Taller de Técnicas Observacionales CASLEO - Agosto de 2016

Procesamiento básico de imágenes con IRAF

Ricardo Gil-Hutton

1 El programa IRAF

El programa IRAF (*Imagen Reduction and Analysis Facility*) es un programa de procesamiento general de imágenes especialmente diseñado para la reducción y análisis de imágenes astronómicas. IRAF fue creado por el NOAO (National Optical Astronomical Observatories, EEUU) y la última versión disponible es la 2.16 de Marzo de 2012.

IRAF se organiza en **paquetes** y **tareas** agrupadas por sus funciones y objetivos, pero es posible realizar trabajos de procesamiento muy específicos debido a la posibilidad de ejecutar tareas bastante complejas mediante **scripts** creados por el usuario.

Si bien IRAF cuenta con su propio programa para visualizar imágenes, denominado **ximtool**, es usual utilizar un programa externo denominado **ds9** creado en el SAO (Smithsonian Astrophysical Observatory, EEUU). Si bien las nuevas versiones del ds9 presentan inconvenientes para operar con IRAF, la versión 6.2 o anterior funcionan perfectamente y cubren todas las necesidades de visualización necesarias.

1.1 Configurando IRAF en el archivo LOGIN.CL:

Es de suponer que ya cuentan con IRAF y ds9 instalados en sus máquinas pero es importante comprobar que el proceso de instalación fue exitoso y van a poder trabajar sin problemas.

En el directorio raiz de su *home* deben tener un archivo llamado *login.cl* que es el responsable de indicarle al IRAF como ejecutarse, que paquetes cargar al inicio, etc. Al principio del archivo tienen:

```
# LOGIN.CL -- User login file for the IRAF command language.

# Identify login.cl version (checked in images.cl).
if (defpar ("logver"))
    logver = "IRAF V2.16.1 Oct 2013"

set home = "/home/rgh/"
set imdir = "/iraf/imdirs/rgh/"
set cache = "/iraf/cache/rgh/"
set uparm = "home$uparm/"
set userid = "rgh"
```

```
# Set the terminal type.  We assume the user has defined this correctly
# when issuing the MKIRAF and no longer key off the unix TERM to set a
# default.
if (access (".hushiraf") == no)
    print "setting terminal type to 'xgterm' ..."
stty xgterm
```

donde se indican los subdirectorios de interés para IRAF, la identificación del usuario (**set userid = "rgh"**) y el tipo de terminal donde se va a ejecutar IRAF (**stty xgterm**). Tamto los subdirectorios como el tipo de terminal se pueden modificar ejecutando nuevamente el programa de configuración inicial **mkiraf**.

A continuación en *login.cl* se lista un conjunto de parámetros con sus valores iniciales que son necesarios para correr IRAF:

```
#==============================================================================
# Uncomment and edit to change the defaults.
#set editor = vi
#set printer = lp
#set pspage = "letter"
#set stdimage = imt800
#set stdimcur = stdimage
#set stdplot = lw
#set clobber = no
#set imclobber = no
#set filewait = yes
#set cmbuflen = 512000
#set min_lenuserarea = 64000
#set imtype = "imh"
#set imextn = "oif:imh fxf:fits,fit fxb:fxb plf:pl qpf:qp stf:hhh,??h"


# XIMTOOL/DISPLAY stuff.  Set node to the name of your workstation to
# enable remote image display.  The trailing "!" is required.
#set node = "Exidor!"

# CL parameters you mighth want to change.
#ehinit   = "nostandout eol noverify"
#epinit   = "standout showall"
showtype = yes
```

En el listado se incluye el programa de edición de texto que se usará, el tamaño máximo de la imagen que se podrá desplegar, el tamaño del buffer de memoria, etc. Noten que todas las líneas están comentadas (con un "#" adelante) asi que lo que se indica es **el valor del parámetro que se toma por default**. Si desean modificarlos deben remover el "#" inicial y modificar el valor del parámetro según sus necesidades. Dos valores que es deseable modificar son:

1. **stdimage**: este parámetro indica el tamaño máximo de la imagen a desplegar en el ds9 y se encuentra inicialmente seteado para imágenes de $800x800$ pixels. La codificación de los formatos

posibles está en el archivo */iraf/iraf/dev/imtoolrc* pero es conveniente cambiarlo a *imt2048* para considerar imágenes de hasta $2048x2048$ pixels.

2. **imextn**: este parámetro lista las extensiones que IRAF considerará como imágenes e incluye algunos formatos viejos y otros originales de IRAF. Particularmente, las extensiones que considerará para el formato FITS son *...fxf:fits,fit ...* asi que convendría agregar a esa lista la extensión *fts* que es usada a veces.

Un poco más adelante el archivo *login.cl* lista los paquetes de IRAF que se cargan en el inicio y el *mensaje del día*:

```
#==============================================================================
# Load the default CL package.  Doing so here allows us to override package
# paths and load personalized packages from our loginuser.cl.
clpackage


# List any packages you want loaded at login time, ONE PER LINE.
images          # general image operators
plot            # graphics tasks
dataio          # data conversions, import export
lists           # list processing

# The if(deftask...) is needed for V2.9 compatibility.
if (deftask ("proto"))
    proto       # prototype or ad hoc tasks

tv              # image display
utilities       # miscellaneous utilities
noao            # optical astronomy packages
vo              # Virtual Observatory tools

prcache directory
cache   directory page type help

# Print the message of the day.
if (access (".hushiraf"))
    menus = no
else {
    clear; type hlib$motd
}
```

Es posible cambiar el texto del "mensaje del día" modificando el archivo */iraf/iraf/unix/hlib/motd*, agregar a la lista nuevos paquetes que se cargarán al inicio del programa o eliminar algunos otros que resulten innecesarios simplemente comentando la línea con un "#" al principio.

Para ejecutar IRAF se inicia una ventana **xgterm** y se hace correr IRAF con el comando "*cl*" o "*ecl*", dependiendo de la versión instalada. En la figura 1 se muestra la pantalla de inicio del IRAF.

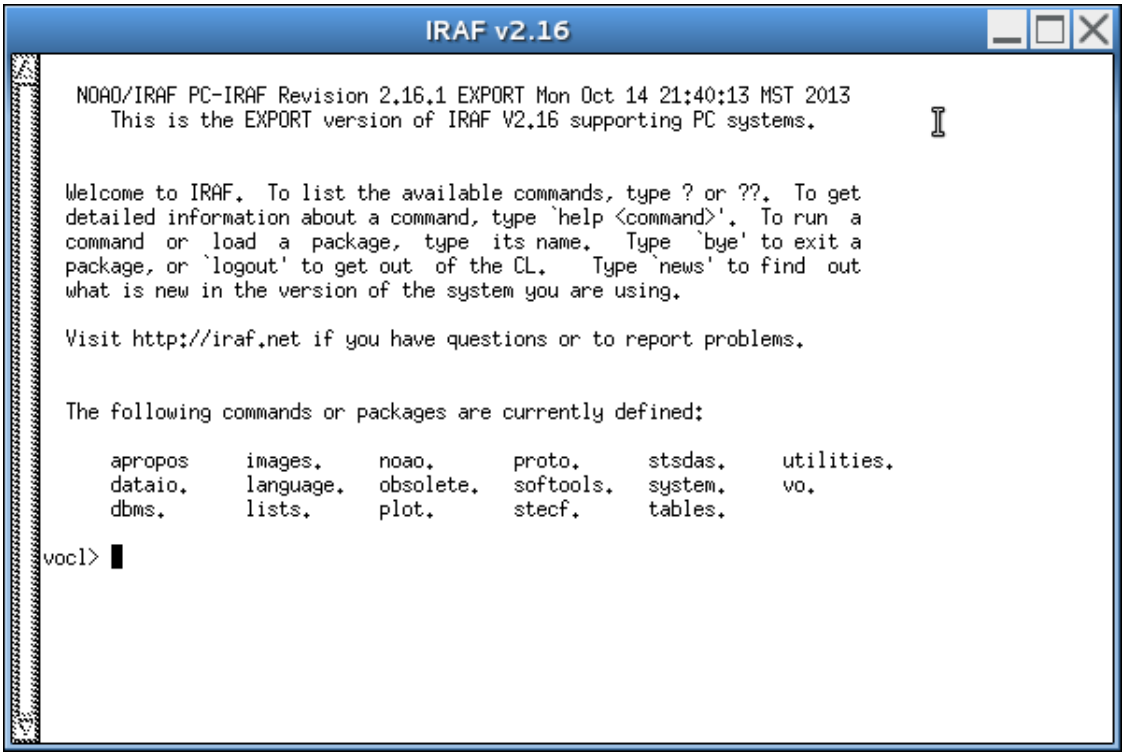

Figure 1: Pantalla de inicio de IRAF.

1.2 Paquetes, tareas y sus parámetros:

Al comenzar IRAF muestra una pantalla de inicio donde, luego de un mensaje inicial, se listan los paquetes disponibles para cargar. El prompt que se muestra (*vocl*>) siempre indicará **el último paquete cargado**. Para salir de IRAF es suficiente escribir LOGOUT o directamente LO.

Para cargar un paquete de tareas se debe ingresar su nombre seguido de ENTER. No es necesario ingresar el nombre completo y si se ingresa un número suficiente de letras para evitar ambigüedades IRAF reconoce el comando sin problemas. Por ejemplo, para cargar el paquete NOAO se ingresa:

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

noao>
```

Luego de cargar el paquete automáticamente se lista por pantalla los paquetes y tareas que integran el paquete cargado (en este caso,NOAO). En cualquier listado por pantalla todos los paquetes se identifican con un punto al final de su nombre. Observen que también cambió el prompt para indicar el último paquete cargado. Cargar un paquete no anula los paquetes cargados anteriormente y si se quiere consultar qué paquetes están cargados se debe ejecutar el comando PACKAGE. Si se quiere saber que paquetes y tareas hay en el último paquete cargado se debe ingresar "**?**" o si se quiere consultar algún otro paquete ya cargado se debe ingresar "**?**< *paquete* >". Ingresando "**??**" se listan todos los paquetes cargados y sus respectivas tareas. Para descargar el último paquete cargado se debe ejecutar el comando BYE:

```
noao> plot
      calcomp     gkidecode   graph       nsppkern    pradprof    sgidecode   stdplot
      contour     gkidir      hafton      pcol        prow        sgikern     surface
      crtpict     gkiextract  imdkern     pcols       prows       showcap     velvect
      gdevices    gkimosaic   implot      phistogram  pvector     stdgraph

plot> bye
      artdata.      astutil.      imred.        nproto.       onedspec.     twodspec.
      astcat.       digiphot.     mtlocal.      observatory   rv.
      astrometry.   focas.        nobsolete.    obsutil.      surfphot.

noao>
```

Todos los paquetes y tareas tienen una ayuda que se obtiene con la tarea HELP. Por ejemplo, si se quiere ver la ayuda del último paquete cargado se puede ingresar HELP sin parámetros:

```
noao> help

rootnoao.noao:
        artdata - Artificial data generation package                    [up]
     astrometry - Astrometry package
         astcat - Astronomical catalog and surveys access package       [up]
        astutil - Astronomical utilities package                        [up]
       digiphot - Digital stellar photometry package                    [up]
          focas - Faint object classification and analysis package
          imred - Image reductions package                              [up]
        mtlocal - Magtape i/o for special NOAO format tapes             [up]
      nobsolete - Obsolete tasks to be phased out in a future release   [up]
         nproto - Prototype (temporary, contributed) tasks              [up]
    observatory - Examine and define observatory parameters             [up]
        obsutil - Observing utilities (planning or evaluation)          [up]
       onedspec - One dimensional spectral red & analysis package       [up]
             rv - Radial velocity analysis package                      [up]
       surfphot - Galaxy isophotal analysis package
       twodspec - Two dimensional spectral red & analysis package       [up]
noao>
```

En la ayuda del paquete se lista el conjunto de paquetes y tareas que lo integran con una pequeña descripción de para qué se utiliza. Si el paquete no fue el último cargado es necesario indicar qué paquete interesa:

```
noao> help imred

noao.imred:
          argus - CTIO ARGUS reduction package
           bias - General bias subtraction tools
         crutil - Tasks for detecting and removing cosmic rays
         ccdred - Generic CCD reductions
       ctioslit - CTIO spectrophotometric reduction package
           dtoi - Density to Intensity reductions for photographic plates
```

```
        echelle - Echelle spectral reductions (slit and FOE)
        generic - Generic image reductions tools
          hydra - KPNO HYDRA (and NESSIE) reduction package
           iids - KPNO IIDS spectral reductions
          irred - KPNO IR camera reductions
            irs - KPNO IRS spectral reductions
      kpnocoude - KPNO coude reduction package (slit and 3 fiber)
       kpnoslit - KPNO low/moderate dispersion slits (Goldcam, RCspec, Whitecam)
        quadred - CCD reductions for QUAD amplifier data
        specred - Generic slit and fiber spectral reduction package
           vtel - NSO Solar vacuum telescope image reductions
noao>
```

Por otra parte, si se quiere consultar la ayuda de una tarea, por ejemplo IMSTATISTICS:

```
noao> help imstatistics
...

IMSTATISTICS (Feb01)          images.imutil         IMSTATISTICS (Feb01)



NAME
    imstatistics -- compute and print image pixel statistics


USAGE
    imstatistics images


PARAMETERS

    images
        The input images or image sections for  which  pixel  statistics
        are to be computed.

    fields = "image,npix,mean,stddev,min,max"
        The statistical quantities to be computed and printed.

    lower = INDEF
        The  minimum  good data limit.  All pixels are above the default
        value of INDEF.

    upper = INDEF
        The maximum good data limit.  All pixels are above  the  default
        value of INDEF.

    nclip = 0
        The  maximum  number of iterative clipping cycles. By default no
        clipping is performed.
```

```
    lsigma = 3.0
        The low side clipping factor in sigma.

    usigma = 3.0
        The high side clipping factor in sigma.

    binwidth = 0.1
        The width of the histogram bins used for computing the  midpoint
        (estimate of the median) and the mode.  The units are in sigma.

    format = yes
        Label  the  output columns and print the result in fixed format.
        If format is "no" no column labels are printed  and  the  output
        is in free format.
...
...
```

En el caso de una tarea la ayuda indica en qué paquete está la tarea, para qué sirve, como se ejecuta, se listan los parámetros necesarios, se describe el proceso que realiza, se dan ejemplos y se listan tareas que puedan tener relación o una función similar. Todas las tareas estan controladas por un listado de **parámetros** que indican sobre qué imagen actuará, como va a atender diversas situaciones de procesamiento, que procedimiento utilizará para realizar su función, si guardará o no la información resultante, si realizará un proceso interactivo, etc. Más alla de la descripción de cada parámetro que hay en la ayuda, el listado de los parámetros válidos para una tarea se pueden obtener ejecutando LPARAM. Por ejemplo:

```
noao> lparam imstatistics
       images =                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")
noao>
```

Para llamar a una tarea es necesario definir qué valor tendrá cada parametro, más allá de algunos de ellos que ya tienen valores por default (por ejemplo, en la tarea IMSTATISTICS el parámetro *lsigma* tiene inicialmente el valor "*3*"). Entonces, si se quiere ejecutar IMSTATISTICS y queremos modificar algunos parámetros debemos editar el **archivo de parámetros de la tarea** con la tarea EPARAM:

```
noao> eparam imstatistics
...
```

```
                                    I R A F
                          Image Reduction and Analysis Facility
PACKAGE = imutil
   TASK = imstatistics

images  =                       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
(binwidt=                   0.1) Bin width of histogram in sigma
(format =                   yes) Format output and print column labels ?
(cache  =                    no) Cache image in memory ?
(mode   =                    ql)
```

donde se pueden modificar los valores de los parámetros para finalmente grabarlos con el comando "*:wq*" (que es un comando del editor *vi*. Si se quiere salir sin guardar hay que presionar el comando "*:q*"). Si se conoce el nombre del parámetro que se quiere modificar es posible cambiarlo directamente por la línea de comandos indicando el parámetro a cambiar y la tarea a la que pertenece. Por ejemplo:

```
noao> imstatistics.lsigma=5.
noao> lparam imstatistics
       images =                 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 = 5.)            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")
noao>
```

Los archivos de parámetro se guardan en un subdirectorio llamado "*uparm*" que se define en *login.cl* tal como vimos antes, y contienen el último conjunto de parámetros que definió el usuario. Esta característica hace necesario tener mucho cuidado porque si en una sesión de IRAF se modifica el archivo de parámetros de una tarea, **estos parámetros serán válidos en sesiones sucesivas por más que se salga de IRAF y se vuelva a ingresar**. Si se quiere restaurar el archivo de parámetros a sus valores originales, se debe utilizar la tarea UNLEARN seguida del nombre de la tarea. Por ejemplo:

```
noao> unlearn imstatistics
noao> lparam imstatistics
```

```
       images =                 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")
noao>
```

Si solo se ingresa UNLEARN **se restaurarán los archivos de parámetros de todas las tareas**.

Una opción para definir los parámetros es no modificar directamente el archivo de parámetros de la tarea y ejecutarla indicando explícitamente los parametros a modificar y sus valores. Por ejemplo, si quiero ejecutar la tarea IMSTATISTICS sobre la imagen "xxx.fits", que me muestre el nombre de la imagen y los valores medio, maximo y minimo, que guarde todo en un log del proceso en el archivo "xxx.log", y que no le de formato a la salida de información, debo ingresar:

```
noao> imstatistics xxx.fits fields="image,mean,max,min" format=no
...
...
```

Noten que en el listado que muestra LPARAM hay algunos parámetros que están entre paréntesis y otros no. Los primeros se denominan **parámetros ocultos** que no son obligatorios y para indicarlos en la línea de comandos se debe especificar el nombre del parámetro seguido de un signo "=" y de su valor; mientras que los segundos se denominan **parámetros posicionales** que si son obligatorios y si se los incluye en la línea de comandos deben ingresarse en el orden en el que aparecen en el listado que da LPAR. Entre los parámetros ocultos también se incluyen los **flags** que son aquellos parámetros que aceptan como valor sólo los valores "*yes*" o "*no*" exclusivamente (también puede usarse "*+*" o "*-*" sin el signo "=", por ejemplo: *format+*).

1.3 Probando si la instalación es correcta:

Para probar si la instalación de IRAF es correcta y se dispone de las prestaciones mínimas, se puede realizar el siguiente proceso:

1. bajar de la página del taller la imagen de prueba "xxx.fits" (en el archivo "*iraf-basico.tgz*").
2. iniciar el programa IRAF con "*ecl*" (ver figura 1) y cambiar al subdirectorio donde guardan la imagen de prueba:

   ```
   vocl> cd /home/rgh/taller/
   vocl>
   ```

3. inicien el ds9.

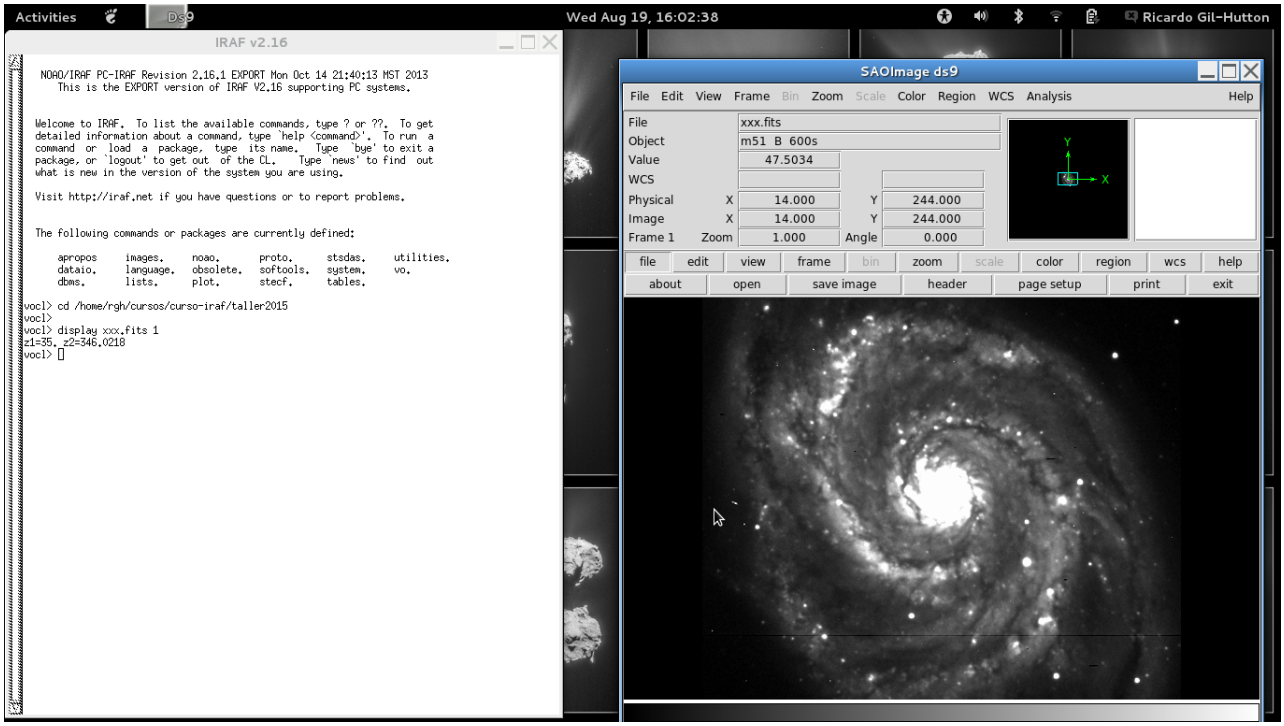

Figure 2: Desplegando la imagen de prueba en ds9 desde IRAF.

4. en la pantalla de IRAF ejecuten (ver figura 2):

```
vocl> display xxx.fits 1
z1=35. z2=346.0218
vocl>
```

que muestra la imagen en el *ds9*.

5. en la pantalla de IRAF ejecuten la tarea:

```
vocl> imexamine xxx.fits 1
...
```

y el cursos va a saltar a la imagen en el ds9 y permanecerá parpadeando. Sin tocar el cursor presionen la tecla "**c**" y les deberá aparecer una nueva ventana con el corte de la imagen a través de la columna en la que estaba posicionado el cursor (figura 3). Con el cursor sobre la ventana del ds9 presionen la tecla "**q**" para salir de IMEXAMINE.

Si todo funcionó correctamente quiere decir que IRAF esta bien instalado y que se cuenta con las prestaciones mínimas para desarrollar el curso.

2 Cuestiones adicionales y algunas tareas básicas

En esta sección vamos a describir algunas cuestiones adicionales pero que son de importancia para usar IRAF, a enumerar y describir brevemente algunas tareas de IRAF que son de mucha utilidad la gran mayoría de las veces, y comentar también ciertos procedimientos y características de IRAF que es relevante conocer.

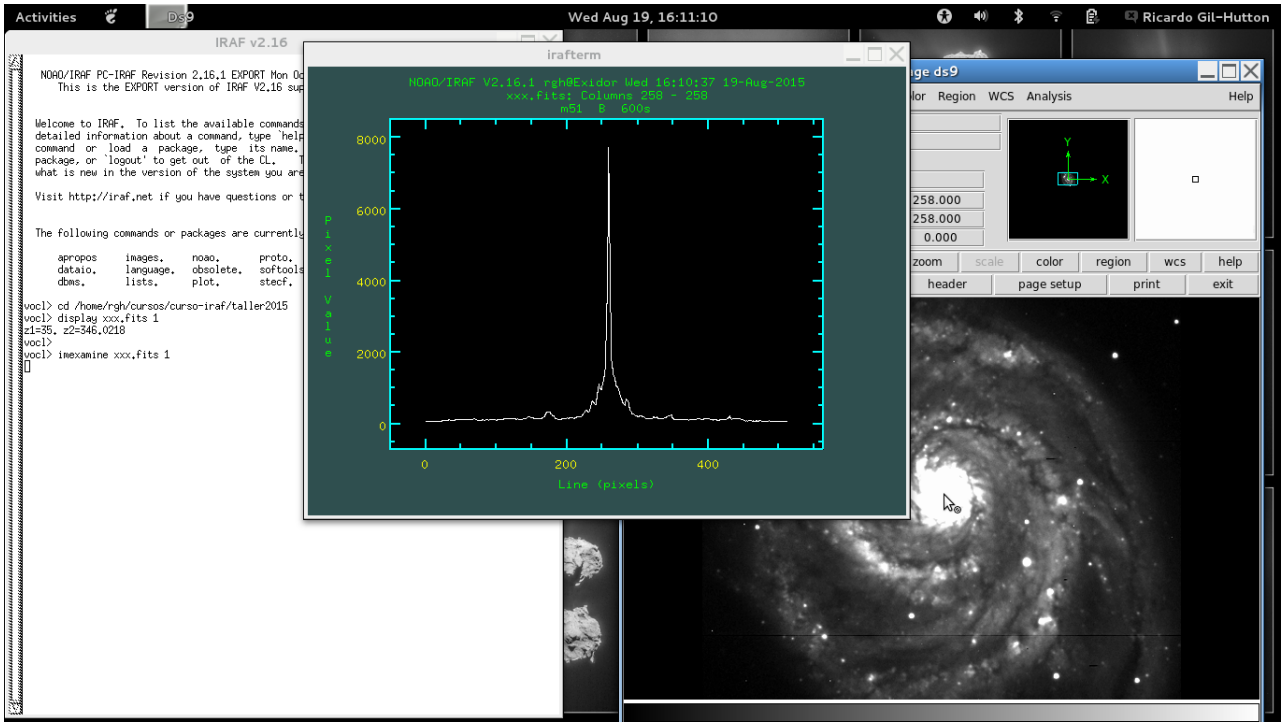

Figure 3: Corte sobre la imagen de prueba con IMEXAMINE.

2.1 El formato FITS:

Si bien IRAF permite operar con imágenes y tablas en diferentes formatos, vamos a asumir en este taller que se trabajará sólo con imágenes en formato **FITS** (*Flexible Image Transport System*) con extensiones *.fits*, *.fit* o *.fts*. El formato FITS fue desarrollado a fines de los años 70 y se describe en varios trabajos, pero recientemente Pence et al. (2010) presenta una descripción actualizada (está disponible en la página del taller).

Un archivo FITS esta formado por:

- Una **cabecera**, denominada *header*, que contiene uno o más bloques de 36 líneas de 80 caracteres ASCII cada una. Cada bloque tiene 2880 bytes. En la cabecera se guarda la información que permite reconstruir la imagen correctamente.
- Una **matriz de datos** que contiene a la imagen en si y que se denomina *unidad primaria de datos*. Su extensión debe ser un múltiplo de 2880 bytes.
- Otras **matrices de datos** adicionales opcionales, denominadas *extensiones*, que pueden ser otras imágenes o tablas de datos.

Un ejemplo del uso de extensiones es cuando se quiere guardar en un mismo archivo FITS imagenenes de una misma región adquirida en diferentes filtros, en diferentes planos de polarización, etc. Nosotros siempre consideraremos para este taller **imágenes simples** formadas por una cabecera y una unidad de datos que contiene a la imagen que vamos a procesar.

En la cabecera de la imagen se listan **palabras clave** o **keywords** que nos permiten conocer las propiedades y características de la imagen que está en la unidad de datos, con un keyword por línea como máximo. El nombre del keyword en **mayúsculas** ocupa los **ocho primeros caracteres** de la

línea, el noveno es el signo "=", el décimo un **espacio**, y entre el lugar 11 y 80 se incluye el **valor asignado** y un **comentario** que no es obligatorio, ambos separados por el caracter "/". La cabecera mínima posible de cualquier archivo FITS está formada por los **seis keywords obligatorios**:

```
SIMPLE  =                    T / Fits standard
BITPIX  =                   16 / Bits per pixel
NAXIS   =                    2 / Number of axes
NAXIS1  =                  512 / Axis length
NAXIS2  =                  512 / Axis length
...
...
...
END
```

donde:

- **SIMPLE:** debe ser siempre el primero en aparecer. Indica si el archivo se adapta al standard FITS o no. Puede ser cierto ("T") o falso ("F").
- **BITPIX:** indica el tipo de dato que se guarda en la unidad de datos (ver tabla 1).

Table 1: Valores válidos para BITPIX.

valor	datos representados
8	Caracter o byte sin signo
16	Enteros con signo de 2 bytes
32	Enteros con signo de 4 bytes
64	Enteros con signo de 8 bytes
-32	Real en simple precisión
-64	Real en doble precisión

- **NAXIS:** dimensiones de la imagen. Usualmente "2".
- **NAXIS1:** número de pixels de la primera dimensión.
- **NAXIS2:** número de pixels de la segunda dimensión.
- **END:** indica el final de la cabecera.

Entre los keywords *NAXIS2* y *END* pueden aparecer otros keywords que indiquen diferentes parámetros o datos de valor para el observador, como la fecha de la observación, el instrumento utilizado, el nombre del observador, el nombre del objeto, el tiempo de exposición, la ganancia y ruido de lectura del detector, el binning empleado, etc.

Dos keywords que resultan de importancia son **BZERO** y **BSCALE** que permiten ajustar el rango de valores permitidos para cada pixel de la imagen. Como se indicó en la tabla 1, el formato FITS guarda datos en valores **enteros con signo** o **reales**, pero los detectores modernos digitalizan la imagen en valores **enteros sin signo** (usualmente, entre 0 y 65535) y entonces se requiere usar los keywords *BZERO* y *BSCALE* para que la imagen se represente correctamente mediante:

$$valor\ final = BZERO + BSCALE \times valor\ guardado,$$

En el caso mencionado, para recuperar un entero sin signo el valor de *BSCALE* es 1.0 y *BZERO* toma valores de 2^{15} (32768), 2^{31} y 2^{63} para valores de *BITPIX* de 16, 32 o 64, respectivamente.

2.2 Algunas tareas básicas:

Antes de comentar las tareas básicas vamos mencionar la posibilidad que tienen **algunas** tareas de IRAF de procesar **listas** de imágenes en lugar de imágenes individuales. Por ejemplo, si tenemos diez imágenes a las cuales queremos restarle un valor constante, por ejemplo 100, y gardarlas con un nombre diferente al original, en lugar de procesar imagen por imagen podemos utilizar listas para hacer ese trabajo. Para eso:

- se crea un archivo de texto con el nombre de las diez imagenes a procesar y lo nombro, por ejemplo, "*lista-in*".
- se crea un archivo de texto con el nombre que se le quiere dar a cada imagen a procesar y lo nombro, por ejemplo, "*lista-out*".
- para restarle la constante utilizo la tarea IMARITH y, en lugar del nombre de una imagen utilizo el nombre del archivo con la lista de imagenes al que le antepongo el caracter "**@**":

```
vocl> imarith @lista-in - 100 @lista-out
vocl>
```

Otra posibilidad de IRAF es la de trabajar sobre **secciones de imágenes** en lugar de imágenes completas. Para defir una sección de imagen se le agrega a continuación del nombre de la imagen inicial un **delimitador** de la forma $[X_{inicio} : X_{fin}, Y_{inicio} : Y_{fin}]$. Por ejemplo:

- *xxx[100:199,100:199]* es una sección de 100×100 pixels de la imagen "*xxx.fits*".
- *xxx[1:100,*]* son las 100 primeras columnas de la imagen. El símbolo "*" indica todo el rango en la coordenada Y.

Por último, IRAF acepta algunos comandos de Unix directamente, tales como **cat**, **cd**, **ls**, **mkdir**, **pwd** o **cp**, pero puede ejecutar cualquier comando de Unix si se le antepone el símbolo "**!**". Por ejemplo, para borrar el archivo "*xxx.jpg*" desde IRAF:

```
vocl> !rm xxx.jpg
vocl>
```

A continuación vamos a comentar brevemente algunas tareas de IRAF que se consideran fundamentales. Una descripción completa de cada uno de ellas se puede obtener en las respectivas ayudas (*help* < *tarea* >).

- **Copiar, renombrar y borrar una imagen FITS:** Se utilizan las tareas IMCOPY, IMRENAME e IMDELETE, respectivamente:

```
vocl> imcopy xxx.fits zzz.fits
xxx.fits -> zzz.fits
vocl> imrename zzz.fits yyy.fits
vocl> imdelete yyy.fits
vocl>
```

- **Crear listas de imágenes:** Se utiliza la tarea SECTIONS. Por ejemplo para crear un archivo "*lista*" con todas las imagenes FITS en el directorio:

```
vocl> sections *.fits > lista
vocl>
```

Para crear un archivo "*lista*" con todas las imagenes FITS en el directorio pero sólo considerando las sección de imagen $[100:200, 100:200]$:

```
vocl> sections *.fits[100:200,100:200] > lista
vocl>
```

La tarea FILES hace lo mismo y permite utilizar comodines ("*****" y "**?**") en los nombres de las imágenes, pero no trabaja con secciones.

- **Listar los keywords de la cabecera:** Se utiliza la tarea IMHEADER. Si se quiere listar solo el titulo, tipo y dimensiones de la imagen:

```
vocl> imheader xxx.fits
xxx.fits[512,512][short]: m51  B  600s
vocl>
```

Si se quiere listar toda la cabecera:

```
vocl> imheader xxx.fits l+
xxx.fits[512,512][short]: m51  B  600s
No bad pixels, min=0., max=0. (old)
Line storage mode, physdim [512,512], length of user area 2673 s.u.
Created Wed 16:02:17 19-Aug-2015, Last modified Tue 17:35:38 20-Jul-2004
Pixel file "xxx.fits" [ok]
EXTEND  =                    F / File may contain extensions
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 2003' / FITS file originator
DATE    = '2004-07-20T20:35:38' / Date FITS file was generated
IRAF-TLM= '17:35:38 (20/07/2004)' / Time of last modification
OBJECT  = 'm51  B  600s'       / Name of the object observed
IRAF-MAX=           1.993600E4  /  DATA MAX
IRAF-MIN=          -1.000000E0  /  DATA MIN
CCDPICNO=                   53  /  ORIGINAL CCD PICTURE NUMBER
ITIME   =                  600  /  REQUESTED INTEGRATION TIME (SECS)
TTIME   =                  600  /  TOTAL ELAPSED TIME (SECS)
OTIME   =                  600  /  ACTUAL INTEGRATION TIME (SECS)
DATA-TYP= 'OBJECT (0)'          /  OBJECT,DARK,BIAS,ETC.
DATE-OBS= '05/04/87'            /  DATE DD/MM/YY
RA      = '13:29:24.00'         /  RIGHT ASCENSION
DEC     = '47:15:34.00'         /  DECLINATION
...
...
...

vocl>
```

- **Editar los keywords de la cabecera:** Se utiliza HEDIT. Por ejemplo, si se quiere cambiar el valor del keyword *CCDPICNO* de 53 a 63 en la imagen "*zzz.fits*":

```
vocl> hedit zzz.fits ccdpicno 63 add+ up+ ver+
zzz.fits,ccdpicno (53 -> 63):
zzz.fits,ccdpicno: 53 -> 63
update zzz.fits ? (yes): y
zzz.fits updated
vocl>
```

  Lo mismo, pero no quiero verificar cada paso:

```
vocl> hedit zzz.fits ccdpicno 63 add+ up+ ver-
zzz.fits,ccdpicno: 53 -> 63
zzz.fits updated
vocl>
```

  O listar el valor del keyword *RA* de todas las imagenes en la lista "*listado*":

```
vocl> hedit @listado ra .
xxx.fits,RA = 13:29:24.00
yyy.fits,RA = 13:39:04.00
zzz.fits,RA = 13:49:14.00
vocl>
```

- **Operaciones matemáticas simples con imágenes:** La tarea IMARITH permite operar con imágenes de manera simple. La forma de escribir la tarea es *imarith Operador-1 op Operador-2 Resultado*, donde "Operador-1" y "Operador-2" son imagenes, listas o constantes, "op" es una operación matemática (*+, -, *, /, min, max*) y "Resultado" es una imagen o lista donde se guarda el resultado de la operación. Por ejemplo:

```
vocl> imarith xxx.fits * 10. xxx-r.fits
vocl> imarith xxx.fits - zzz.fits final.fits
vocl> imarith @listado-in - 100 @listado-out
vocl>
```

  Una tarea del mismo tipo pero que permite realizar operaciones mucho más complejas en imágenes es IMEXPR, que incluye tomas de decisiones, asignaciones pixel a pixel, etc.

- **Obtener el histograma y algunos valores estadísticos:** Se utilizan las tareas IMHISTOGRAM (ver figura 4) e IMSTATISTICS, que puede dar el número total de pixels considerado, el valor medio, la desviación standard, el máximo y el mínimo de la imagen:

```
vocl> imstatistics @lista fields="image,mean,stddev" format+
#               IMAGE      MEAN    STDDEV
 xxx.fits[100:200,100:200]     108.4     50.82
 zzz.fits[100:200,100:200]     108.4     50.82
```

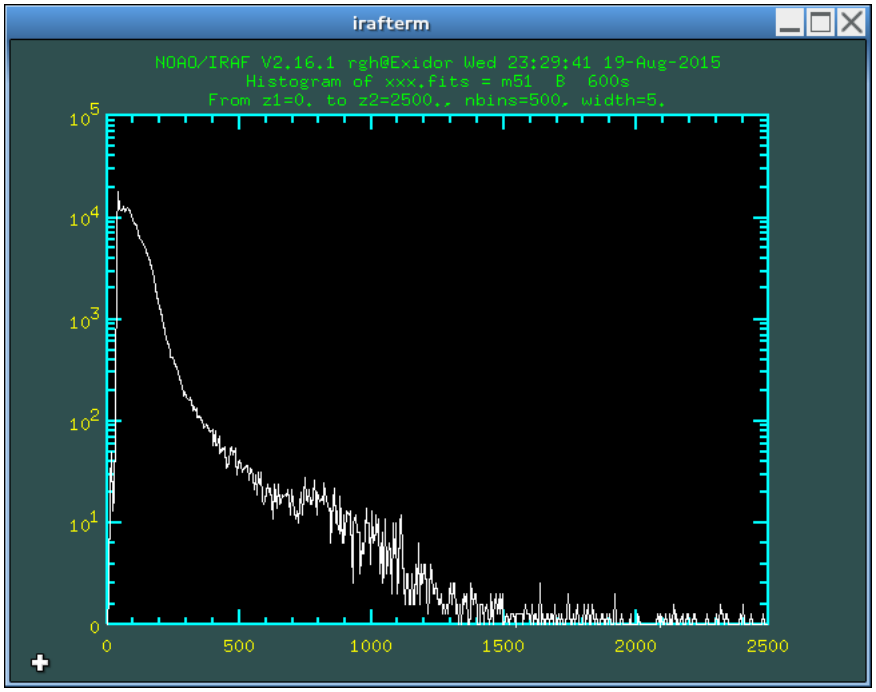

Figure 4: Histograma de "*xxx.fits*" creado por IMHISTOGRAM.

```
vocl>
vocl> imstatistics xxx.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
             xxx.fits    262144     108.3     131.3        -1.     19936.
vocl>
vocl> imhisto xxx.fits z1=0 z2=2500
vocl>
```

- **Cambiar el tipo numérico de una imagen:** Se puede cambiar el tipo numérico de los pixels de una imagen utilizando la tarea CHPIXTYPE que permita cambiar de un tipo numérico a otro. Las opciones para el tipo numérico son "*ushort*", "*short*", "*int*", "*long*", "*real*", "*double*" y "*complex*". Por ejemplo, para convertir la imagen "*xxx.fits*" a real:

```
vocl> chpixtype xxx.fits yyy.fits real
Image: xxx.fits (short) -> Image: yyy.fits (real)
vocl>
```

- **Mostrar una imagen en ds9 desde IRAF:** Se utiliza la tarea DISPLAY indicando la imagen a mostrar y en que marco o frame lo va a hacer. ds9 tiene la posibilidad de mostrar imágenes en 16 frames diferentes lo que posibilita comparar imágenes, entre otras utilidades. Por ejemplo:

```
vocl> display xxx.fits 1
z1=35. z2=346.0218
vocl>
```

nos indica que la imagen se mostrará en el frame "1" y los 256 tonos que puede manejar ds9 para mostrar la imagen se distribuirán de tal modo que todos los pixels con valores ≤ 35 tomarán el tono "0" y aquellos con valores ≥ 346.0218 el tono "255". Este rango de valores se elige automáticamente en base a algunos parámetros estadísticos de la imagen pero es posible fijarlos manualmente. Por ejemplo:

```
vocl> display xxx.fits 1 z1=50. z2=380. zscale- zrange-
z1=50. z2=380
vocl>
```

donde los flags *zscale* y *zrange* puestos al valor "no" indican que la tarea no distribuirá el rango de tonos cerca del valor medio de la imagen y que no desplegara el rango completo de brillo de la imagen.

El ds9 tiene la posibilidad de realizar algunas tareas sobre las imágenes mostradas utilizando los **botones** que se encuentran directamente sobre la imagen. Por ejemplo:

- presionando el botón **color** permite elegir diferentes paletas para mostrar las imágenes.
- presionando el botón **zoom** permite agrandar o achicar la sección de imagen que se muestra.
- presionando el botón **frame** permite moverse entre las imágenes desplegadas en frames diferentes, alternar continuamente entre las imágenes desplegadas para poder compararlas, borrar alguna imagen, o desplegarlas de tal manera que se muestren todas al mismo tiempo.
- presionando el botón **view** permite ocultar o mostrar diferentes ventanas que muestra el ds9 con la barra de tonos, la imagen ampliada o reducida y los gráficos de corte en sentido horizontal y vertical.
- si en cualquier momento se presiona el botón del medio en el mouse, la imágen se centra en el punto que indicaba el cursor. Para volver a centrar la imagen es necesario presionar el botón "zoom" y "to fit".
- si en cualquier momento se mueve el mouse manteniendo presionado el botón derecho un movimiento en sentido vertical cambia el contraste de la imagen y un movimiento en sentido horizontal modifica el punto cero de la escala de tonos.

- **Cómo combinar imágenes:** Una de las tareas más importantes es IMCOMBINE la cual permite combinar imágenes para formar una imagen final obteniendo la suma o promedio y aplicando ciertos procedimientos para descartar pixeles malos. Por ejemplo, si queremos combinar una serie de imagenes bias para mejorar la S/N en una imagen combinada de salida, podemos hacer:

```
vocl> sections bias*.fits > lista
vocl> imcombine @lista zero.fits combine=average reject=minmax nlow=0 nhigh=1

Aug 20 14:46: IMCOMBINE
  combine = average, scale = none, zero = none, weight = none
  reject = minmax, nlow = 0, nhigh = 1
  blank = 0.
                Images
            bias00.fits
            bias01.fits
```

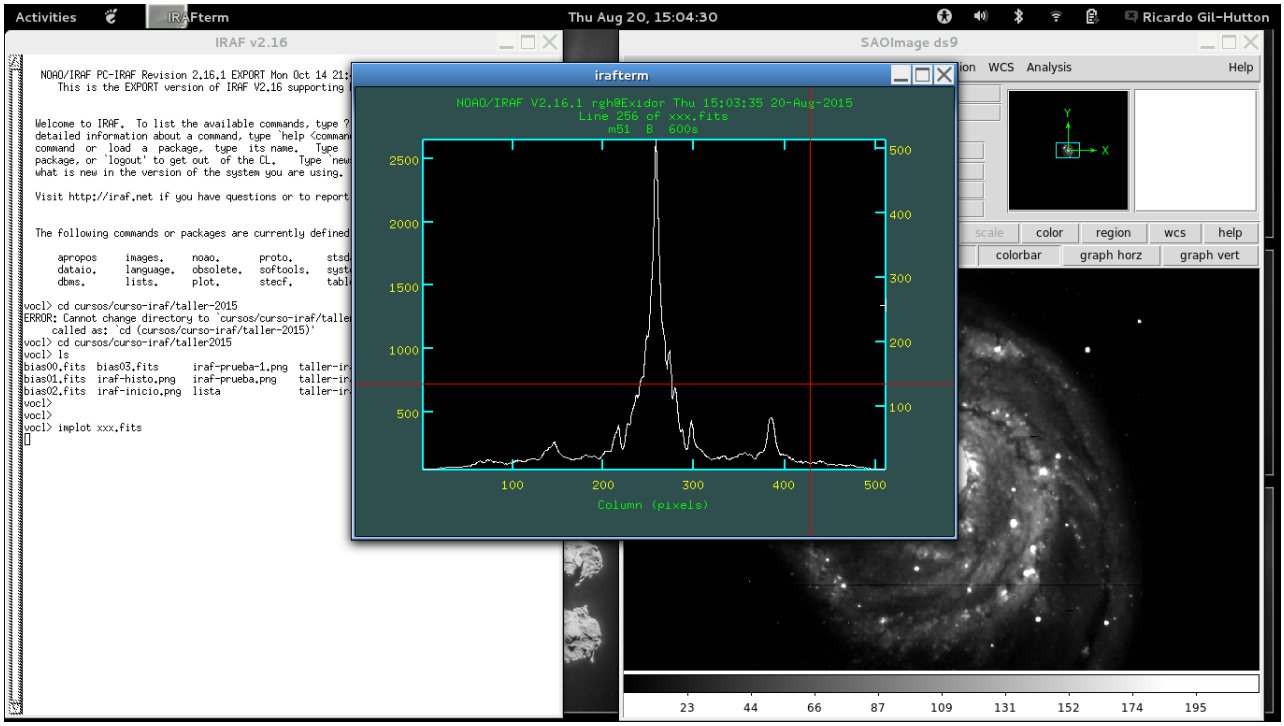

Figure 5: Graficos con la tarea IMPLOT.

```
            bias02.fits
            bias03.fits

  Output image = zero.fits, ncombine = 4
vocl>
```

donde combina una serie de imágenes bias, que se listan en el archivo "*lista*", para formar la imagen final "*zero.fits*", calculando el promedio y utilizando un algoritmo de descarte de pixels denominano *minmax* que obtiene el promedio descartando del cálculo los *nhigh* valores más grandes y lo *nlow* valores más chicos en cada pixel.

La tarea IMCOMBINE puede utilizar diferentes métodos para combinar las imágenes, diferentes modos de descarte de pixels, diversas maneras de asignarle pesos a los datos, utilizar máscaras, etc.

- **Cómo hacer gráficos de una imagen:** Hay diferentes tareas que pueden mostrar gráficos de toda o parte de la imagen pero aqui mencionaremos sólo IMPLOT que permite graficar columnas o líneas de una imagen desplegada en ds9. Para hacer gráficos de la imagen "*xxx.fits*" ejecutamos:

```
vocl> implot xxx.fits
...
```

y luego de iniciar la tarea aparece una ventana gráfica con un corte por la línea central de la imagen y el cursor se convierte a dos líneas rojas que se mueven con el mouse (figura 5). Si se quiere obtener ayuda, con el cursor sobre la ventana gráfica se debe presionar la tecla "**?**" y en

Table 2: Comandos más frecuentes de la tarea IMPLOT.

comando	función
?	imprime la ayuda y otra información
a	marca un rango de líneas o columnas para ser promediadas
c	grafica una columna
e	expande el gráfico marcando las esquinas de la región de interés
l	grafica una línea
m	retrocede a la imagen anterior de la lista de entrada
n	salta a la imagen siguiente de la lista de entrada
o	sobre-escribe el gráfico
p	mide el perfil
q	sale de IMPLOT
r	grafica nuevamente el último gráfico
s	imprime valores estadiśticos de la región
< *space* >	imprime las coordenas y el valor del pixel
:c M [N]	grafica el promedio de las columnas M a N
:l M [N]	grafica el promedio de las columnas M a N

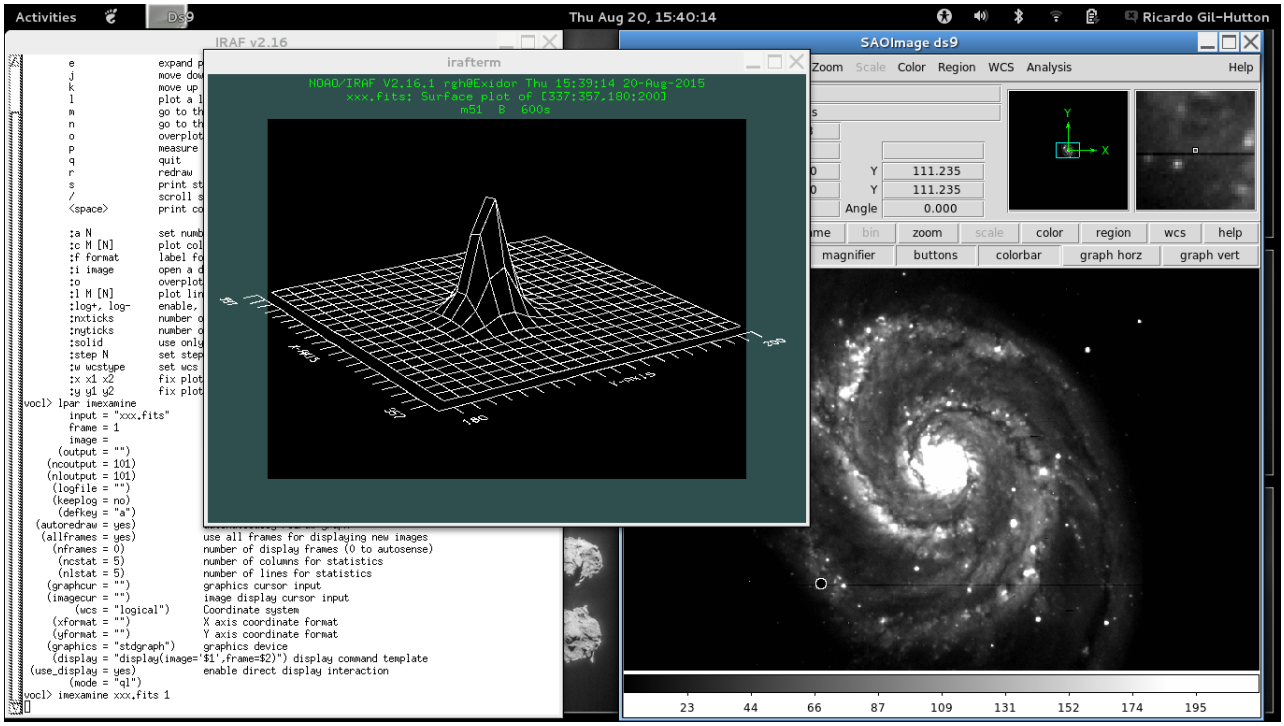

Figure 6: Un gráfico de superficie obtenido al presionar "**s**" en la tarea IMEXAMINE.

Table 3: Comandos más frecuentes de la tarea IMEXAMINE.

comando	función
?	imprime la ayuda y otra información
a	realiza fotometría de apertura aproximada
c	grafica una columna
e	realiza un gráfico de contornos
h	obtiene el histograma
l	grafica una línea
m	obtiene valores estadísticos de la región
q	sale de IMEXAMINE
r	obtiene un gráfico radial
s	realiza un gráfico de superficie
v	grafica un corte entre dos puntos de la imagen
x	da las coordenadas del punto

la ventana de texto del IRAF aparece un listado con los comandos posibles. Los más utilizados se muestran en la tabla 2.

- **Cómo examinar una imagen:** Por último vamos a ver como examinar una imágen para obtener desde gráficos hasta información estadística utilizando la tarea IMEXAMINE. El help de esta tarea es extenso y se encuentran bien documentadas todas sus funciones. Para analizar la imagen "*xxx.fits*" debemos ejecutar:

```
vocl> imexamine xxx.fits 1

...
```

donde el "1" indica el frame donde está desplegada la imagen. Al iniciar la tarea el cursor salta a la imagen desplegada en ds9 y queda parpadeando a la espera de un comando (figura 6). Nuevamente, para obtener ayuda se debe presionar la tecla "**?**" y en la ventana de texto del IRAF aparece un listado con los comandos posibles. Los más utilizados se muestran en la tabla 3.

3 La reducción básica de imágenes

Cuando se adquiere una imagen astronómica se le incorpora involuntariamente una serie de contribuciones producidas por la electrónica del detector, su sistema de enfriamiento, la difracción producida por polvo en alguna superficie óptica, etc. Estas contribuciones se pueden clasificar en **contribuciones aditivas** y **contribuciones multiplicativas**, siendo las primeras una señal que se le agrega a la imagen útil producto de la electrónica del detector y sus variaciones en el tiempo, y las segundas cambios en la sensibilidad pixel a pixel de la imagen por defectos de fabricación o sombras sobre el detector. También debemos considerar que algunos pixeles pueden ser defectuosos y permanecer siempre en el valor máximo posible o en cero, por lo cual sería conveniente identificarlos.

Para remover cada tipo de contribución se debe adquirir una serie de **imágenes auxiliares** que permitirán corregir las imágenes para dejar sólo la información útil y a este proceso se lo denomina **reducción básica** de las imágenes. Algunas de las contribuciones más frecuentes que se pueden encontrar son:

- Contribución aditiva producida por el amplificador que lee la imagen del chip. Se corrigen adquiriendo imágenes con tiempo de exposición cero denominadas **bias**. Como el número de cuentas siempre es bajo en este tipo de imágenes es muy conveniente adquirir un número relevante de ellas para después combinarlas y así mejorar su relación S/N.

- Contribución aditiva producida por la corriente de oscuridad generada por el funcionamiento del detector. Esta contribución depende del tiempo durante el cual el equipo funcionó para adquirir la imagen (o sea, el tiempo de exposición) y es importante en detectores enfriados por efecto Peltier. Se corrige adquiriendo imágenes del mismo tiempo de exposición que las imágenes a corregir pero sin exponer a la luz y que se denominan **darks**. Como el número de cuentas siempre es bajo en este tipo de imágenes es muy conveniente adquirir un número relevante de ellas para después combinarlas y así mejorar su relación S/N. En detectores enfriados por nitrógeno líquido usualmente esta contribución no es significativa.

- Contribución aditiva producida por variaciones electrónicas en el amplificador de lectura del chip. Esta contribución depende del tiempo y puede afectar a cada imágen en forma diferente. Se corrige obligando al amplificador a leer algunas columnas o filas más de las realmente existentes en el chip para poder muestrear la variación temporal de la señal. A esta región adicional se la denomina **overscan** y no todos los equipos permiten leer mas elementos de los que existen físicamente.

- Contribución multiplicativa por diferente eficiencia en cada pixel de la imagen. Esto se produce por la diferente sensibilidad a la luz de cada pixel, sombras sobre el detector producidas por mala iluminación, difracción producida por granos de polvo depositados en alguna superficie óptica, etc. Se corrige adquiriendo imágenes de una pantalla iluminada en forma pareja (o del cielo bien iluminado) adquiridas con muy buena relación S/N y en número suficiente. A estas imágenes se las denomina **flats** y también dependen del filtro utilizado.

Entonces, para proceder a remover estas contribuciones es necesario disponer de una serie de imágenes auxiliares cuyo número y tipo dependerán fuertemente del tipo de observación que se esté realizando. En general, es deseable disponer de un número significativo de bias, darks (si fueran necesarios) y flats para poder aplicar procesos estadísticos y mejorar su relación S/N final.

A continuación se describirá el proceso de reducción básica realizado de dos formas diferentes: una manual y otra semi-automatizada. Si bien la forma manual es algo más trabajosa que la semi-automatizada permite comprender mejor cuales son los procesos que se estan realizando. Para trabajar sobre ambos procedimientos se debe bajar de la página del taller el archivo comprimido "*iraf-basico.tgz*" y descomprimirlo en el subdirectorio donde se quiera trabajar. Al descomprimir el archivo se creará un subdirectorio "*./iraf-basico/imagenes*" en donde se podrá trabajar con los tutoriales que siguen y en los ejercicios.

3.1 Reducción manual:

Luego de descomprimir el archivo con imágenes se inicia IRAF y ds9 de la manera usual y se cambia de directorio al "*iraf-basico*" y se crea un nuevo subdirectorio llamado "*reduccion*" donde se copian las imágenes que tenemos en el subdirectorio "*imagenes*" para trabajar con ellas. Esto se puede hacer desde UNIX o desde IRAF:

```
vocl> cd <tu directorio de trabajo>/iraf-basico
```

```
vocl> mkdir reduccion
vocl> cd reduccion
vocl> imcopy ../imagenes/* .
../imagenes/10p0014.fit -> ./10p0014.fit
../imagenes/10p0019.fit -> ./10p0019.fit
../imagenes/10p0024.fit -> ./10p0024.fit
../imagenes/bias0000.fit -> ./bias0000.fit
../imagenes/bias0001.fit -> ./bias0001.fit
../imagenes/bias0002.fit -> ./bias0002.fit
...
...
...
../imagenes/dfv0006.fit -> ./dfv0006.fit
../imagenes/dfv0007.fit -> ./dfv0007.fit
../imagenes/dfv0008.fit -> ./dfv0008.fit
../imagenes/dfv0009.fit -> ./dfv0009.fit
vocl>
```

A continuación podemos ver cuáles son las dimensiones de las imágenes, el typo de pixel que contienen, etc. con:

```
vocl> imhea *.fit
10p0014.fit[2048,2058][short]: 10p
10p0019.fit[2048,2058][short]: 10p
10p0024.fit[2048,2058][short]: 10p
bias0000.fit[2048,2058][short]:
bias0001.fit[2048,2058][short]:
bias0002.fit[2048,2058][short]:
bias0003.fit[2048,2058][short]:
...
...
...
dfv0007.fit[2048,2058][short]:
dfv0008.fit[2048,2058][short]:
dfv0009.fit[2048,2058][short]:
vocl>
```

que nos dice que tenemos 3 imágenes del cometa *10P/Tempel 2*, 10 bias y 10 flats, todas con dimensiones de $2048x2058$ y de tipo *short* o *entero con signo*. Esto último es un problema porque cualquier valor que sea superior a 32767 ADUs (Unidades Analógicas Digitales) **será representado en la imagen como un valor negativo**. Dado que los detectores digitalizan en un rango de 0 a 65535, es necesario convertir las imágenes a *enteros sin signo* para evitar problemas con el procesamiento. Para eso hacemos una lista con los nombres de todas las imágenes y utilizamos la tarea CHPIXTYPE:

```
vocl> files *..fit > lista
vocl> chpixtype @lista @lista ushort
Image: 10p0014.fit (short) -> Image: 10p0014.fit (ushort)
Image: 10p0019.fit (short) -> Image: 10p0019.fit (ushort)
Image: 10p0024.fit (short) -> Image: 10p0024.fit (ushort)
...
```

```
...
...
vocl>
```

Para asegurarse que durante el procedimiento no se pierda precisión por truncamiento es conveniente utilizar una vez más la tarea CHPIXTYPE para convertir todas las imágenes a *reales*:

```
vocl> chpixtype @lista @lista real
Image: 10p0014.fit (ushort) -> Image: 10p0014.fit (real)
Image: 10p0019.fit (ushort) -> Image: 10p0019.fit (real)
Image: 10p0024.fit (ushort) -> Image: 10p0024.fit (real)
...
...
...
vocl>
```

A continuación conviene revisar la cabecera de alguna de las imágenes para verificar si falta algún keyword que sea necesario durante el proceso de reducción:

```
vocl> imhea 10p0014 l+
10p0014[2048,2058][real]: 10p
No bad pixels, min=0., max=0. (old)
Line storage mode, physdim [2048,2058], length of user area 2673 s.u.
Created Thu 20:25:56 20-Aug-2015, Last modified Thu 20:25:56 20-Aug-2015
Pixel file "10p0014.fit" [ok]
EXTEND  =                    F / File may contain extensions
ORIGIN  = 'NOAO-IRAF FITS Image Kernel July 2003' / FITS file originator
DATE    = '2015-08-20T23:25:56' / Date FITS file was generated
IRAF-TLM= '2015-08-20T23:25:56' / Time of last modification
OBJECT  = '10p     '           / Name of the object observed
CCDSIZE = '[1:2048,1:2058] '     / CCD size
CCDSUM  = '1 1 '                 / CCD binning factors
PIXSIZE1=                 13,5   / pixel size for axis 1 (um)
PIXSIZE2=                 13,5   / pixel size for axis 2 (um)
GAIN    =                    2   / (1) low, (2) medium, (3) high
ADCRATE = ' 100 KHz '            / analog to digital converter rate
CAMTEM  =               -110.0   / camera temperature (degrees Celsius)
EXPTIME =                30.00   / exposure time - seconds
OBSERVAT= 'CASLEO '              / organization or institution
DATE-OBS= '2015-07-18 '          / date of observation (yyyy-mm-dd)
TIME-OBS= '00:47:33.9 '          / time at the start of the observation
UT      = '00:47:33.9 '          / universal time
ST      = '15:52:20.4 '          / sidereal time
HA      = '+01:45:38.5 '         / hour angle
RA      = '14:06:41.8 '          / right ascension
DEC     = '+00:24:43.9 '         / declination
EQUINOX =               2015.5   / epoch of ra y dec
MJD-OBS =         57221.033021   / modified Julian date of the observation
ZD      =                 40.8   / zenith distance
AIRMASS =                 1.32   / airmass
```

```
IMAGETYP= 'object '              / object, dark, zero, etc.
INSTRUME= 'Direct CCD '
OBSERVER= 'R. Gil Hutton'
COMMENT = 'Roper Directo '
COMMENT = 'Filtros B V R I  +   C/R. Focal '
FILTER01= '(5) Libre        '    / Filter wheel No.1 (Lower)
FILTER02= '(3) V            '    / Filter wheel No.2 (Upper)
vocl>
```

Lo que se observa es que el keyword *GAIN* indica un valor de referencia para la ganancia y no el correcto, pero además falta un keyword que indique el valor para el *ruido de lectura*. Si se consulta la página de CASLEO se obtiene que para una tasa de lectura de $100\,khz$ la ganancia del detector para la ganancia media es de $2.18\,e^-/ADU$ y el ruido de lectura de $3.14\,e^-$, entonces sería conveniente agregar los keywords correspondientes para poder utilizar esos valores para poder utilizarlos oportunamente:

```
vocl> hedit @lista gain 2.18 add+ up+ ver-
10p0014.fit,gain: 2 -> 2.18
10p0014.fit updated
10p0019.fit,gain: 2 -> 2.18
10p0019.fit updated
10p0024.fit,gain: 2 -> 2.18
10p0024.fit updated
...
...
vocl> hedit @lista rdnoise 3.14 add+ up+ ver-
add 10p0014.fit,rdnoise = 3.14
10p0014.fit updated
add 10p0019.fit,rdnoise = 3.14
10p0019.fit updated
add 10p0024.fit,rdnoise = 3.14
10p0024.fit updated
...
...
vocl>
```

El sigiente paso en conseguir una imagen bias con la mejor relación S/N posible, para lo cual vamos a combinar los bias usando la tarea IMCOMBINE para crear una imagen final "*zero.fit*" utilizando el algoritmo de rechazo *minmax*, y despreciando el valor más alto de cada pixel (*nhigh=1*, *nlow=0*):

```
vocl> files bias*.fit > lista
vocl> imcombine @lista zero.fit reject=minmax nhigh=1 nlow=0

Aug 20 20:50: IMCOMBINE
combine = average, scale = none, zero = none, weight = none
reject = minmax, nlow = 0, nhigh = 1
blank = 0.
Images
bias0000.fit
bias0001.fit
bias0002.fit
```

```
bias0003.fit
bias0004.fit
bias0005.fit
bias0006.fit
bias0007.fit
bias0008.fit
bias0009.fit

Output image = zero.fit, ncombine = 10
vocl>
```

Para corregir por la contribución del bias simplemente se debe restar éste a las imágenes del objeto y los flats:

```
vocl> files 10p*.fit > lista
vocl> files dfv*.fit >> lista
vocl> imarith @lista - zero.fit @lista
vocl>
```

Las dimensiones de las imágenes de 2048×2058 pixels contienen 10 filas adicionales que no corresponden a una región física del chip sino a la región de *overscan* para corregir por la contribución aditiva que introduce las variaciones del amplificador de lectura del detector. Para obtener la corrección que se necesita vamos a utilizar la tarea LINEBIAS del subpaquete **bias** del paquete **imred** para promediar las 10 líneas de la región de overscan, ajustar a esos datos una función para determinar el error que se introduce en cada columna, restarle esa contribución a la imagen y realizar un recorte para retener la región con información útil. Además de LINEBIAS, en el paquete existe otra tarea denominada COLBIAS que permite hacer el mismo trabajo cuando la región de overscan está en la otra dirección.

Antes de seguir adelante debemos encontrar la región útil de las imágenes. Si desplegamos en el ds9 una imagen de flat cualquiera vemos que la región útil es circular debido al uso del reductor focal y las zonas externas permanecen oscuras debido a que no han sido expuestas. Utilizando la tarea IMEXAMINE se puede determinar la región rectangular que enmarca a la parte efectivamente expuesta la cual, aproximadamente, corresponde a la sección *[405:1705,420:1740]*. Pare ejecutar LINEBIAS debemos armar una lista con las imágenes a procesar, definir la región de overscan (*[*,2049:2058]*) y, en este caso, ejecutar la tarea usando los parámetros por default:

```
vocl> imred
imred> bias
bias> lpar linebias
        input =                 Input images
       output =                 Output images
        (bias = "[]")           Bias section
        (trim = "[]")           Trim section
      (median = no)             Use median instead of average in line bias?
 (interactive = yes)            Interactive?
    (function = "spline3")      Fitting function
       (order = 1)              Order of fitting function
  (low_reject = 3.)             Low sigma rejection factor
 (high_reject = 3.)             High sigma rejection factor
```

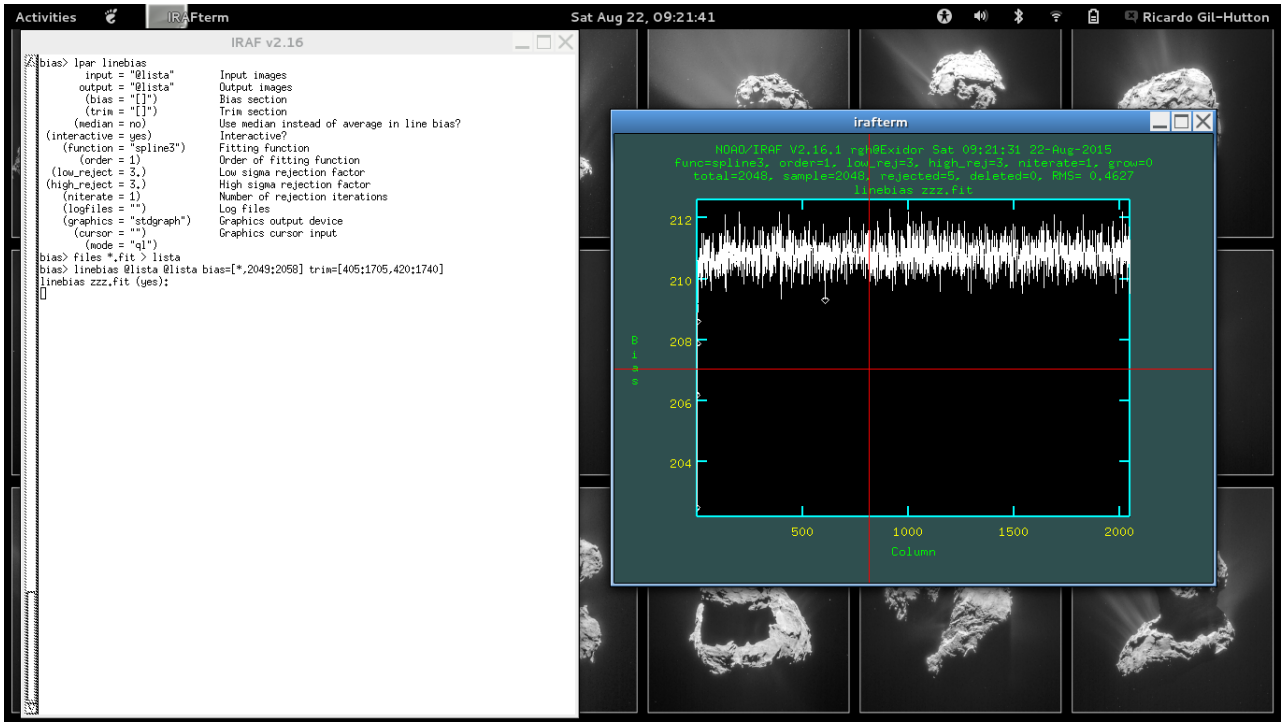

Figure 7: Ajuste interactivo de la región de overscan con la tarea LINEBIAS donde se aprecia la poca variación a de columna a columna.

```
    (niterate = 1)              Number of rejection iterations
    (logfiles = "")             Log files
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input
        (mode = "ql")

bias> files 10p* > lista
bias> files dfv* >> lista
bias> linebias @lista @lista bias=[*,2049:2058] trim=[405:1705,420:1740]
linebias 10p0014.fit (yes):
...
```

La tarea pregunta si efectivamente se quiere procesar la primera imagen y si se contesta "**yes**" se despliega una ventana gráfica mostrando el ajuste, la función que se utiliza, el número de puntos no considerados (marcados en el gráfico con un diamante), etc. (figura 7).

La tarea que en realidad hace el ajuste del overscan es ICFIT y mientras se está en la ventana gráfica se puede pedir un help presionando la tecla "**?**" y encontrar que se puede cambiar el orden y tipo de función de ajuste y modificar otros valores. Cuando se está conforme con el ajuste logrado se sale presionando "**q**" para pasar a la imagen siguiente y así sucesivamente.

La última corrección a aplicar es la que corrige por diferencia de sensibilidad pixel a pixel en el detector, para lo cual se debe conseguir una imagen flat con la mejor relación S/N posible. Entonces, vamos a combinar los flats usando la tarea IMCOMBINE para crear una imagen final "*flat.fit*" utilizando como algoritmo de rechazo *avsigclip* considerando como límites 3σ alrededor del valor promedio, asegurando al menos un valor por pixel (*nkeep=1*) y escalando las imágenes con su valor de *moda*:

```
vocl> files dfv* > lista
vocl> imcombine @lista flat.fit reject=avsigclip lsigma=3. hsigma=3. nkeep=1 scale=mode

Aug 20 23:16: IMCOMBINE
combine = average, scale = mode, zero = none, weight = none
reject = avsigclip, mclip = yes, nkeep = 1
lsigma = 3., hsigma = 3.
blank = 0.
Images     Mode  Scale
dfv0000.fit  29960.  1.000
dfv0001.fit  29946.  1.000
dfv0002.fit  29920.  1.001
dfv0003.fit  29824.  1.005
dfv0004.fit  30092.  0.996
dfv0005.fit  30237.  0.991
dfv0006.fit  30177.  0.993
dfv0007.fit  30063.  0.997
dfv0008.fit  30281.  0.989
dfv0009.fit  29926.  1.001

Output image = flat.fit, ncombine = 10
vocl>
```

Con el flat final listo, debemos corregir la contribución multiplicativa dividiendo las imágenes de objetos por el flat final renormalizado con su valor medio para no alterar el nivel de cuentas de las imágenes:

```
vocl> imstatistics flat
#  IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
flat        1718621    23272.    12413.     154.4    32325.
vocl> imarith flat / 23272. nflat
vocl> files 10p* > lista
vocl> imarith @lista / nflat @lista
vocl>
```

Con esto se termina la reducción básica realizada de forma manual. Si desplegamos las tres imágenes de objeto en tres frames del ds9, centramos las imágenes en alguna de las estrellas que están próximas al centro (poniendo el cursor sobre ellas y presionando el botón del medio del mouse), en ds9 presionamos el botón *frame* y luego en *blink* se podrá ver el cometa moviéndose a la izquierda del centro de cada imagen.

4 Reducción semi-automática:

Para realizar el proceso de reducción semi-automático tenemos que copiar nuevamente las imágenes originales en un subdirectorio para poder procesarlas. Entonces, luego de iniciar IRAF y ds9 de la manera usual se cambia al directorio "*iraf-basico*" y se crea un nuevo subdirectorio llamado "*reduccion-1*" donde se copian las imágenes que tenemos en el subdirectorio "*imagenes*" para trabajar con ellas:

```
vocl> cd <tu directorio de trabajo>/iraf-basico
```

```
vocl> mkdir reduccion-1
vocl> cd reduccion-1
vocl> imcopy ../imagenes/* .
../imagenes/10p0014.fit -> ./10p0014.fit
../imagenes/10p0019.fit -> ./10p0019.fit
../imagenes/10p0024.fit -> ./10p0024.fit
../imagenes/bias0000.fit -> ./bias0000.fit
...
...
...
vocl>
```

Una vez más debemos preparar las imagenes para procesarlas cambiando el tipo de *short* a *ushort* y de *ushort* a *real*, y agregar los keywords *GAIN* y *RDNOISE* tal como se hizo en la reducción manual. Con las imágenes listas podemos proceder a efectuar la reducción básica.

IRAF cuenta con un paquete especial para efectuar la reducción basica de las imágenes que se denomina **ccdred** y que es un sub-paquete de **imred**, entonces lo primero que debemos hacer es cargar ese paquete:

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

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

ccdred>
```

Las tareas básicas en este paquete para efectuar la reducción son ZEROCOMBINE, DARKCOMBINE, FLATCOMBINE y CCDPROC. Las primeras tres sirven para combinar imagenes bias, dark y flat con el objeto de mejorar su relación S/N. En esencia son idénticas a IMCOMBINE pero ya tienen seteados los parámetros a valores usuales para procesar esas imágenes. La tarea CCDPROC es la que realiza las correcciones por bias, dark y flat, corrige por overscan y recorta las imágenes.

Para comenzar hay que definir cuál es el instrumento con el cual se adquirieron las imágenes debido a que en ciertos casos es necesario separar las imágenes por sub-conjuntos (por ejemplo, por filtros en CCD directo) y configurar las tareas del paquete en función de esa información. La tarea que realiza este trabajo es SETINSTRUMENT y si consultamos sus parámetros se puede ver que es posible definir el sitio con su instrumental particular mediante el keyword *site*. CASLEO cuenta con su respectivo archivo de configuración pero aquí usaremos el sitio por default (*site=kpno*). Entonces:

```
ccdred> setinstrument
Instrument ID (type ? for a list) (?): ?

direct          Current headers for Sun plus CCDPROC setup for direct CCD
specphot        Current headers for Sun plus CCDPROC setup for spectropho-
```

```
tometry, ie GoldCam, barefoot CCD
hydra           WIYN Hydra with Arcon
foe             Current headers for Sun plus CCDPROC setup for FOE
fibers          Current headers for Sun plus CCDPROC setup for fiber array
coude           Current headers for Sun plus CCDPROC setup for Coude
cyrocam         Current headers for Sun plus CCDPROC setup for Cryo Cam
echelle         Current headers for Sun plus CCDPROC setup for Echelle
kpnoheaders     Current headers with no changes to CCDPROC parameters
fits            Mountain FITS header prior to Aug. 87 (?)
camera          Mountain CAMERA header for IRAF Version 2.6 and earlier

Instrument ID (type q to quit) (q): direct
...
```

Al elegir el instrumento, IRAF muestra primero el archivo de parámetros del paquete CCDRED que como no es necesario modificar se sale del editor presionando "**:q**". Inmediatamente, IRAF muestra el archivo de parámetros de la tarea CCDPROC donde debemos poner a "**no**" los parámetros *oversca*, *trim*, *zerocor*, y *flatcor* porque los controlaremos desde la línea de comandos, y el parámetro *readaxi* se debe cambiar a *column*. Al finalizar se guardan los cambios y se sale con "**:wq**":

```
                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = ccdred
TASK = ccdproc

images  =                     List of CCD images to correct
(output =                   ) List of output CCD images
(ccdtype=                   ) CCD image type to correct
(max_cac=                  0) Maximum image caching memory (in Mbytes)
(noproc =                 no) List processing steps only?

(fixpix =                 no) Fix bad CCD lines and columns?
(oversca=                 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?
(illumco=                 no) Apply illumination correction?
(fringec=                 no) Apply fringe correction?
(readcor=                 no) Convert zero level image to readout correction?
(scancor=                 no) Convert flat field image to scan correction?

(readaxi=             column) Read out axis (column|line)
(fixfile=                   ) File describing the bad lines and columns
(biassec=              image) Overscan strip image section
(trimsec=              image) 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
(minrepl=                   1.) Minimum flat field value
(scantyp=             shortscan) Scan type (shortscan|longscan)
(nscan  =                     1) Number of short scan lines

(interac=                   yes) Fit overscan interactively?
(functio=             chebyshev) Fitting function
(order  =                     1) Number of polynomial terms or spline pieces
(sample =                     *) Sample points to fit
(naverag=                     1) Number of sample points to combine
(niterat=                     1) Number of rejection iterations
(low_rej=                   3.) Low sigma rejection factor
(high_re=                   3.) High sigma rejection factor
(grow   =                   0.) Rejection growing radius
(mode   =                     q)
```

En la lista de parámetros de CCDPROC aparecen dos que son importantes y que no modificamos: *biassec* y *trimsec* donde se indica cual es la región de la imagen que se usará para calcular el overscan y cual es la sección útil de la imagen, respectivamente. Ambas ya fueron calculadas cuando se hizo la reducción manual pero en el archivo de parámetros tienen el valor "*image*" que implica que se va a leer desde un keyword de la cabecera. Esto es incorrecto pero como es preferible definir ambas regiones por línea de comandos no lo modificamos en el archivo de parámetros.

Para ver si el paquete reconoce correctamente las imágenes a procesar ejecutamos la tarea CCDLIST:

```
ccdred> ccdlist *.fit
10p0014.fit[2048,2058][real][object][]:10p
10p0019.fit[2048,2058][real][object][]:10p
10p0024.fit[2048,2058][real][object][]:10p
bias0000.fit[2048,2058][real][zero][]:
bias0001.fit[2048,2058][real][zero][]:
...
...
...
dfv0008.fit[2048,2058][real][flat][]:
dfv0009.fit[2048,2058][real][flat][]:
ccdred>
```

que indica que las imágenes son reconocidas correctamente como "*object*", "*zero*" (o bias) y "*flat*". El último campo que en este paso aparece vacio ("[ ]") usualmente contiene el filtro utilizado, pero ese valor se obtiene de un keyword llamado "*filters*" y que no se utiliza en CASLEO ni resulta importante en este caso ya que todas las imágenes fueron adquiridas con el mismo filtro. De todos modos vamos a agregar ese keyword en las cabeceras de las imágenes y vamos a repetir la tarea CCDLIST:

```
ccdred> hedit *.fit filters 3 add+ up+ ver-
add 10p0014.fit,filters = 3
10p0014.fit updated
add 10p0019.fit,filters = 3
10p0019.fit updated
add 10p0024.fit,filters = 3
```

```
10p0024.fit updated
...
...

ccdred> ccdlist *.fit
10p0014.fit[2048,2058][real][object][3]:10p
10p0019.fit[2048,2058][real][object][3]:10p
10p0024.fit[2048,2058][real][object][3]:10p
bias0000.fit[2048,2058][real][zero][3]:
bias0001.fit[2048,2058][real][zero][3]:
...
...
...
dfv0008.fit[2048,2058][real][flat][3]:
dfv0009.fit[2048,2058][real][flat][3]:
ccdred>
```

El sigiente paso en conseguir una imagen bias final para lo cual vamos a combinar los bias usando la tarea ZEROIMCOMBINE para crear una imagen final "*zero.fit*". La tarea usa por default el algoritmo de rechazo *minmax* despreciando el valor más alto de cada pixel (*nhigh=1*, *nlow=0*) y sólo considera imágenes que son del tipo "*bias*":

```
ccdred> zerocombine *.fit output=zero.fit

Aug 21 13:40: IMCOMBINE
combine = average, scale = none, zero = none, weight = none
reject = minmax, nlow = 0, nhigh = 1
blank = 0.
Images
bias0000.fit
bias0001.fit
bias0002.fit
bias0003.fit
bias0004.fit
bias0005.fit
bias0006.fit
bias0007.fit
bias0008.fit
bias0009.fit

Output image = zero.fit, ncombine = 10
ccdred>
```

Hay que notar que a pesar de que la tarea se llama ZEROCOMBINE utiliza la tarea general IMCOMBINE para obtener el bias final.

Como ahora disponemos del bias para reducir las imágenes de objeto y los flats podemos proceder a utilizar la tarea CCDPROC para remover las contribuciones aditivas. Cuando se hizo la reducción manual ya se definieron las región útil de la imagen y la que se usa para calcular el overscan, *[405:1705,420:1740]* y *[*,2049:2058]* respectivamente, asi que usaremos las mismas para esta reducción. Entonces:

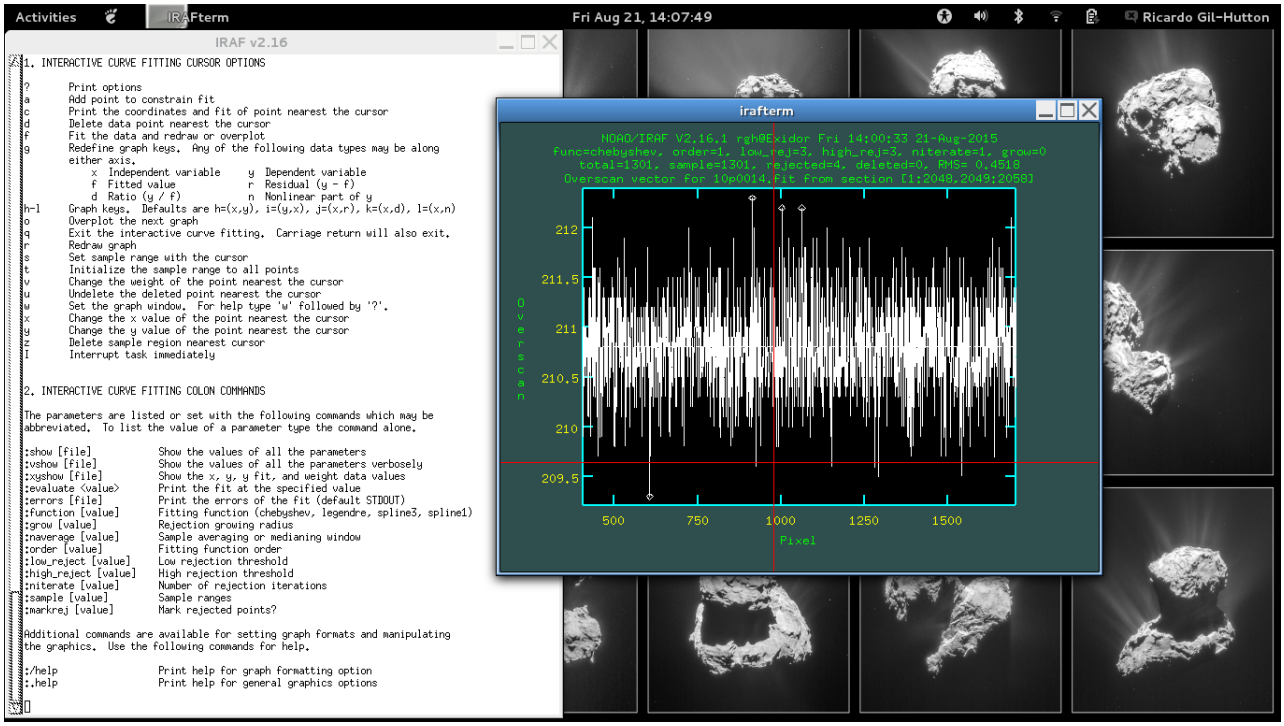

Figure 8: Ajuste interactivo de la región de overscan con la tarea CCDPROC donde se aprecia la poca variación de columna a columna.

```
ccdred> files 10p* > lista
ccdred> files dfv* >> lista
ccdred> ccdproc @lista over+ trim+ zeroc+ zero=zero biassec=[*,2049:2058] trimsec=[405:1705,420:
10p0014.fit: Aug 21 13:56 Trim data section is [405:1705,420:1740]
Fit overscan vector for 10p0014.fit interactively (yes):
...
```

Como la tareas CCDPROC se esta corriento con el valor por default para el parámetro *interactive*, pregunta para la primera imagen si el usuario quiere ajustar la región de overscan en forma interactiva. Si se contesta "**yes**" se despliega una ventana gráfica mostrando el ajuste, la función que se utiliza, el número de puntos no considerados (marcados en el gráfico con un diamante), etc. (figura 8). Nótese que en este caso sólo hace el ajuste para la región definida en *trimsec* ya que esa es la región útil de la imagen.

Al igual que en el caso de la reducción manual, la tarea que en realidad hace el ajuste del overscan es ICFIT y mientras se está en la ventana gráfica se puede pedir un help del modo usual (con la tecla "**?**") y encontrar que se puede cambiar el orden y tipo de función de ajuste y modificar otros valores. Cuando se está conforme con el ajuste logrado se sale presionando "**q**". Luego de ajustar el overscan a la primera imagen necesita repetir el proceso con la imagen bias final porque la va a usar para la reducción:

```
...
10p0014.fit: Aug 21 14:08 Overscan section is [1:2048,2049:2058] with mean=210.8013
zero: Aug 21 14:08 Trim data section is [405:1705,420:1740]
Fit overscan vector for zero interactively (yes): yes
zero: Aug 21 14:08 Overscan section is [1:2048,2049:2058] with mean=210.2302
```

```
10p0014.fit: Aug 21 14:08 Zero level correction image is zero
10p0019.fit: Aug 21 14:08 Trim data section is [405:1705,420:1740]
Fit overscan vector for 10p0019.fit interactively (yes): NO
10p0019.fit: Aug 21 14:11 Overscan section is [1:2048,2049:2058] with mean=210.6591
10p0019.fit: Aug 21 14:11 Zero level correction image is zero
10p0024.fit: Aug 21 14:11 Trim data section is [405:1705,420:1740]
10p0024.fit: Aug 21 14:11 Overscan section is [1:2048,2049:2058] with mean=210.8601
10p0024.fit: Aug 21 14:11 Zero level correction image is zero
...
...
...
```

Como el ajuste del overscan es de buena calidad en el caso de la primera imagen, cuando CCDPROC salta a la segunda imagen y pregunta si se quiere realizar un ajuste interactivo se puede contestar "**NO**" con mayúscula para que no vuelva a preguntar y siga ajustando las regiones de overscan en forma automática.

Ahora se debe obtener un flat de buena relación S/N con la tarea FLATCOMBINE. Al igual que con ZEROCOMBINE esta tarea ya tiene definidos los valores de sus parámetros por default y sólo va a trabajar con imágenes de tipo "*flat*":

```
ccdred> flatcombine *.fit output=flatf

Aug 21 14:18: IMCOMBINE
combine = average, scale = mode, zero = none, weight = none
reject = crreject, mclip = yes, nkeep = 1
rdnoise = rdnoise, gain = gain, snoise = 0., hsigma = 3.
blank = 1.
Images         Mode Rdnoise   Gain  Scale
dfv0000.fit  29943.    3.14   2.18  1.003
dfv0001.fit  29928.    3.14   2.18  1.003
dfv0002.fit  29902.    3.14   2.18  1.004
dfv0003.fit  29806.    3.14   2.18  1.007
dfv0004.fit  30072.    3.14   2.18  0.998
dfv0005.fit  30218.    3.14   2.18  0.994
dfv0006.fit  30160.    3.14   2.18  0.995
dfv0007.fit  30042.    3.14   2.18  0.999
dfv0008.fit  30258.    3.14   2.18  0.992
dfv0009.fit  29905.    3.14   2.18  1.004

Output image = flatf3, ncombine = 10
ccdred>
```

Como la corrección por flat depende del filtro utilizado, la tarea FLATCOMBINE le agrega al final del nombre un identificador (un "**3**") para no confundir este flat con otro adquirido para otro filtro.

Para terminar la reducción es necesario corregir las imágenes por contribuciones multiplicativas usando una vez más la tarea CCDPROC para dividir por el flat final y renormalizar:

```
ccdred> files 10p* > lista
ccdred> ccdproc @lista flatc+ flat=flatf3
10p0014.fit: Aug 21 14:25 Flat field image is flatf3 with scale=23284.79
```

```
10p0019.fit: Aug 21 14:25 Flat field image is flatf3 with scale=23284.79
10p0024.fit: Aug 21 14:25 Flat field image is flatf3 with scale=23284.79
ccdred>
```

Como se puede ver, los resultados obtenidos son idénticos a los logrados con la reducción manual. Una ventaja del proceso semi-automático es que a medida que se desarrolla le agrega información a las cabeceras de las imágenes. Si utilizamos nuevamente la tarea CCDLIST:

```
ccdred> ccdlist *.fit
10p0014.fit[1301,1321][real][object][3][OTZF]:10p
10p0019.fit[1301,1321][real][object][3][OTZF]:10p
10p0024.fit[1301,1321][real][object][3][OTZF]:10p
bias0000.fit[2048,2058][real][zero][3]:
bias0001.fit[2048,2058][real][zero][3]:
...
...
...
dfv0008.fit[1301,1321][real][flat][3][OTZ]:
dfv0009.fit[1301,1321][real][flat][3][OTZ]:
ccdred>
```

En el listado se agrega ahora un campo donde se indica qué procesos se realizaron sobre la imagen. Esta información la escribe la tarea CCDPROC al final de las cabeceras donde también se incluye información sobre el proceso de reducción.

5 Algunas notas adicionales

- El proceso de reducción que se describió corresponde a un proceso que corresponde a observaciones de imagen directa. En el caso de espectroscopía resulta necesario realizar algunos pasos más para terminar la reducción básica.

  En **espectroscopía de ranura larga** usualmente se requiere remover ciertos defectos que aparecen en los flats debido a diferentes razones (por ejemplo, la lámpara es de temperatura muy diferente que las estrellas, hay efectos de trasmisión en los filtros de las lámparas, la reflectividad de la pantalla tiene cierta dependencia con la longitud de onda, etc.). Para corregir estos efectos es necesario normalizar el flat final obtenido con la tarea RESPONSE del paquete **specred**. Esta tarea permite ajustar interactivamente el flat en la dirección de dispersión y lo que se obtiene es una imagen de salida que corresponde a el cociente del flat al ajuste obtenido. Los parámetros de esta tarea son:

  ```
  ccdred> specred
        aidpars@      aprecenter    continuum     lscombine     scopy         specplot
        apall         apresize      deredden      msresp1d      sensfunc      specshift
        apdefault@    apscatter     dispcor       odcombine     setairmass    splot
        apedit        apsum         dofibers      refspectra    setjd         standard
        apfind        aptrace       dopcor        reidentify    sfit          telluric
        apfit         autoidentify  doslit        response      sflip         transform
        apflatten     background    fitprofs      sapertures    skysub
        apmask        bplot         identify      sarith        skytweak
  ```

```
      apnormalize   calibrate     illumination  scombine      slist

specred> lpar response
  calibration =                 Longslit calibration images
 normalizatio =                 Normalization spectrum images
     response =                 Response function images
 (interactive = yes)            Fit normalization spectrum interactively?
   (threshold = INDEF)          Response threshold
      (sample = "*")            Sample of points to use in fit
    (naverage = 1)              Number of points in sample averaging
    (function = "spline3")      Fitting function
       (order = 1)              Order of fitting function
  (low_reject = 0.)             Low rejection in sigma of fit
 (high_reject = 0.)             High rejection in sigma of fit
    (niterate = 1)              Number of rejection iterations
        (grow = 0.)             Rejection growing radius
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input
        (mode = "q")
specred>
```

El ajuste se hace en forma interactiva en la dirección de dispersión. Hay que prestar atención al detalle de que se pretende ajustar **sólo** la estructura en gran escala del flat que depende de la longitud de onda, no los detalles de pequeña escala. Tanto la imagen de calibración como la de normalización corresponden al flat final que ya se tiene y la de respuesta será la que se obtenga después del ajuste. Por ejemplo:

```
specred> response flatf flatf resp low_rej=3. high_rej=3.
...
specred>
```

La imagen final obtenida para el flat corregido es la que se debe usar en el último paso por CCDPROC para corregir las imágenes de espectroscopía de ranura larga.

En el caso de **espectroscopía echelle** es necesario normalizar el flat a lo largo del eje de dispersión para cada orden. Para esto se utiliza la tarea APFLATTEN del paquete **echelle**:

```
ccdred> echelle
      apall         apnormalize   calibrate     dopcor        scopy         specshift
      apdefault@    aprecenter    continuum     ecidentify    sensfunc      splot
      apedit        apresize      demos         ecreidentify  setairmass    standard
      apfind        apscatter     deredden      refspectra    setjd
      apfit         apsum         dispcor       sapertures    sflip
      apflatten     aptrace       doecslit      sarith        slist
      apmask        bplot         dofoe         scombine      specplot

echelle> lpar apflatten
        input =                 List of images to flatten
       output =                 List of output flatten images
```

```
   (apertures = "")             Apertures
  (references = "")             List of reference images
 (interactive = yes)            Run task interactively?
        (find = yes)            Find apertures?
     (recenter = yes)           Recenter apertures?
       (resize = yes)           Resize apertures?
         (edit = yes)           Edit apertures?
        (trace = yes)           Trace apertures?
     (fittrace = yes)           Fit traced points interactively?
      (flatten = yes)           Flatten spectra?
      (fitspec = yes)           Fit normalization spectra interactively?
         (line = INDEF)         Dispersion line
         (nsum = 10)            Number of dispersion lines to sum or median
    (threshold = 10.)           Threshold for flattening spectra
         (pfit = "fit1d")       Profile fitting type (fit1d|fit2d)
        (clean = no)            Detect and replace bad pixels?
   (saturation = INDEF)         Saturation level
    (readnoise = "0.")          Read out noise sigma (photons)
         (gain = "1.")          Photon gain (photons/data number)
       (lsigma = 4.)            Lower rejection threshold
       (usigma = 4.)            Upper rejection threshold\n
     (function = "legendre")    Fitting function for normalization spectra
        (order = 1)             Fitting function order
       (sample = "*")           Sample regions
     (naverage = 1)             Average or median
     (niterate = 0)             Number of rejection iterations
   (low_reject = 3.)            Lower rejection sigma
  (high_reject = 3.)            High upper rejection sigma
        (grow = 0.)             Rejection growing radius
        (mode = "q")
echelle>
```

Esta tarea es relativamente compleja y en primera medida ingresa a un editor donde se pueden marcar la posición y ancho de las aperturas de cada orden, para luego trazar interactivamente el espectro y realizar el ajuste. Finalmente, antes de utilizar este flat normalizado es necesario confirmar en su cabecera que el keyword *ccdmean*, el cual fue agregado originalmente por FLATCOMBINE, cambió su valor a "1" en lugar de su valor original que correspondía al valor medio del flat final original. Técnicamente, este último paso se hace automáticamente en las últimas versiones de IRAF.

- Muchas veces es importante remover todos los pixels malos de una imagen para evitar defectos en la fotometría o en la extracción de un espectro. Para logra este objetivo es necesario identificar los pixeles defectuosos en el detector y el mejor modo de hacerlo es construir una máscara.

  Para obtener la máscara se deben adquirir dos conjuntos de flats de cúpula, uno con alta relación S/N (por ejemplo, decenas de miles de e^-/px) y otro con baja relación S/N (algunos cientos de e^-/px). Con estos dos conjuntos de flats, y luego de haber removido todas las contribuciones aditivas, se combinan ambos conjuntos de flats independientemente con la tarea FLATCOMBINE para luego dividir uno por el otro. Si los flats con alta relación S/N se llaman "*flth*.fit*" y los otros "*fltl*.fit*":

```
ccdred> files flth* > largos
ccdred> files fltl* > cortos
ccdred> flatcombine @largos output=flat-lar reject=crreject scale=mode
ccdred> flatcombine @cortos output=flat-cor reject=crreject scale=mode
ccdred> imarith flat-lar / flat-cor flat-div
ccdred> ccdmask flat-div mask=mascara
ccdred> imdelete @largos, @cortos, flat-lar, flat-cor
ccdred>
```

La máscara que se creó tiene un valor de "0" en los pixeles que no son defectuosos, un valor de "1" donde es necesario interpolar en sentido vertical, y un valor de "2" donde se debe interpolar en sentido horizontal. Es conveniente guardar la máscara en un directorio diferente a donde se están procesando las imágenes para evitar que sea alterada al ser procesada por CCDPROC en los pasos siguientes.

Luego de terminado el proceso de reducción básica de las imágenes de objetos se puede utilizar esta máscara para interpolar en la imagen para corregir los pixeles defectuosos. Para esto utilizamos la tarea FIXPIX del paquete **proto** y la máscara creada previamente:

```
ccdred> proto
      binfil          fixpix          interp          mskexpr         text2mask
      bscale          hfix            irafil          mskregions      vol.
      color.          imcntr          joinlines       ringavg
      epix            imextensions    mimstatistics   rskysub
      fields          imscale         mkglbhdr        suntoiraf

proto> lpar fixpix
       images =                List of images to be fixed
        masks =                List of bad pixel masks
     (linterp = "INDEF")       Mask values for line interpolation
     (cinterp = "INDEF")       Mask values for column interpolation
     (verbose = no)            Verbose output?
      (pixels = no)            List pixels?
        (mode = "ql")
proto> fixpix *.fit mask=mascara lint=1 cint=2
proto>
```