 1-D (spectrum) visualization.

  - I'm not aware of what are the ``standard'' 3-D data display tasks.

FITS data visualization

- When we open an image, a program like DS9 will basically:
  - open the FITS file,
  - check the header for image dimensions,
  - apply any linear transformation of data array values (through the BZERO and BSCALE keywords),
  - read any WCS keywords, if present,
  - choose the lowest/highest values **to be displayed** from your data,

and apply a *Transfer Function (note, similar to but not the same as the Optical Transfer Function we learned about in class).*

You can adjust the display range values interactively in DS9, which can be altered to bring up faint details, or washed out details, that you may miss when using the default display values.

FITS data visualization

It is usual to assign the limits to the **displayed data** as *z1* and *z2*. *z1* is the lowest value in the *Trasfer Function output*, and z2 is the highest. You will notice this when you use IRAF.

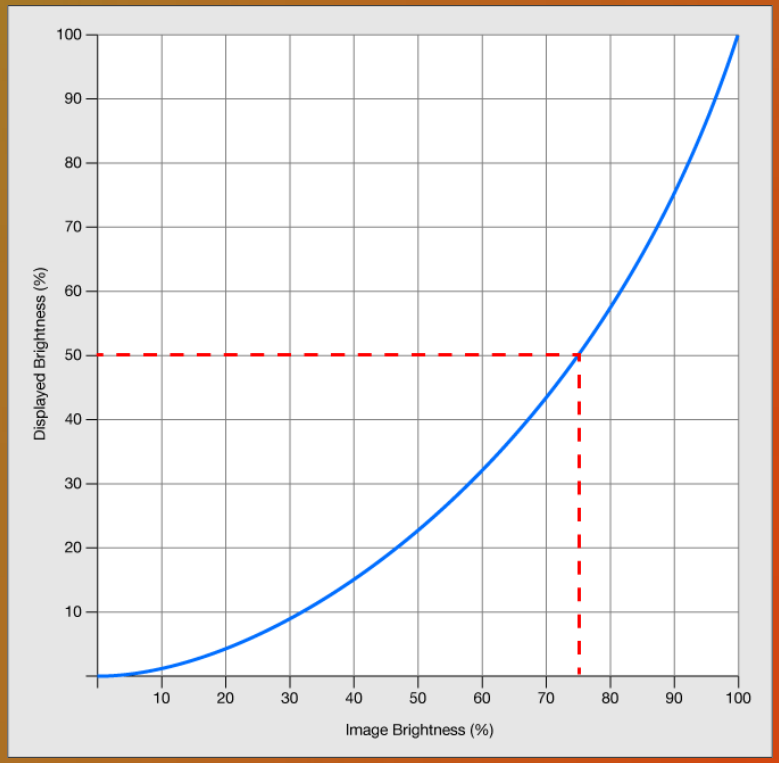
Depending on the *Transfer Function*, the values in between of z1 and z2 will be assigned different gray shades, representing the *intensity* of each pixel.

By default, values below z1 will be black, and values over z2 will be white.

Remember that your data may exceed the hardware -- and even the eye -- *dynamic range!*

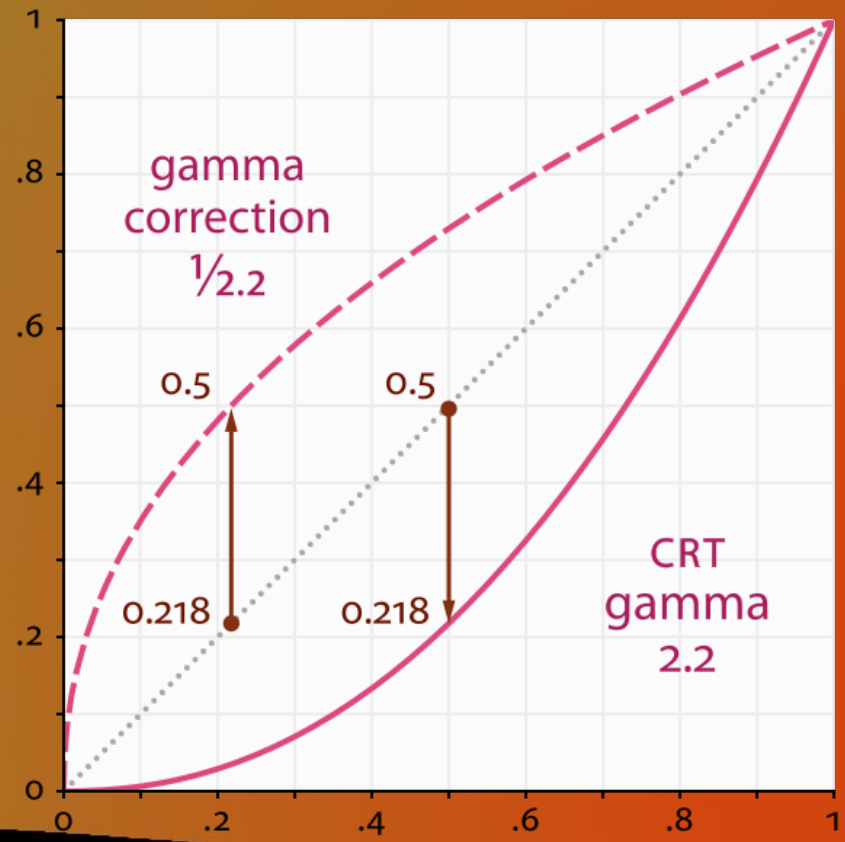
The transfer function

- Describes how the (input) data values in the data array are transferred to the relative intensity of the display (output).
- Can account for how the eye perceives light non-linearly.
- Also sometimes called the ”contrast function/curve”,”display function”, ”gamma function” (where gamma is often taken as a descriptor of the function when written as the power law $V_{out} = A\, V_{in}^{\gamma}$).



The transfer function

- Describes how the (input) data values in the data array are transferred to the relative intensity of the display (output).
- Can account for how the eye perceives light non-linearly.
- Also sometimes called the ”contrast function/curve”,”display function”, ”gamma function” (where gamma is often taken as a descriptor of the function when written as the power law $V_{out} = A\, V_{in}^{\gamma}$).



Images from Wikipedia.

The transfer function

- Describes how the (input) data values in the data array are transferred to the relative intensity of the display (output).
- Can account for how the eye perceives light non-linearly.
- Also sometimes called the ”contrast function/curve”,”display function”, ”gamma function” (where gamma is often taken as a descriptor of the function when written as the power law $V_{out} = A\, V_{in}^{\gamma}$).
- Of course, many shapes of transfer functions are possible, depending on your display goals.

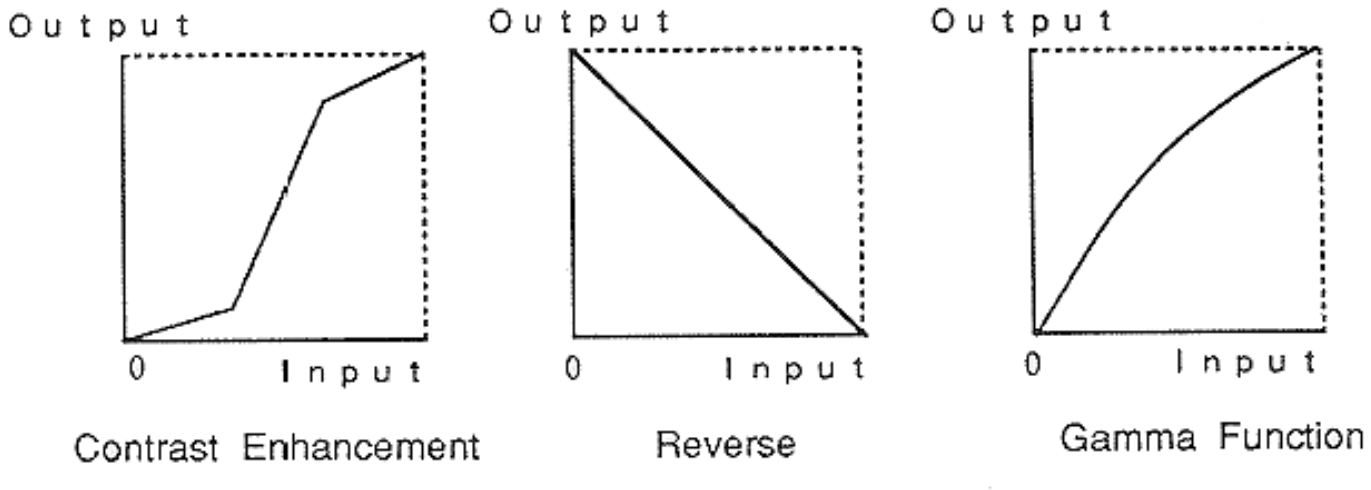


Figure 8.4.2 Examples of Lookup Table

Running DS9

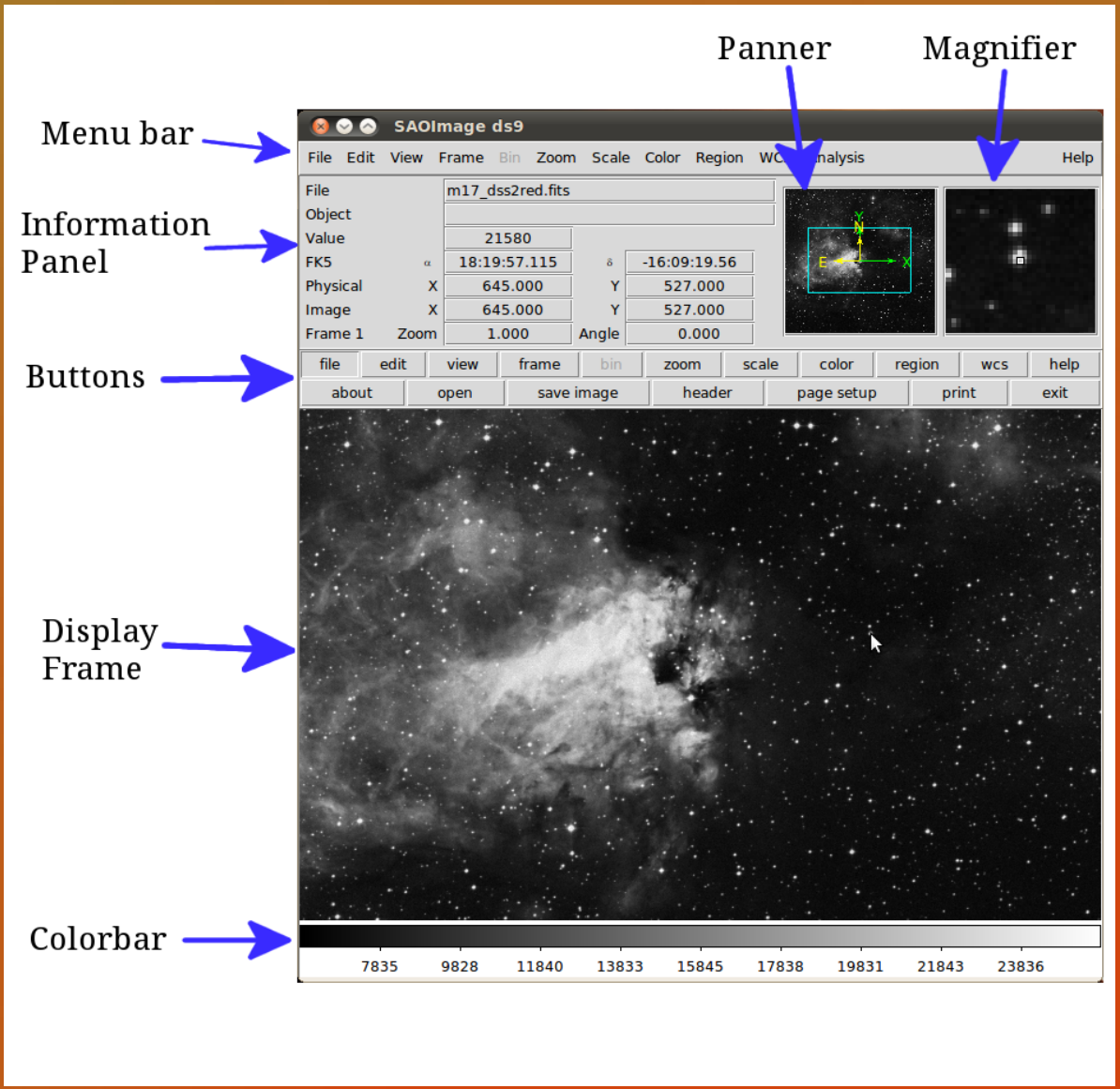
- Run DS9 by typing:

% ds9 image.fits &

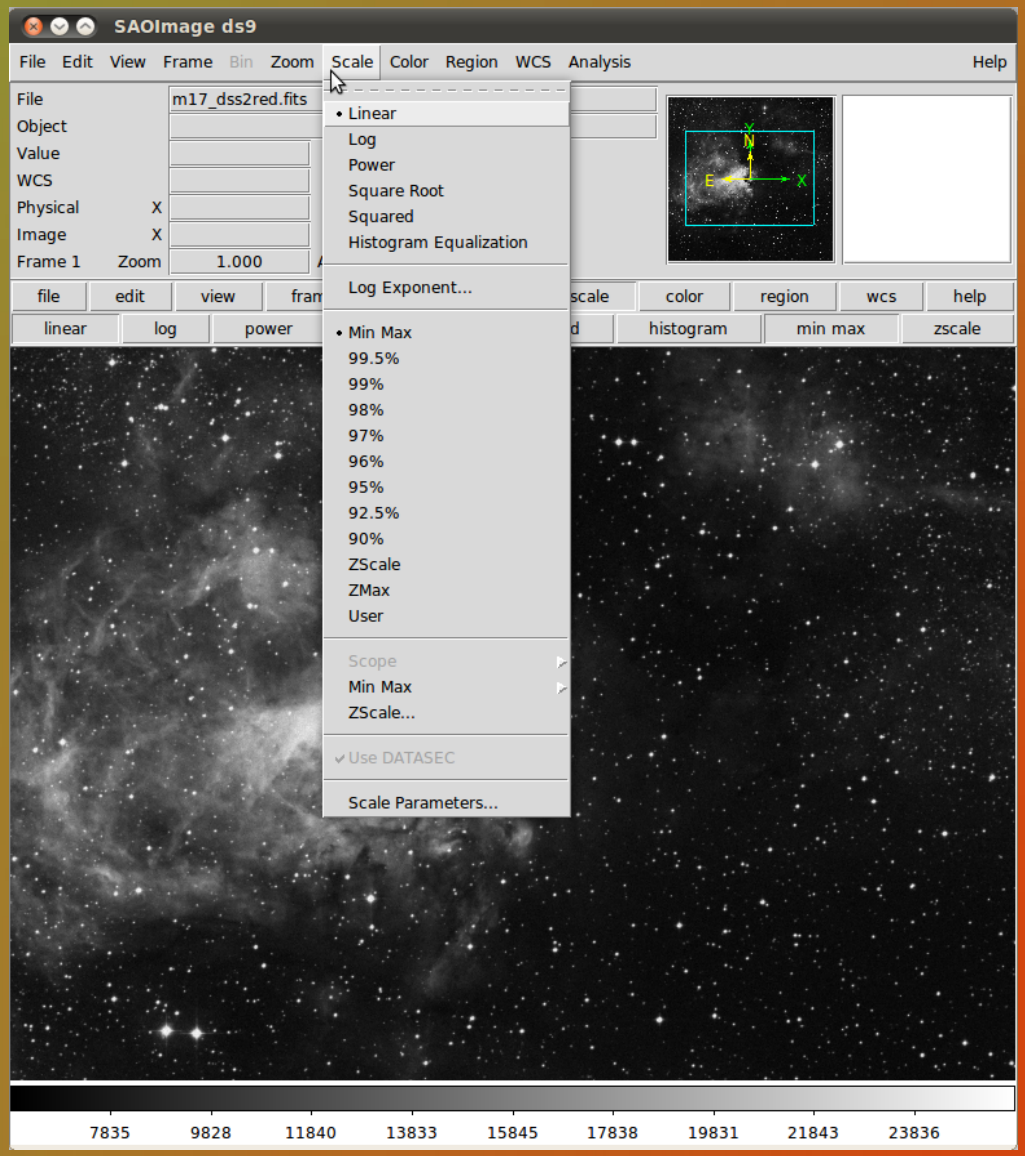
- Play with the 3 mouse buttons on the Display Frame!
- Left button → draw a region.
- Middle → Center the image.
- Right → Hold this button down and move it left/right and up/down.

  Pay attention to the image and the colorbar.

  What do you see?



The scale menu



Note that the *Scale menu* contains all the options for *Scale.*

The *Scale* option in *Buttons* only contains the commonly used options.

This applies for all menus/buttons.

The scale menu

- The *Scale* menu may be one of most useful DS9 menus.

  - It gives you the option of choosing a *Transfer Function* (TF) and the lowest/highest values displayed.

  - The TF relates the pixel values in our data and the intensity of the pixels displayed.

- Among the options for shapes of *Transfer Functions*, we find:

  Linear

  Logarithmic

  Power

  Square root

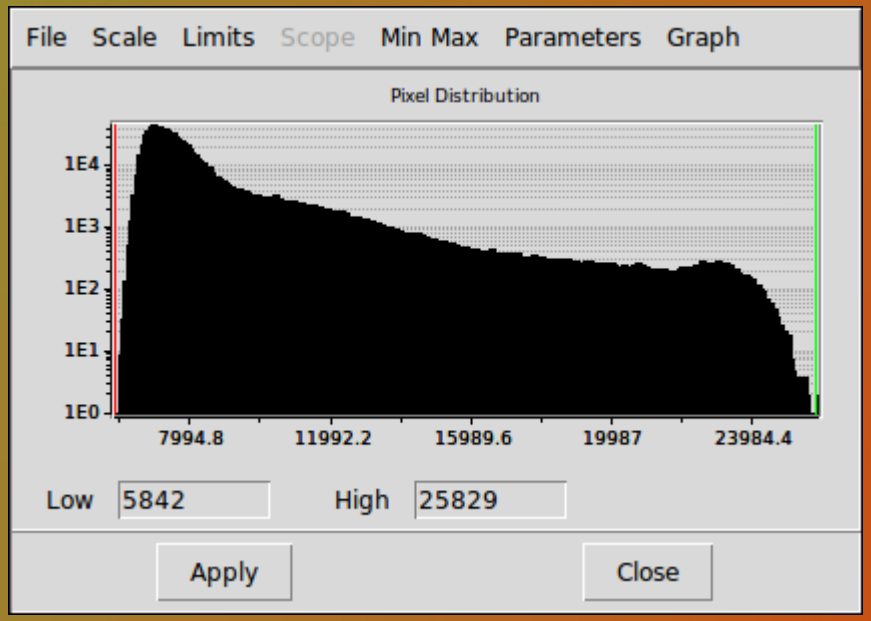
  Square

  Histogram equalization.

- The first five TF simply apply the given function to our image pixel values and assign the pixel intensity based on the resulting values.

The scale menu

- The Histogram Equalization (HistEq) calculates the pixel intensities to be displayed from a histogram of the data values.
- This is particularly useful when most of the interesting pixels have similar values, in this case, a HistEq will unveil these details in an image.
- However, it may increase the background noise in some cases.



A Histogram of the image shown in the previous DS9 figure (M17, DSS2 Red band).

A HistEq TF will assign gray shades for data values about the main peak of the histogram.

Remember that the x-axis shows the data value (pixel intensity).

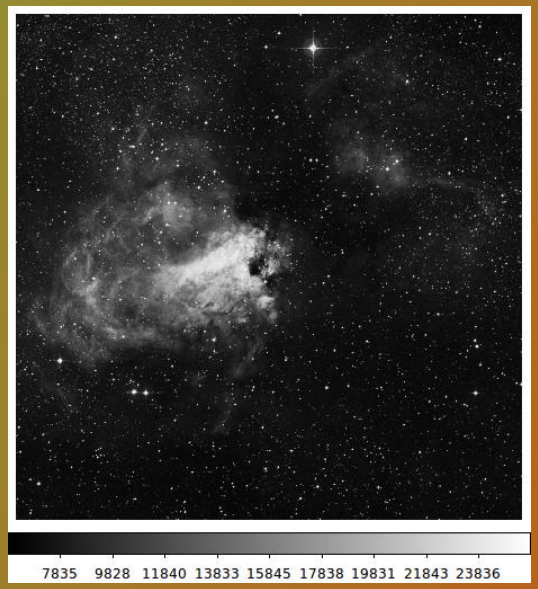
The y-axis shows the number of pixels that have a given intensity (here, shown on a logarithmic scale).

The scale menu

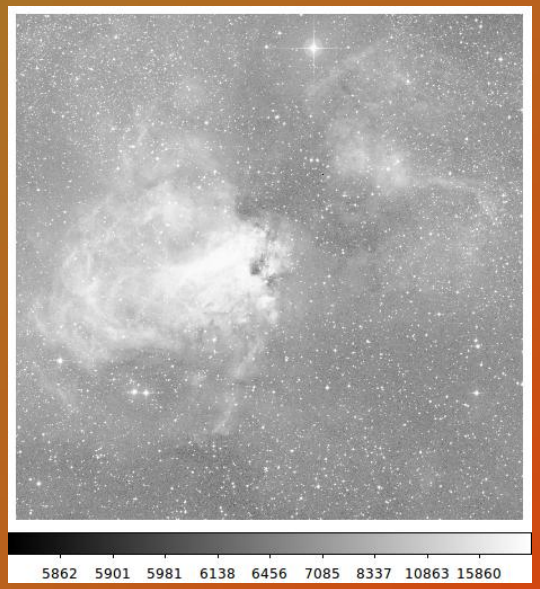
- Besides, you can choose how the lowest/highest values (the limits) of the color scale are selected.
- The option *minmax* uses 100% of the data to pick the limits.
- The options *99.5, 99,..., 90%* create a histogram of the data and uses the given percentage around the median value to pick the limits.
- Zscale uses the IRAF Zscale algorithm.
  - You can edit the parameters of the algorithm.
  - You can find this algorithm in the IRAF *display* task help.
- Zmax uses the lowest value from *Zscale*, and the highest value from *minmax*.

Some TF examples

DSS2 Red image of M17, showing three different TF. *Minmax was used for the three images.*



Linear

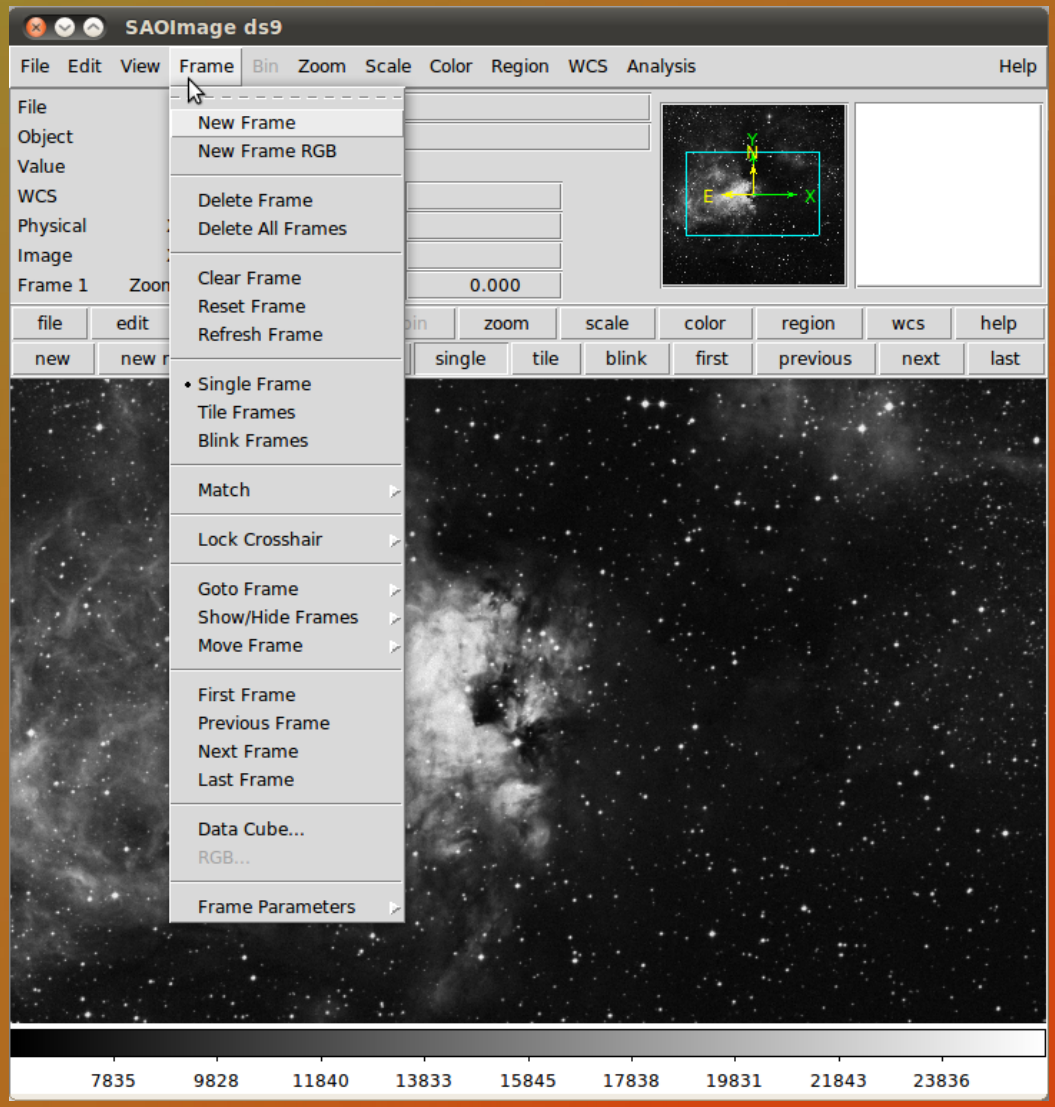


Logarithmic

Histogram Equalization

Note the values in the DS9 colorbar! The variations are the result of the selected TF.

Frame menu



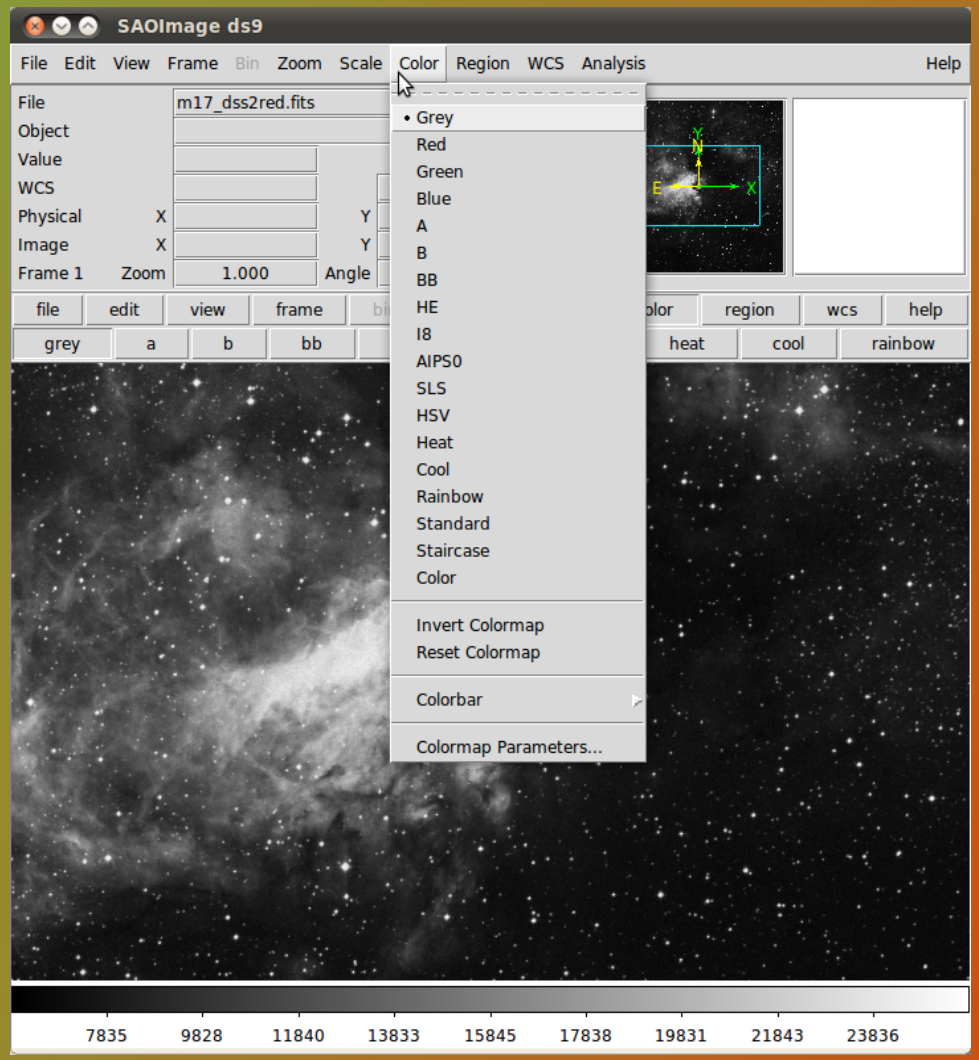
The *Frame* menu

Frame menu

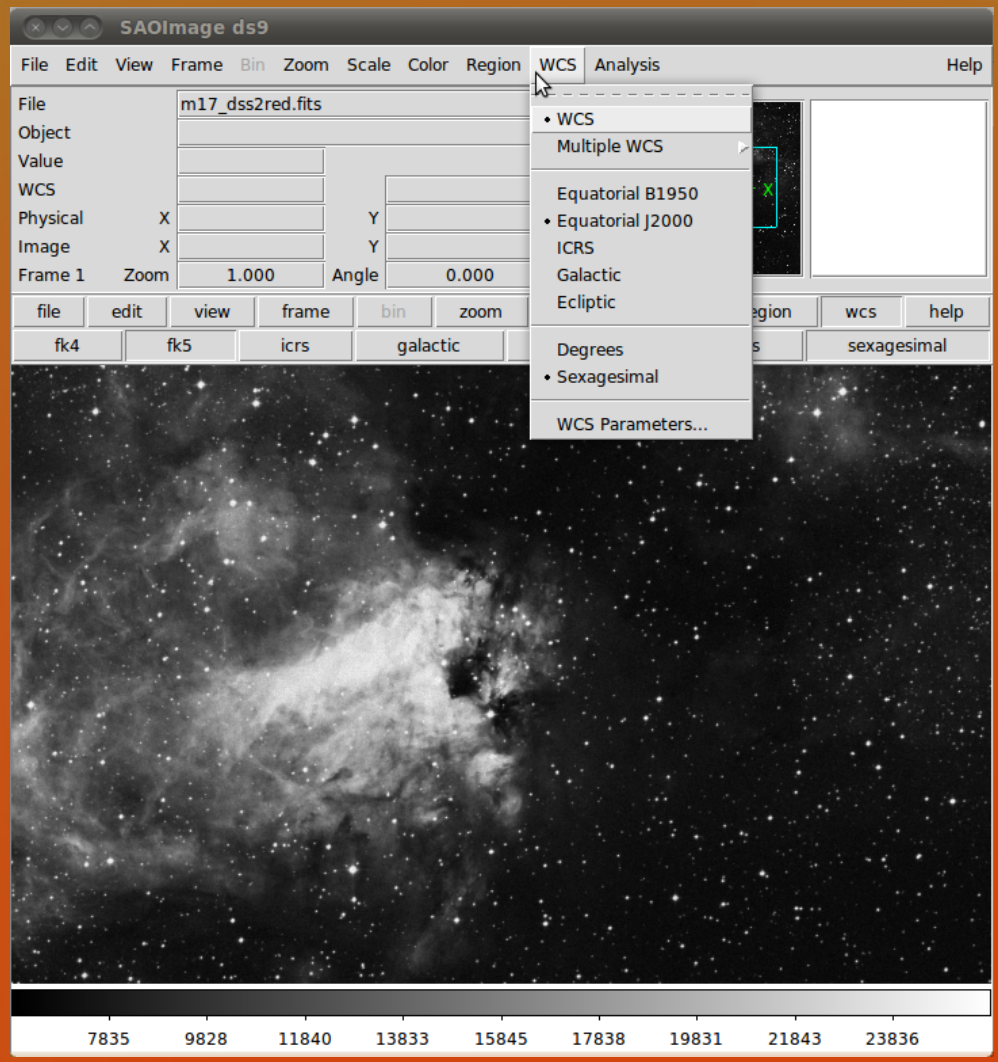
- This menu allows you to open more than one image in DS9.
  - Each image is loaded in a *Frame*.
  - E.g., it is useful when comparing multiple images of the same object.
- Try: Open two images of the same object in different filters into two different frames.
  - Then: go to *Frame*, and click *Blink.*
  - You can see how the object looks different in different filters, but are the images aligned?
- DS9 can use the WCS of the images to align them for you!

  -Try: Go to *Frame* in the *Menu bar,*
  - click on *Match,* then *Frame,* and finally on *WCS.*
  - Now try *Blink* again.
- If you have multiple frames opened, but you want to use blink on some of them, you can use *Show/hide frames* in the *Frame* menu.

Other useful menus



The *Color* menu



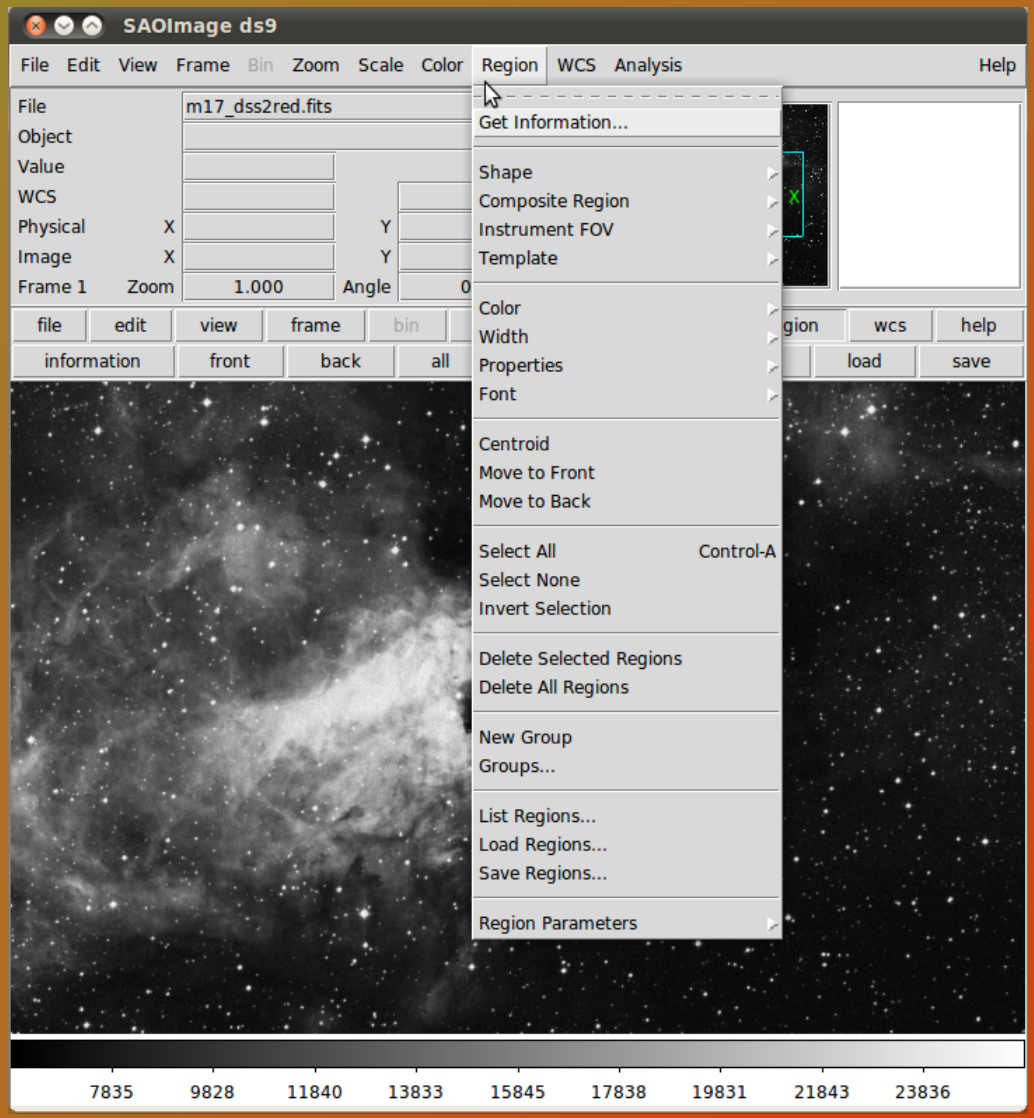
The *WCS* menu

Other useful menus

- The *Color* menu allows you to pick color maps different than the gray scale. Some of them may help you by increasing the dynamic range of what you see. Try some of them!

- The *WCS* menu, of course, only works if your image has a WCS specified in its Header Unit. Among the options, you can choose:
  - A different coordinate system (Equatorial, Galactic, Ecliptic, etc.)
  - Use decimal/sexagesimal degrees.
  - Try Galactic coordinates!

Other useful menus

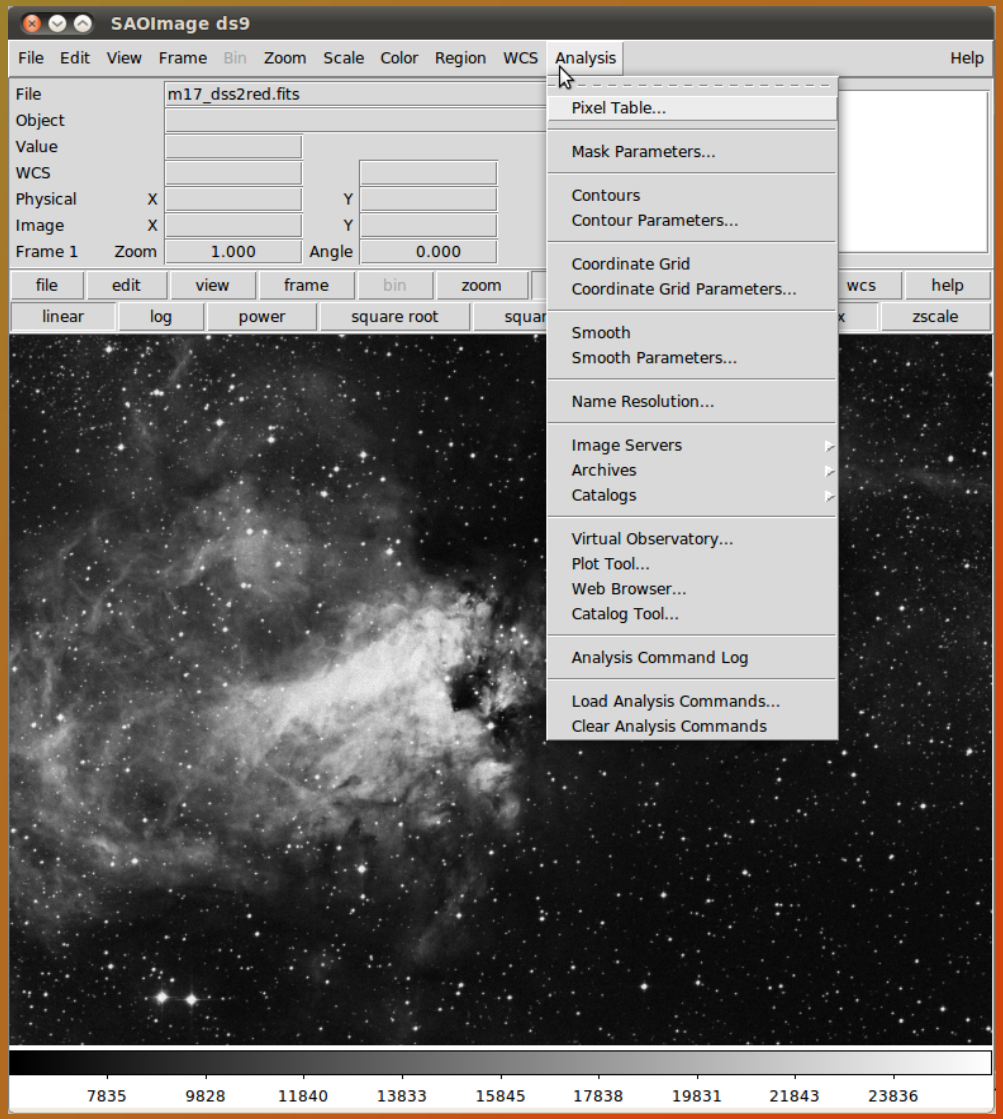


The *Region* menu

Other useful menus

- The *Region* menu allows you to select the properties of the shapes that you draw with the left button of your mouse.
- These regions may be very useful to create annotated figures of your images!
- You can create a region file in .reg format, and display the regions on top of another image.
- Always try to use *WCS* coordinates for this file. This makes them portable!
- Finally, these .reg files are just text files, this allows you to create a .reg file in your favorite programing language from other data (for example, a catalog), or even edit them in Emacs or Vim, and mark any object by loading a .reg file over your image.

Other useful menus



The *Analysis menu*

Other useful menus

- Finally, the *Analysis* menu provides tools such as:
  - *Contours*: Calculates and draws a contour plot over your image.

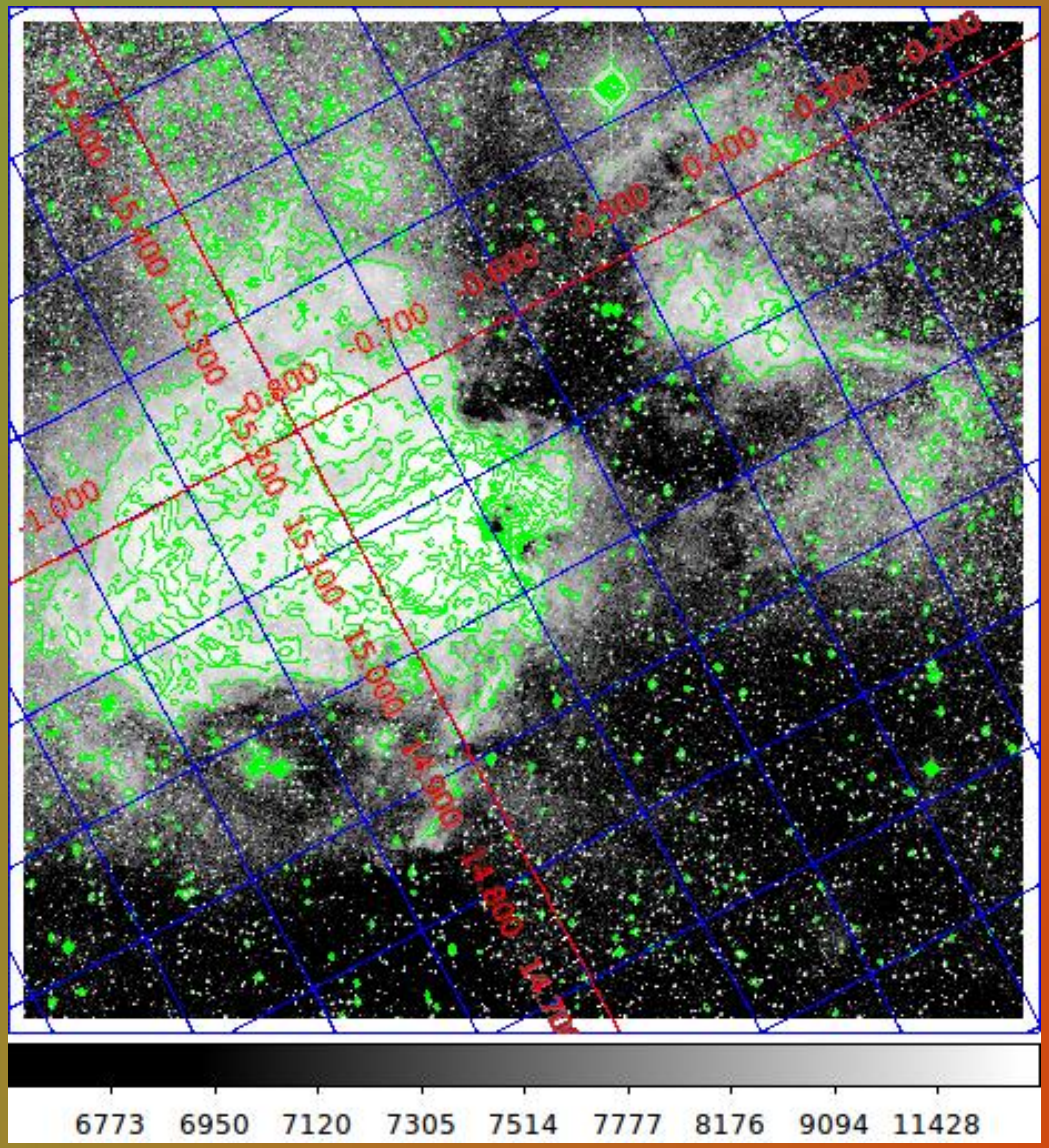
    The contours are calculated from the data intensities.

    You can copy and paste contours from one image into another (for example, same object but different bands).

  - *Coordinate Grid*: Draws a coordinate grid over your image.

    Options include different coordinate systems. The image must have a WCS, obviously.

Other useful menus



Our M17 image with:

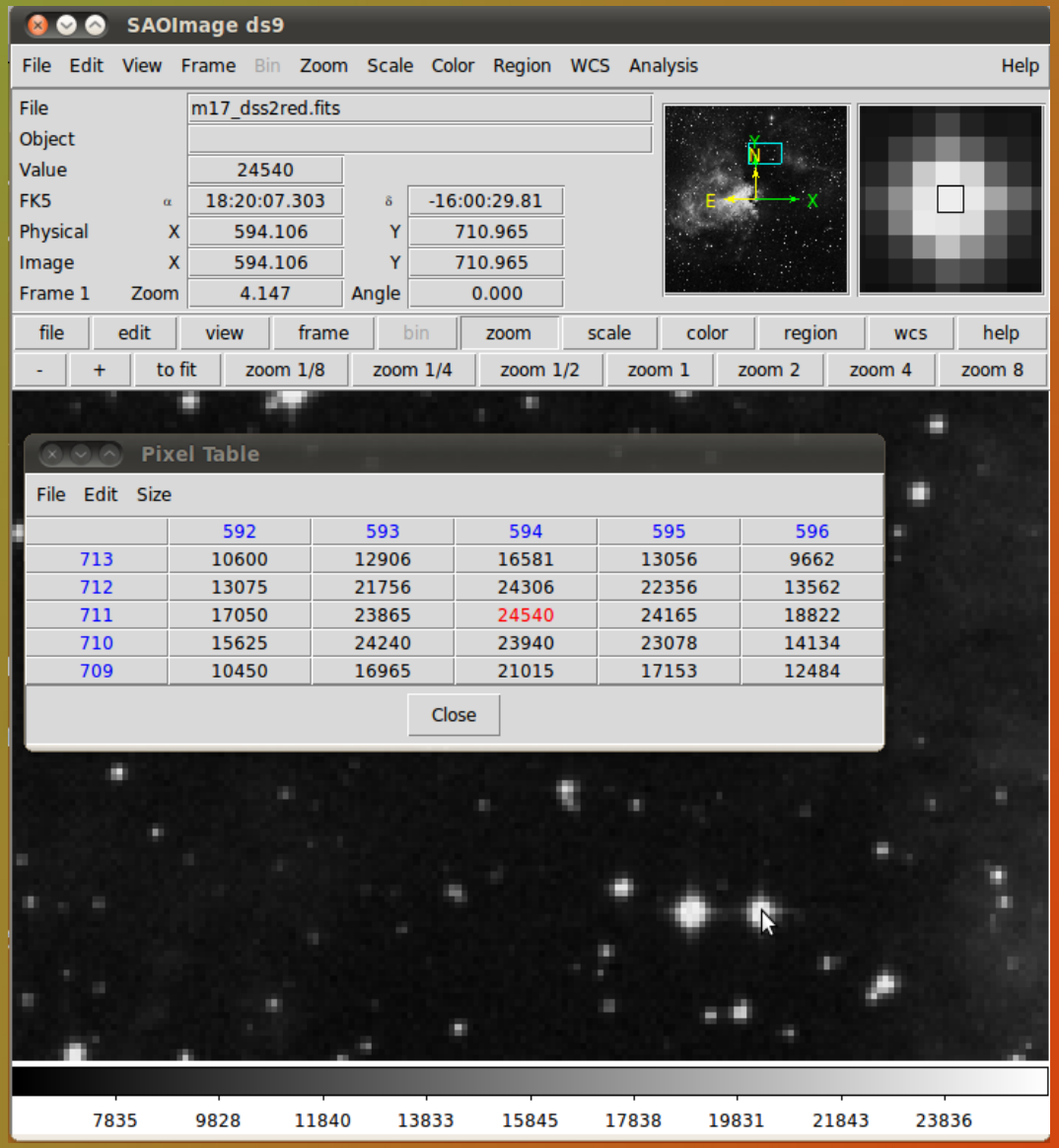
- Contours (green)

- Galactic coordinates grid (red and blue)

Other useful menus

- Furthermore, the *Analysis* menu includes the *Pixel Table* option.
- It displays the pixel intensities around the cursor in a spreadsheet-like format that opens in a new window.
- You can inspect pixel intensities quickly.
- This feature will be useful for your next HW assignment!

Other useful menus



A *Pixel Table*

Note the position of the cursor.

Note the *Magnifier*.

Note the numbers in blue, these are the pixel row/column values.

Note the value in red in the Pixel Table, this is where the cursor is.

The numbers in the table are *pixel intensities*.

Creating an RGB image

- As a final exercise, you will create an RGB (Red-Green-Blue) image using DS9.

  - Here you will apply what we have learned about the TF and scale limits in DS9.

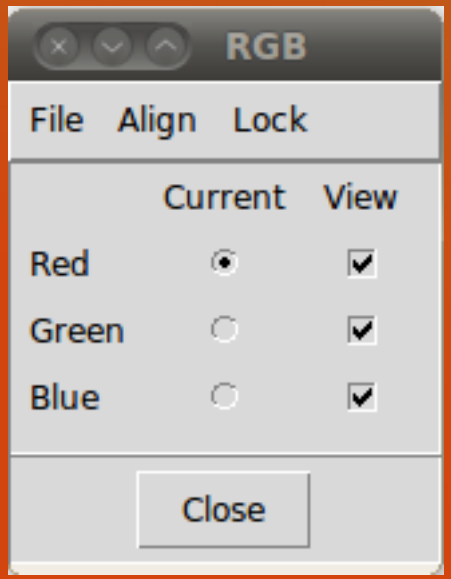
  - We will use DSS2 images in Blue, Red and IR bands.

- Try:
  - Go to *Frame* and click on *New RGB*. A floating window named *RGB* will appear.

  - Choose a color in this window, then go to *File* on DS9,

  - click on *Open,* and load the corresponding image for the color.

- Repeat the process for the other two colors/images.



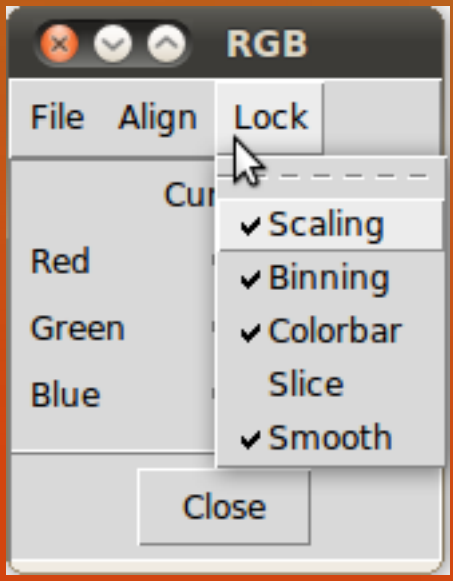
RGB window

Creating an RGB image

- If the images have different binning, scaling, smoothing and colorbar, you can lock them and adjust the scale for the three images together.

In *RGB*, click on *Lock* and tick these four options.

- Or if you prefer to adjust the images separately, untick the *Lock* options and tick/untick the *View* and *Current* buttons in *RGB*.
- Use a convenient scale from the *Scale* menu and convenient limits, and adjust it with the right button.
- If you are adjusting one image at a time, be aware that the changes are made to the image marked as *current* in the *RGB* window, not the ones in *View*.

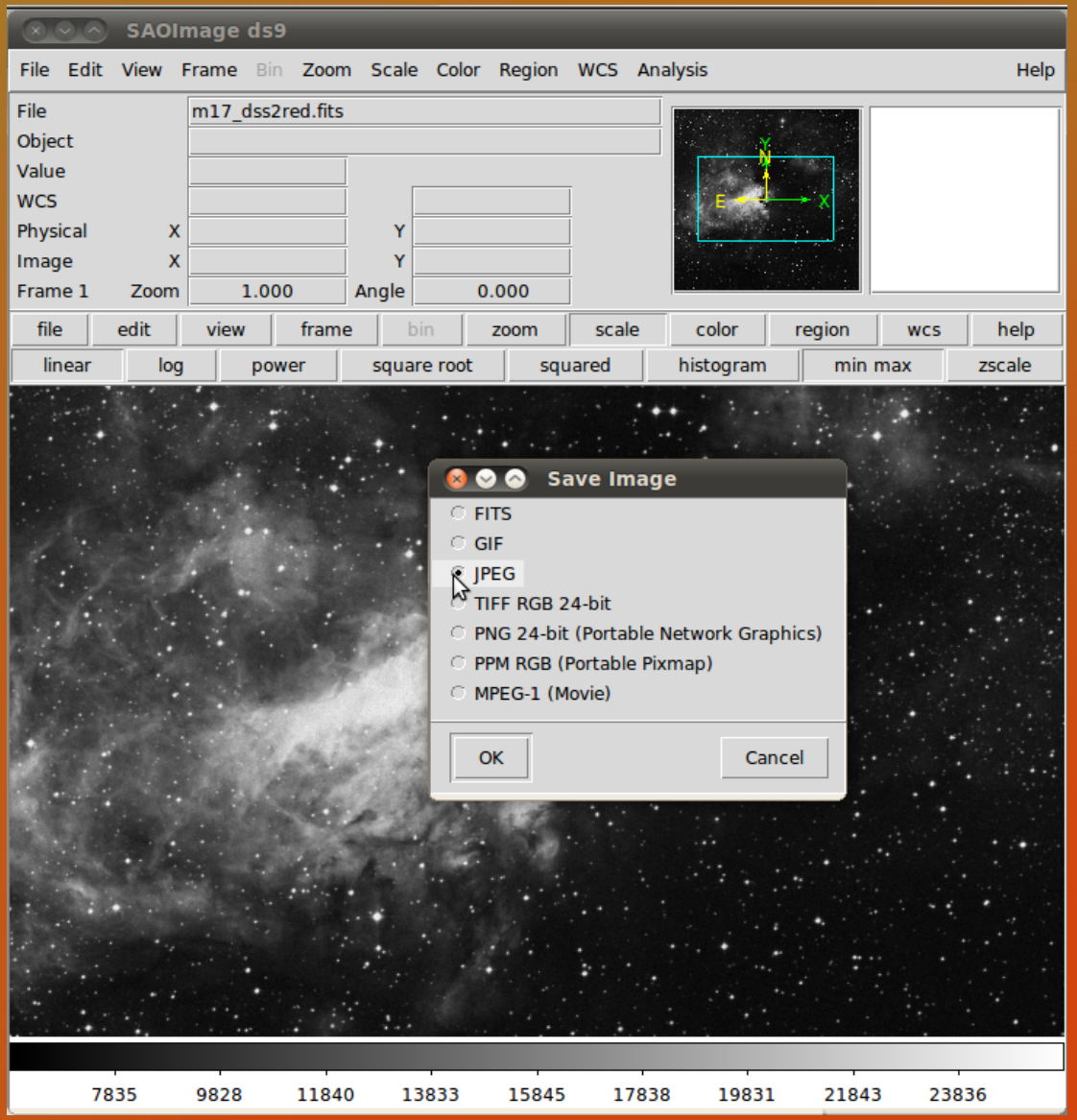


Lock menu

Saving/printing your RGB image

- You can **save** your image as a JPEG in the *File* menu and *Save image* when you are happy with the results.

  (The TA's prefered option for printing is from GIMP).

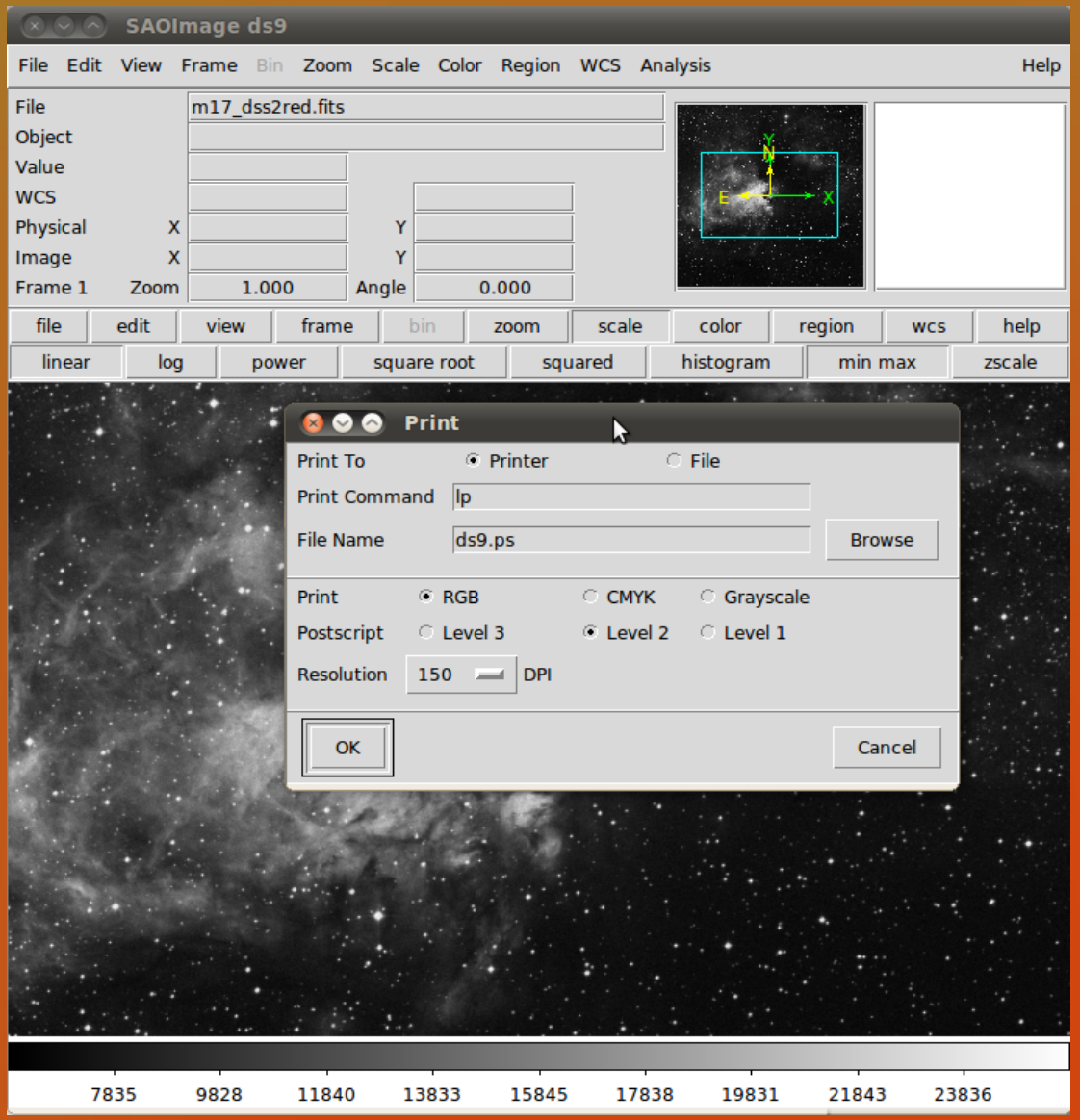


Different image formats are available for saving an image.

Saving/printing your RGB image

- Or you can **print** your image from the *File* menu and *Print*.
- You may choose *printer* or *file*.

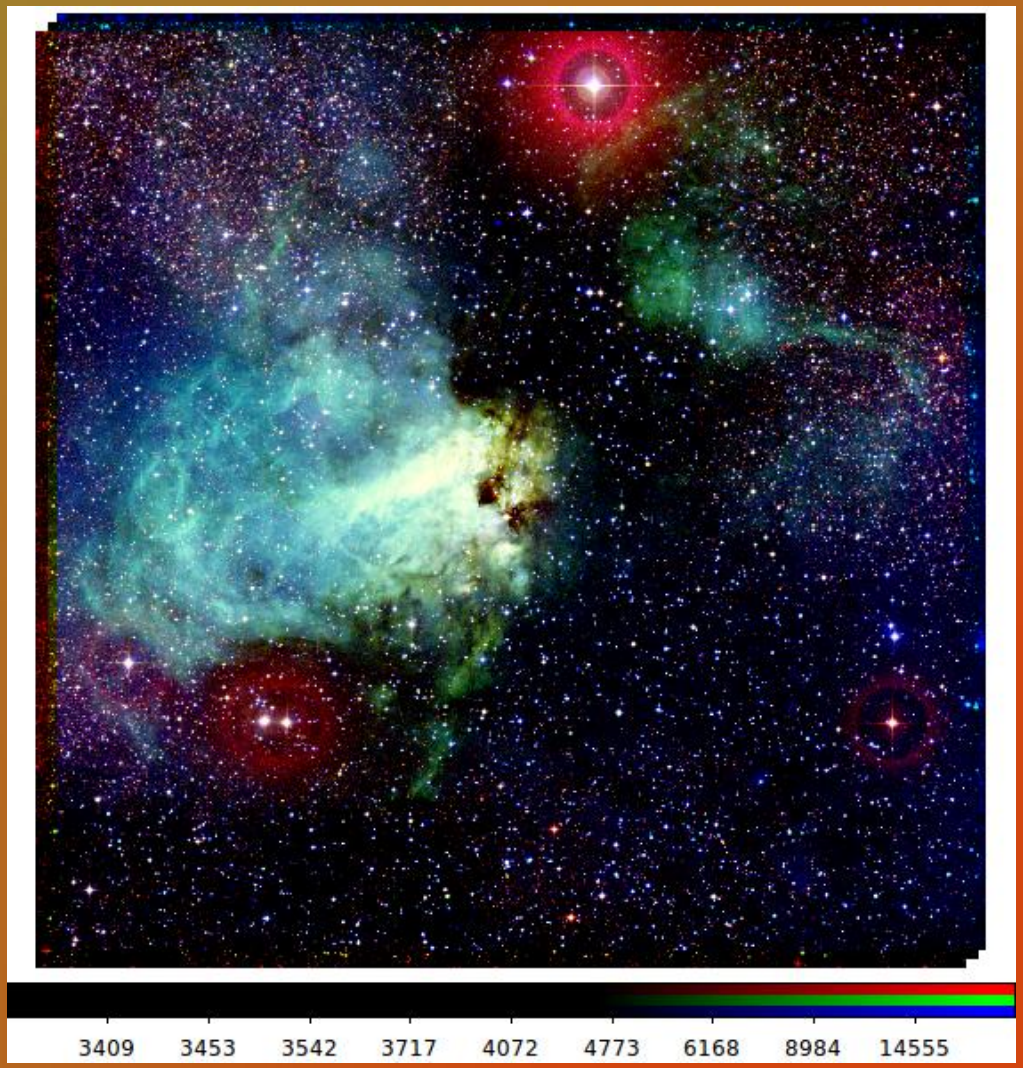
  The later option will create a .ps document.



The *Print* window

Saving/printing your RGB image

- Enjoy!



Example RGB of M17, made by your TA

References

- Part I:
  - FITS documentation:

    http://fits.gsfc.nasa.gov/fits_documentation.html
  - A Primer on the FITS Data Format:

    http://fits.gsfc.nasa.gov/fits_primer.html
  - Definition of the Flexible Image Transport System (FITS) :

    http://archive.stsci.edu/fits/fits_standard/
  - WCS Tools:

    http://tdc-www.harvard.edu/wcstools/
  - FITS files in Python. Demitri Muna - SciCoder's workshop 2010.
- Part II:
  - SAOImage DS9 Reference Manual:

    http://hea-www.harvard.edu/RD/ds9/ref/index.html
  - SAOImage DS9 Users Manual:

    http://hea-www.harvard.edu/RD/ds9/user/index.html
- FITS images from NASA's Skyview Virtual Observatory. Images from DSS1 and DSS2.