

Cúmulos Abiertos

Introducción a IRAF

Javier Alejandro Acevedo Barroso*

Universidad de los Andes, Bogotá, Colombia

18 de septiembre de 2019

1. Image Reduction and Analysis Facility

Image Reduction and Analysis Facility (IRAF) es un software de reducción y análisis de imágenes mantenido por la «National Optical Astronomy Observatories» (NOAO) en Estados Unidos. El proyecto IRAF nace en 1981 en el Observatorio Nacional Kitt Peak, en donde de 1982 a 1984 se escribió la primera versión del software y su lenguaje de script para el usuario «Command Language» (CL). En 1984 empezó su uso científico en el «Space Telescope Science Institute» impulsando la implementación del software en diferentes arquitecturas de máquinas de la época, en particular, se implementó para la arquitectura UNIX en 1986. Para principios de la siguiente década IRAF es el estándar científico de procesamiento de imagen en Estados Unidos y observatorios asociados[1].

Así mismo, durante esa década el software se adaptó a las innovaciones tecnológicas, tales como: la programación orientada a objetos, que surge del refinamiento de las prácticas de programación durante los ochenta; las redes de computadores, en particular la recién nacida World Wide Web; las interfaces de usuario, que en el caso de IRAF evolucionó a XGterm y DS9; y los lenguajes de programación de alto nivel, que en ese momento eran C, C++ y Fortran[2].

Para el nuevo milenio IRAF era el estándar mundial en reducción y análisis de imagen astronómicas, por encima de proyectos similares como «Starlink Project» que fue también un software de reducción y procesamiento de datos pero escrito por la comunidad astronómica de Reino Unido.

*e-mail: ja.acevedo12@uniandes.edu.co

Sin embargo, NOAO dejó de distribuir tanto el código fuente como archivos binarios de IRAF desde finales de 2018. EL proyecto sigue vivo gracias a la comunidad que reescribió buena parte del código como software libre y lo publicó como un repositorio de GITHUB.

IRAF está organizado en «paquetes» que contienen rutinas específicas denominadas «tareas». Cada paquete agrupa tareas relacionadas de acuerdo a su uso. Por ejemplo, hay paquetes que traen todo lo relacionado a la descompresión y carga de imágenes, o paquetes dedicados a ciertos tipos de análisis como astrometría o fotometría.

2. SAOImageDS9

SAOImageDS9 (comúnmente llamado DS9) es un software de visualización y creación de imágenes a partir de datos. Las principales particularidad de DS9 son: Dado lo temprano de su desarrollo, sus algoritmos fueron un fuerte avance no solo para la astronomía, sino también para la visualización computación en sí; su capacidad de procesar archivos de formato .fits, que es el formato más utilizado en astronomía; y la fácil comunicación con herramientas de análisis externas, como IRAF, permitiendo realizar el procesamiento de los datos en un ordenador diferente al que se usa para la visualización.

La historia de DS9 empieza en 1990 cuando Mike Van Hilst escribe SAOImage en el «Smithsonian Astrophysical Observatory» (SAO). El software, siendo uno de los primeros en su tipo, representó técnicas innovadoras en la visualización de datos, al punto de que hoy muchos programas de visualización de datos implementan algoritmos originalmente de SAOImage. El éxito inicial de SAOImage llevó al desarrollo durante los noventa de «SAOImage, The Next Generation», que extendió la funcionalidad de SAOImage a nuevos entornos gráficos (nótese que fue en esta década que se desarrollaron los GUI como los entendemos hoy). Por último, finales de los noventa, se desarrolló el sucesor a «SAOImage, The Next Generation», que llamaron «SAOImageDS9» para continuar con un esquema de nombres relacionado a Star Trek (The Next Generation y Deep Space 9 son los títulos de diferentes series que toman lugar en el universo de Star Trek). La primera versión de DS9 fue lanzada al público en 1999, y, gracias al financiamiento de NASA y el «Chandra X-Ray Science Center» entonces ha recibido soporte continuo de parte de SAO. La versión actual de DS9 es 8 estable y 8.0.1 en beta.[\[3\]](#)

3. Formato de archivos

Aunque para el procesador todos los tipos de archivos son finalmente conjuntos de unos y ceros, para el usuario no es lo mismo un archivo de vídeo con extensión .avi que un programa en c. Por esto, se definieron los formatos de archivo. Cada archivo tiene un formato que describe el tipo de información que tiene, adicionalmente, la mayoría de sistemas operativos relacionan el formato de un archivo con la extensión del mismo. Por ejemplo, un archivo .jpeg es un archivo comprimido de imagen. La extensión puede ser un acrónimo (como Joint Photographic Expert Group JPEG) o en general cualquier conjunto de 1-4 letras. La mayoría de extensiones corresponden a protocolos particulares para ese tipo de archivo. Por ejemplo, las extensiones .avi .mkv .MPEG representan archivos de vídeos, pero a la vez representan el protocolo que se usa para comprimir o codificar los archivos y recuperar el vídeo a partir de una secuencia binaria.

IRAF y la comunidad astronómica internacional se valen del formato FITS (Flexible Image Transport System) para el almacenamiento y transporte de imágenes. El formato se desarrolló a principio de los ochenta y fue pensado para transportar imágenes bastante grandes para el tamaño de las unidades de almacenamiento de la época (no era raro ver astrónomos con Gigabytes de datos almacenados en cintas magnéticas de solo cientos de Kilobytes de capacidad). Otra particularidad del formato FITS es la inclusión de un header en cada imagen que puede ser leído por el usuario (a través de tareas de IRAF) e incluye información del campo de la imagen, las coordenadas, los pixeles muertos, etc.

4. Reporte de la instalación de IRAF

4.1. Versión de NOAO

A continuación se presenta el reporte de la primera instalación exitosa en 2018, cuando IRAF aún era distribuido por NOAO. Estas instrucciones son obsoletas pero las conservo por el registro histórico y porque puede ser útil tener listadas las antiguas dependencias de IRAF.

Dada la época en donde se escribió la primera versión del software, la interfaz de usuario de IRAF es bastante robusta. Si bien, se ha recopilado paquetes binarios para instalar IRAF usando un instalador automático, la necesidad de usar librerías no incluidas y software adicional como «ximtools», así como los diferentes niveles de permisos que requiere IRAF durante su instalación, dan lugar a un zoológico de errores posibles.

Por lo anterior, la instalación de IRAF tomó numerosos intentos. El software se instaló en un computador de mesa y un portátil comercial, ambos con instalaciones funcionales de Ubuntu 18.04. Los primeros intentos de instalación fallaron por errores a la hora de elegir los paquetes binarios a descargar, pues estos deben coincidir no solo con el sistema operativo (Linux) sino también con la arquitectura del procesador. Las siguientes instalaciones sufrieron de problemas a la hora de obtener las librerías de Linux que requiere IRAF pues estas cambiaron ligeramente de nombre. Tras varios intentos se logró tener una instalación exitosa en el computador de mesa, sin embargo, en el computador portátil hay errores adicionales, pues nunca se puede ejecutar correctamente el comando «ecl». Finalmente se optó por usar el computador de mesa para el trabajo en IRAF y el portátil únicamente para tomar nota de los comandos en clase.

La instalación exitosa se realizó utilizando un tutorial de la universidad de Ohio para linux de 64 bits[4]. Dado el cambio en el nombre de los paquetes, esta es la versión correcta del comando a usar:

```
sudo apt install tcsh libxss1 lib32z1 lib32ncurses5  
libxmu-dev libbz2-1.0:i386
```

4.2. Versión de la comunidad

Desde inicios de 2019, IRAF se encuentra en los paquetes disponibles en los repositorios oficiales de Debian. Por lo anterior, la instalación se reduce a:

```
sudo apt install irafcl
```

Sin embargo, si se está usando una distribución no basada en Debian, o no se quiere utilizar la versión de IRAF de los repositorios de Debian, se puede compilar el código fuente de la versión de IRAF de la comunidad.

El código fuente se encuentra disponible como un repositorio de Github. Para descargarlo es necesario instalar Git:

```
sudo apt install git
```

Y luego se puede clonar el repositorio con:

```
git clone https://github.com/iraf-community/iraf.git
```

Para la instalación es necesario seguir detalladamente todos los pasos del archivo README dentro de la carpeta «iraf-community».

Se debe repetir el proceso para X11tools (si se quiere usar XGterm o tareas como implot) y para cada librería externa (como MSCRED).

Respecto a DS9, se puede obtener un binario ejecutable listo para ser utilizado desde la página web oficial <http://ds9.si.edu/site/Download.html> . El archivo se descarga y se puede copiar en la carpeta «/usr/local/bin» para que ejecutarlo sea simplemente llamar desde la consola:

```
ds9
```

También se puede llevar el archivo binario como una version portatil del programa. Por lo que se puede tener en la misma carpeta y simplemente correr:

```
./ds9
```

5. Ejercicio

El ejercicio consiste en seguir el tutorial de IRAF «intro.txt». En este tutorial se presentan algunas tareas y paquetes básicos del programa, para luego hacer un ejercicio de alinear dos imágenes ligeramente corridas entre sí, y promediarlas después. A continuación se presentarán los comandos que considero prudente registrar para futuras referencias y algunos de sus resultados.

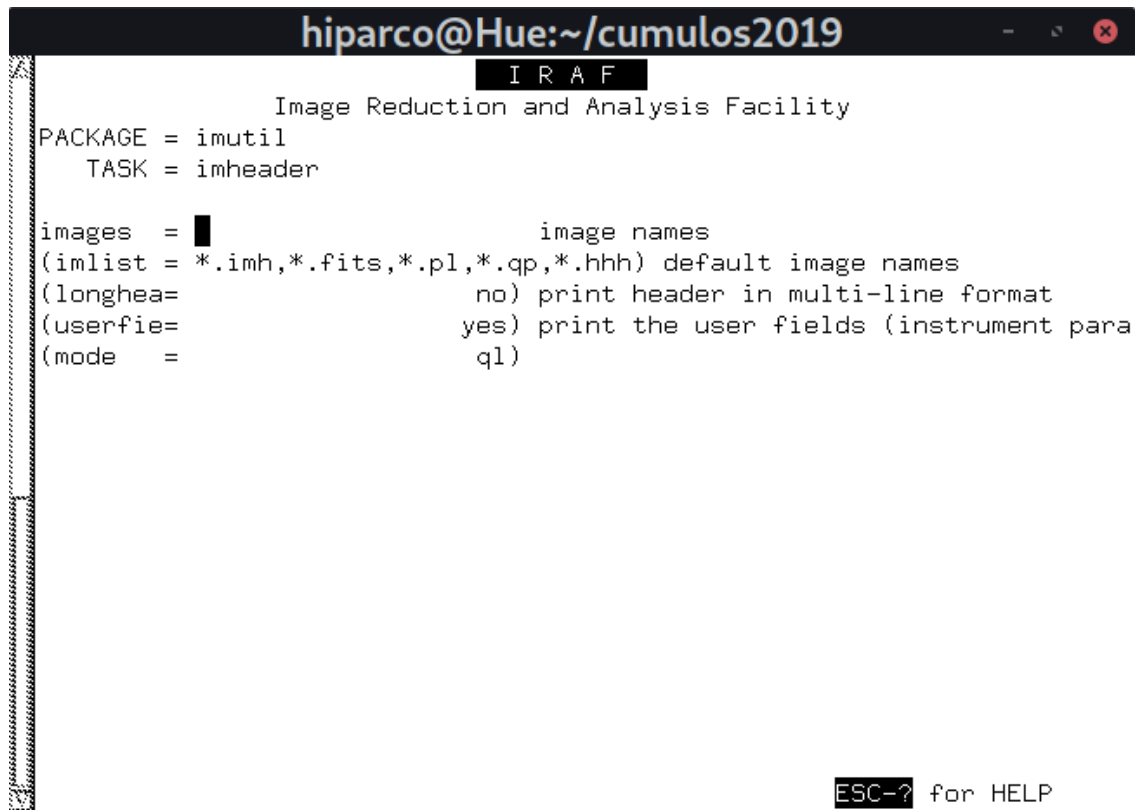
Para descomprimir imágenes se debe primero usar el paquete «rfits». El primer paso con cada paquete que se usa por primera vez, es borrar las instrucciones de uso anteriores o que vengan por default, esto se hace utilizando el comando «UNLEARN»:

```
ecl> unlearn rfits
```

```
hiparco@Hue:~/cumulos2019
ecl> unlearn rfits
ecl> rfits fintro* "" junk old+
File: fintro0001 -> junk0001.fits M-92 V size=318x5
09
    bitpix=-32 scaling=none pixtype=real
    Image junk0001.fits renamed to im010.imh
File: fintro0002 -> junk0002.fits M-92 V size=318x5
09
    bitpix=-32 scaling=none pixtype=real
    Image junk0002.fits renamed to im011.imh
ecl> ls
fintro0001 fintro0002 im010.fits im011.fits
ecl> ls
fintro0001 fintro0002 im010.fits im011.fits
ecl> ~
```

Figura 1: Carga de rfits y sucesiva descompresión de imágenes. se listó los archivos del directorio para confirmar que sí se generaran los .fits.

Para visualizar los parámetros ocultos de una tarea se usa el comando «lpar» y para modificarlos se usa «epar» (ver imagen 2). Para dejar de editar un documento se oprime escape y se teclea «:wq» para guardar y salir. Si se quiere salir sin guardar: «:q!», y si se quiere solo guardar: «:w».



```
hiparco@Hue:~/cumulos2019
I R A F
Image Reduction and Analysis Facility
PACKAGE = imutil
TASK = imheader

images =
(imlist = *.imh,*.fits,*.pl,*.qp,*.hhh) default image names
(longhea= no) print header in multi-line format
(userfie= yes) print the user fields (instrument para
(mode = ql)

ESC-? for HELP
```

Figura 2: Modificando los parámetros ocultos de imhead usando epar.

El ejercicio presenta algunos comandos de «ecl» (el lenguaje de scripting de IRAF) similares a comandos tradicionales de UNIX. No los incluyo porque versiones recientes de IRAF incluyen los comandos clásicos de UNIX, y porque se puede ejecutar cualquier comando de la bash con anteponer un signo de exclamación al comando.

El tutorial enseña que se puede ver las imágenes .fits con el programa DS9 (ver sección 2). Para ejecutarlo, basta correr

```
ecl> !ds9 &
```

Y para finalmente mostrar una imagen, se usa el comando «DISPLAY»:

```
ecl> display [imagen] [recuadro]
```

Donde imagen corresponde a la ruta al archivo (si está en la misma carpeta, es solo el nombre), y recuadro corresponde al número del recuadro en donde dibujar la imagen (mínimo 1).

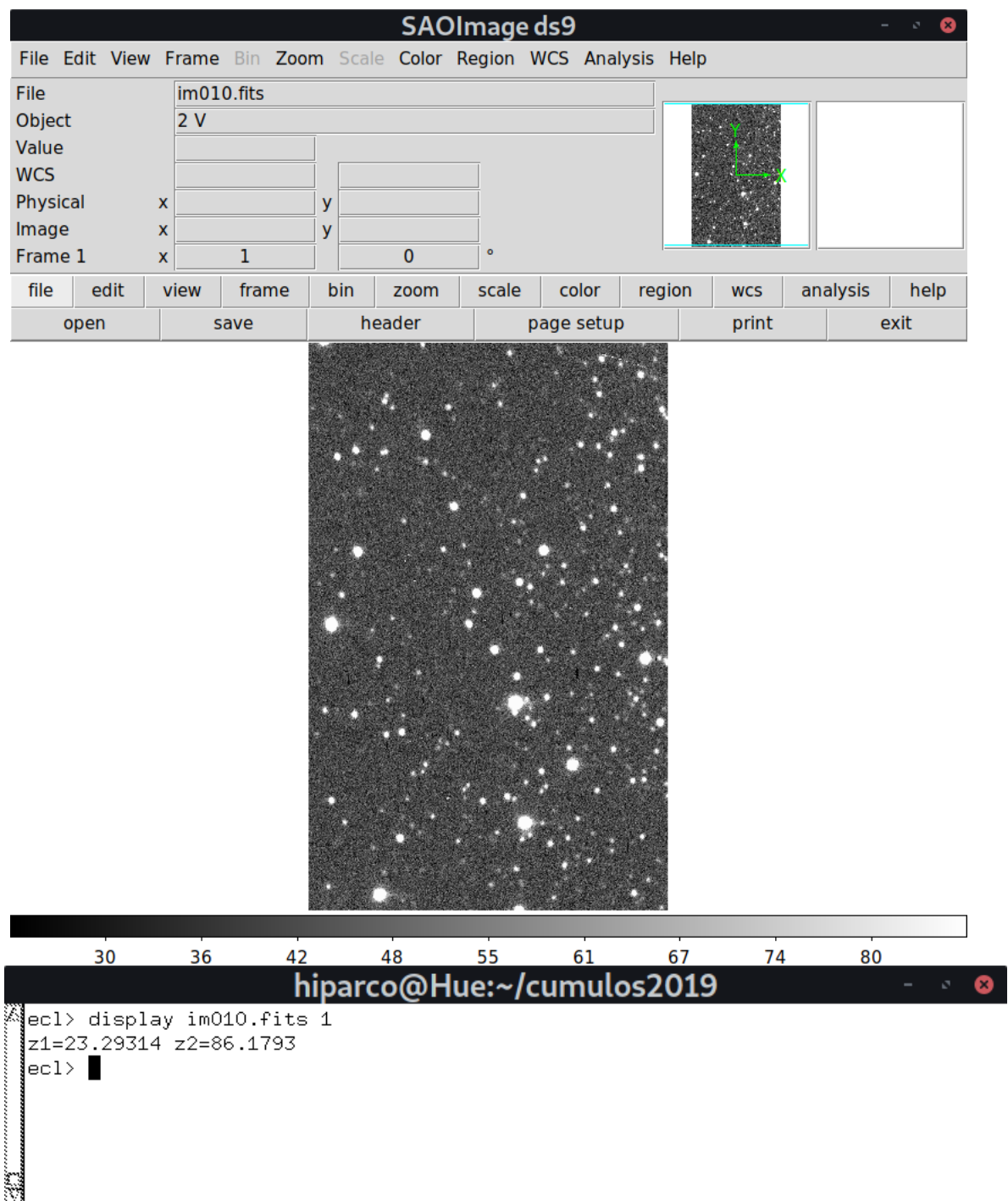


Figura 3: Arriba: imagen im010.fits cargada en ds9. Abajo: respuesta de IRAF al comando DISPLAY.

El siguiente objetivo del ejercicio es alinear dos imágenes astronómicas del mismo campo pero ligeramente desplazadas entre sí. Para esto, se ubicará 10 estrellas o más en ambas imágenes, y con eso se calcula el corrimiento entre ellas. Sin embargo, se debe tener especial cuidado eligiendo cuales son estrellas pues no todos los puntos de la imagen son estrellas. Para verificar si un punto es una estrella se debe estudiar el perfil radial. En IRAF, esto se hace con la tarea «IMEXAMINE»:

```
ecl> imexamine im010.fits
```

Tras ejecutar la tarea en la sesión de IRAF, se vuelve a DS9, en donde el cursor ha cambiado a modo interactivo. Para cada punto candidato a estrella, se ubica el cursor en él y se presiona la tecla «r», si todo se hizo correctamente (y no se está usando la versión 7.2 de DS9 pues tiene un bug en el cálculo de las coordenadas del cursor), se debe obtener una nueva ventana con la gráfica del perfil radial (ver imagen 4).

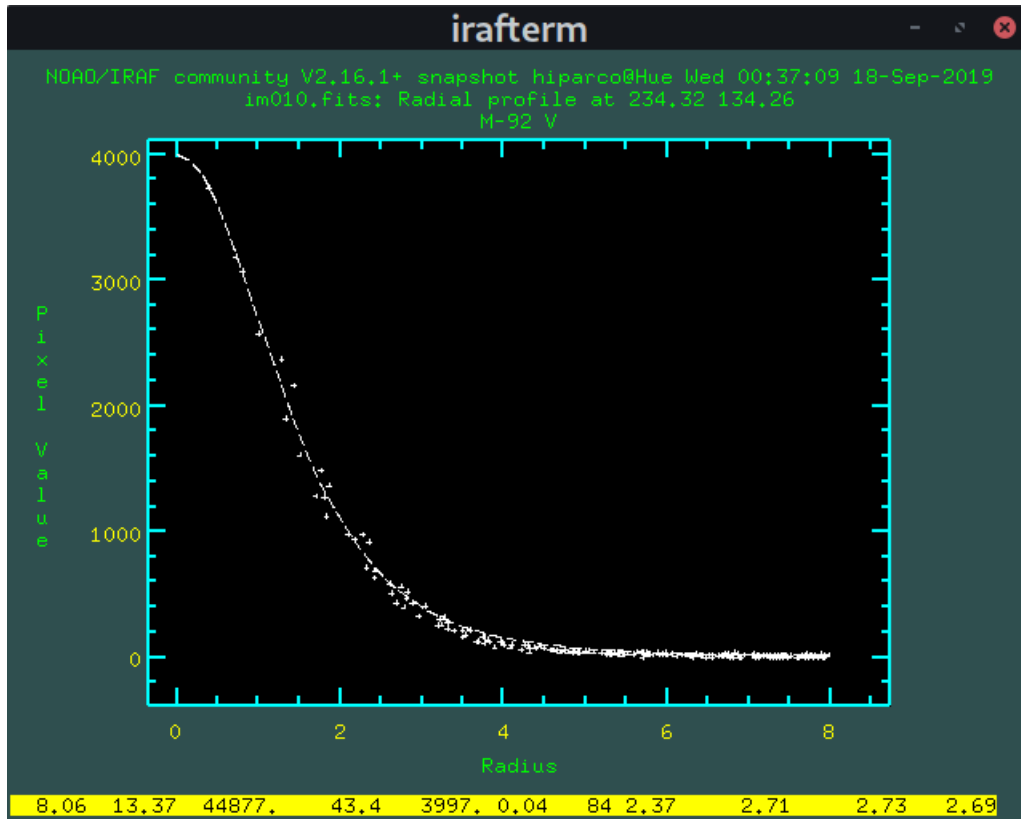


Figura 4: Perfil radial de un buen candidato a estrella. Nótese lo ceñido que están los puntos a la línea de tendencia, esto es una cuantificación visual de que tanto se ajusta el perfil radial del punto al de una estrella.

La gráfica generada muestra el valor del pixel contra distancia al centro del punto, y lo contrasta con una línea de tendencia. Si la fuente es una estrella, se espera que los puntos estén cerca a la línea de tendencia pues son emitidos por una fuente coherente y no son ruido aleatorio. La figura 4 es un ejemplo de un buen candidato a estrella. Un mal candidato a estrella tiene mucha mayor dispersión (ver figura 5).

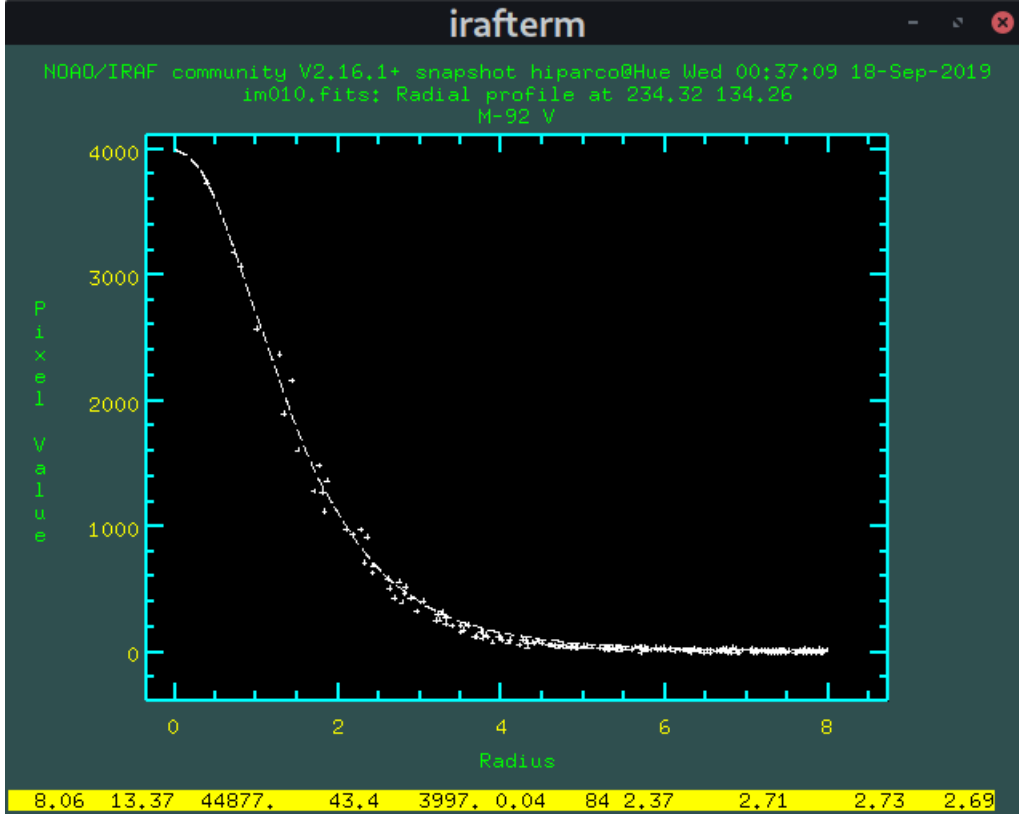


Figura 5: Perfil radial de un mal candidato a estrella. Se observa alta dispersión respecto a la línea de tendencia. y que el máximo del valor del pixel no es muy grande, incluso se observan valores del pixel negativos

Una vez seleccionados los candidatos razonables, y diferenciando con un círculo verde (ver figura 6), se debe tomar la posición de cada uno y contrastarla con la posición de la misma estrella en la otra imagen. Al promediar la diferencia se obtiene un vector de corrimiento (\vec{d}). Dejando fija la imagen «im010», las nuevas coordenadas de «im011» estarán dadas por:

$$\vec{x'} = \vec{x} + \vec{d} \quad (1)$$

Usando Python para promediar se obtiene un vector $\vec{d} = -0,53\hat{x} - 1,63\hat{y}$. Por último, tanto el código utilizado, como los datos de las estrellas tomadas se encuentran en el repositorio de github <https://github.com/ClarkGuilty/2018/tree/master/Astronomia/tareaIrafi>

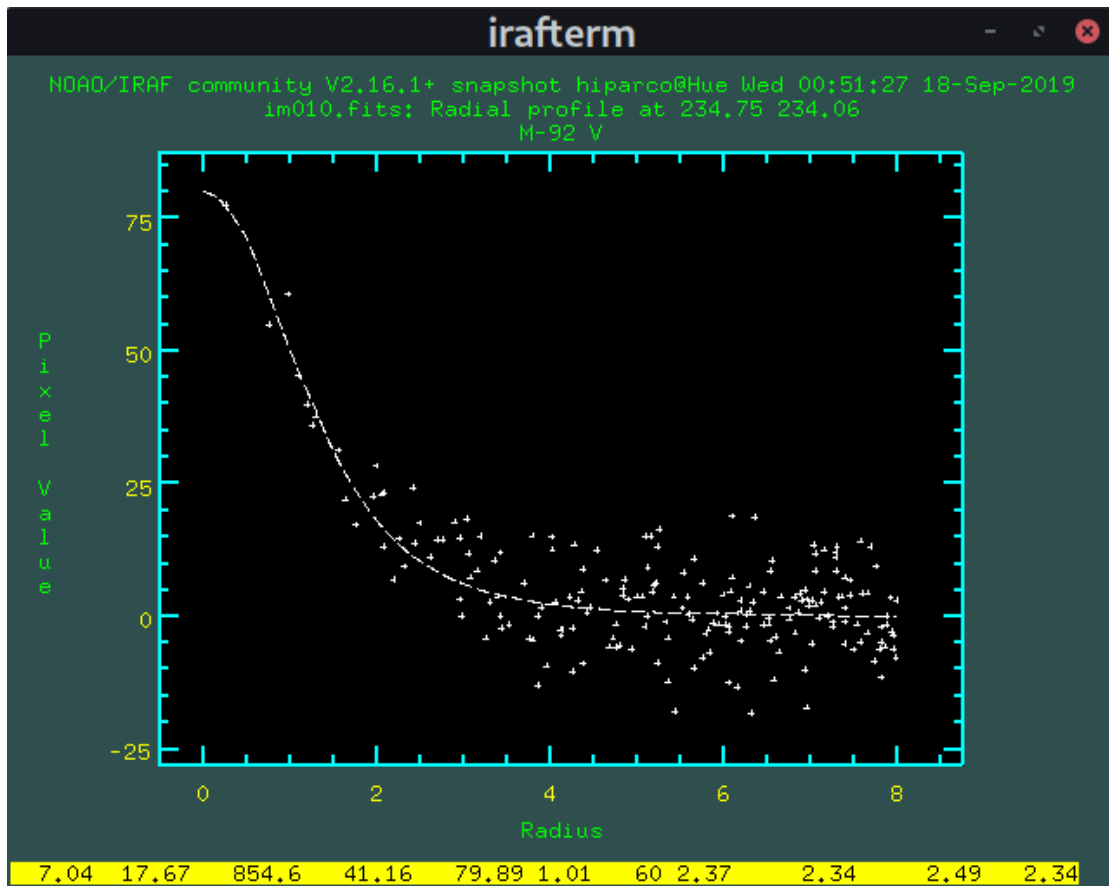


Figura 6: Candidatos a estrella seleccionados por su perfil radial. No se pudo seleccionar ningún buen candidato de la región superior derecha.

Referencias

- [1] D. Tody. The IRAF Data Reduction and Analysis System. In D. L. Crawford, editor, *Instrumentation in astronomy VI*, volume 627, page 733, January 1986.
- [2] D. Tody. IRAF in the Nineties. In R. J. Hanisch, R. J. V. Brissenden, and J. Barnes, editors, *Astronomical Data Analysis Software and Systems II*, volume 52 of *Astronomical Society of the Pacific Conference Series*, page 173, January 1993.
- [3] The story of saomageds9: How ds9 got its name. <http://ds9.si.edu/doc/story.html>. Accessed: 2018-09-06.

- [4] Iraf setup on ubuntu/debian linux. http://www.astronomy.ohio-state.edu/~khan/iraf/iraf_step_by_step_installation_64bit. Accessed: 2018-09-06.