Procesamiento y Análisis de Datos Astronómicos

14.- Procesamiento y Análisis de imágenes

R. Gil-Hutton

Marzo 2020

Actividades:

Práctica 11:

- Utilizando la descomposición en Componentes Principales obtenida haga un agrupamiento jerárquico de los objetos. Indique cuál sería un valor máximo para definir grupos.
- Simule una muestra limitada por flujo (luminosidad, magnitud absoluta) de objetos distribuídos en un gran volumen de espacio utilizando para el flujo una distribución en ley de potencias del tipo:

$$\rho(> f) \propto f^{-\gamma}$$

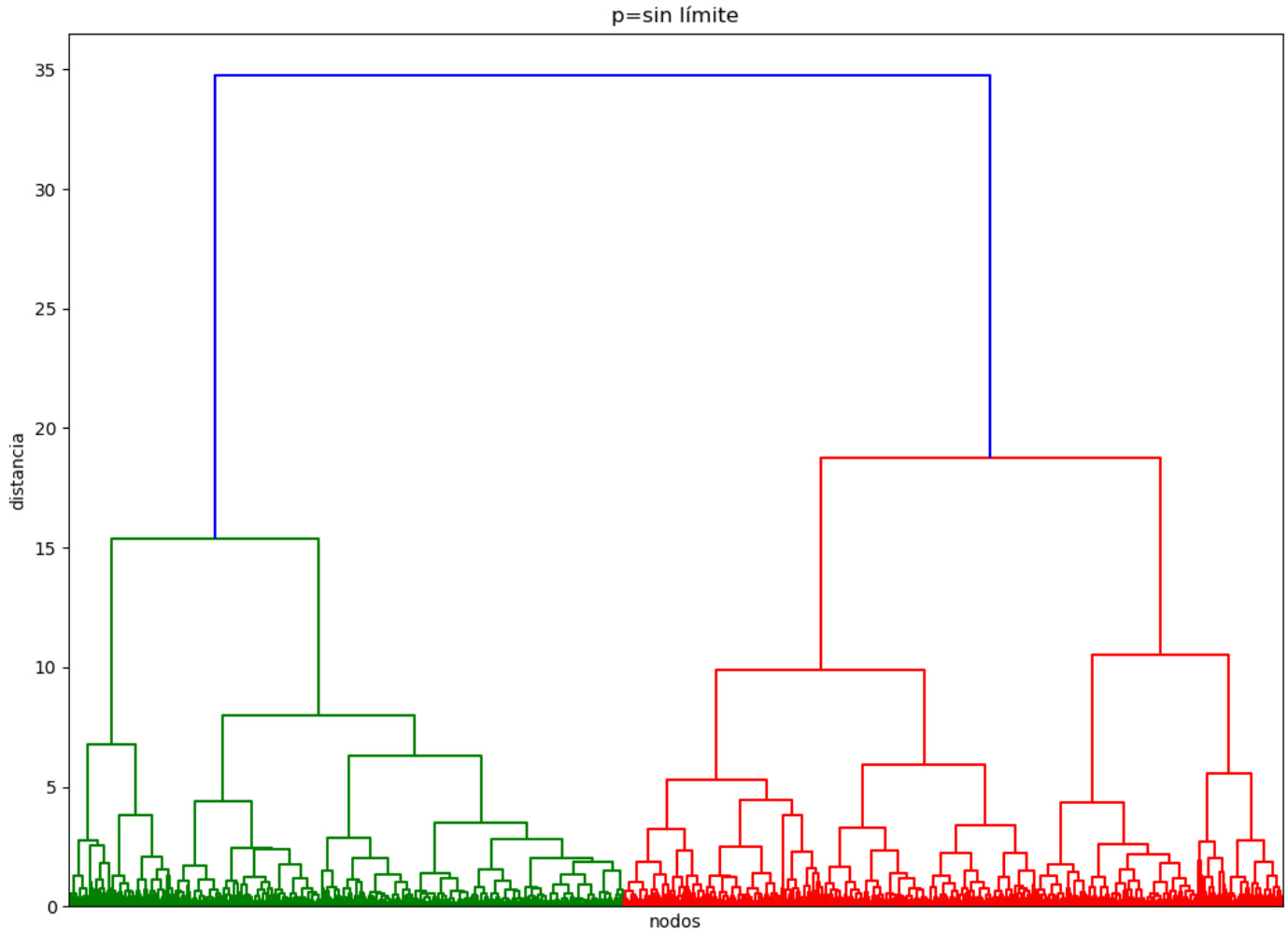
y una distribución volumétrica uniforme para la distancia. Para armar la muestra fije un valor límite para el flujo de manera arbitraria.

Actividades:

Práctica 11:

- Con esta muestra:
  - aplique el método V_{max} y vea si es posible recuperar la distribución de flujo utilizada.
  - produzca errores utilizando bootstrap y compare los resultados con errores basados en $\sqrt{N}$.
  - asigne a cada objeto dos flujos diferentes, detecte y utilice el método de Kaplan-Meier para normalizar la distribución. Aparece alguna correlación?.

Actividades:

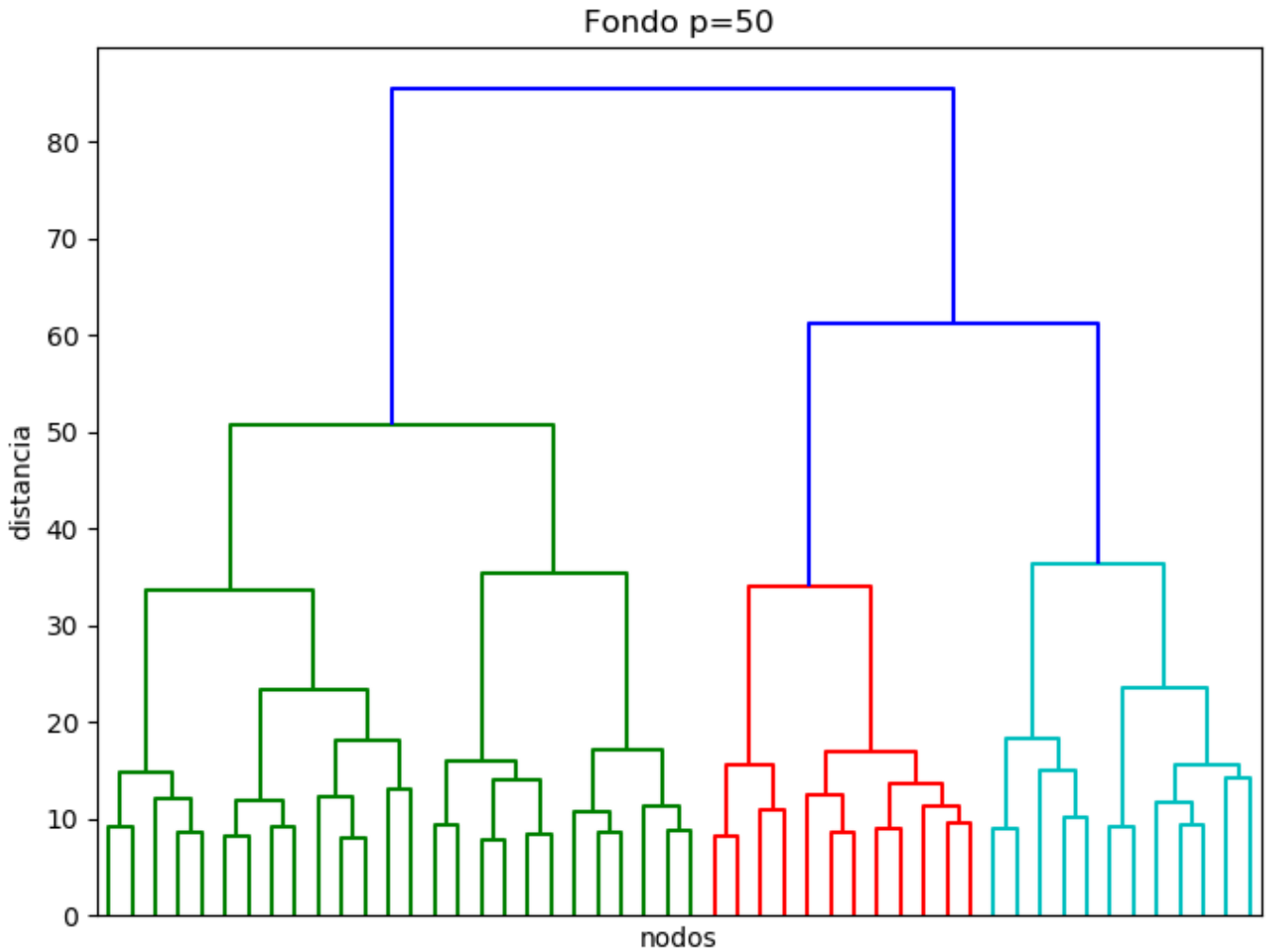


Actividades:

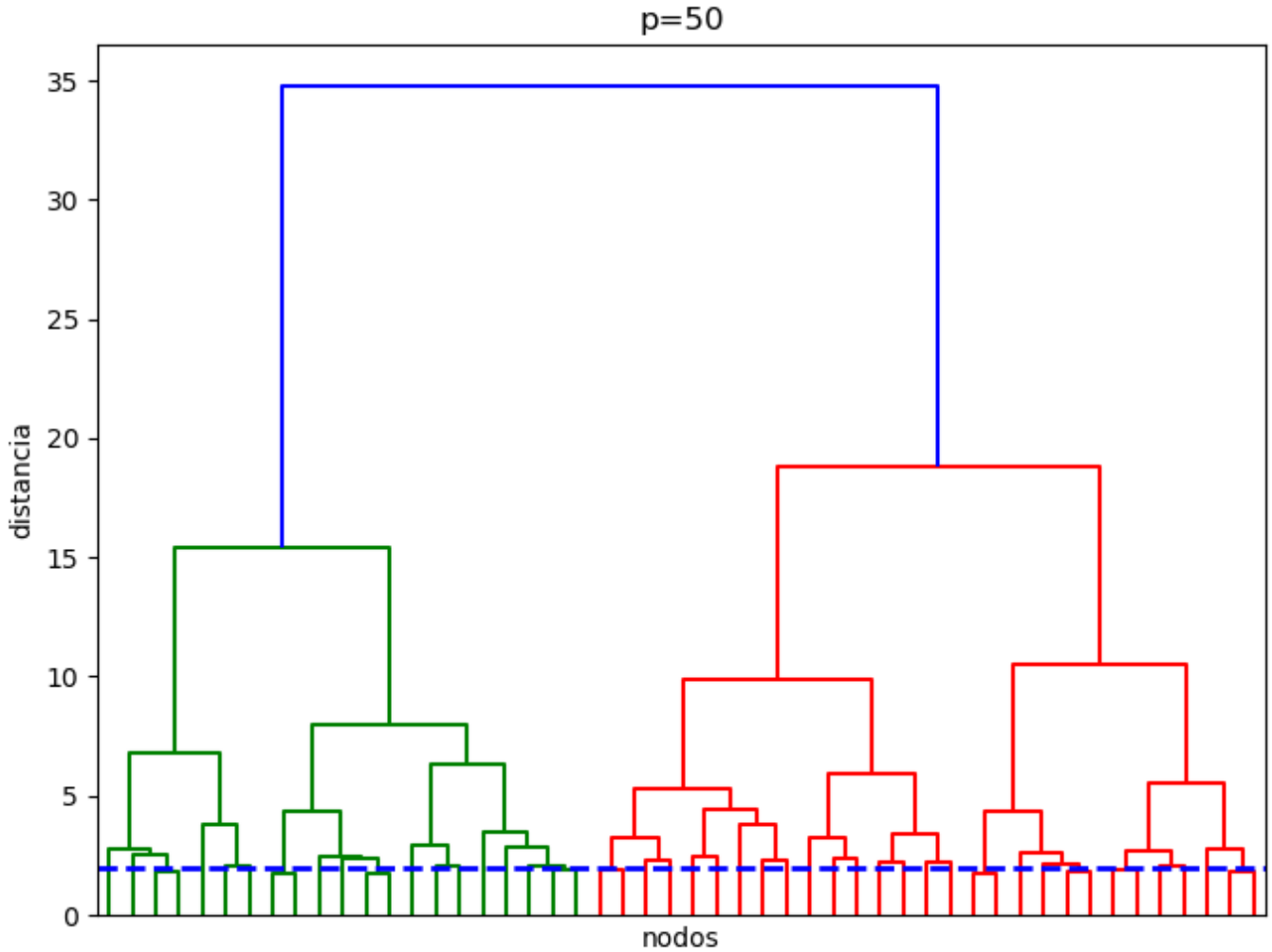
Comparación con el background para comparar el número de nodos:

```
In [3]: import scipy.cluster.hierarchy as shc

In [4]: pca0=pca.T

In [5]: vmin=np.min(pca0,axis=0)

In [6]: vmax=np.max(pca0,axis=0)

In [7]: pca1=np.zeros(np.shape(pca0))

In [8]: for ii in range(4):
   ...:     rr=np.random.random(34858)*(vmax[ii]-vmin[ii])+vmin[ii]
   ...:     pca1[:,ii]=rr
   ...:

In [9]: plt.figure(1)
Out[9]: <Figure size 800x600 with 0 Axes>

In [10]: fondo = shc.dendrogram(shc.linkage(pca1, method='ward'),truncate_mode='
    ...: lastp',p=50,no_labels=True)
```

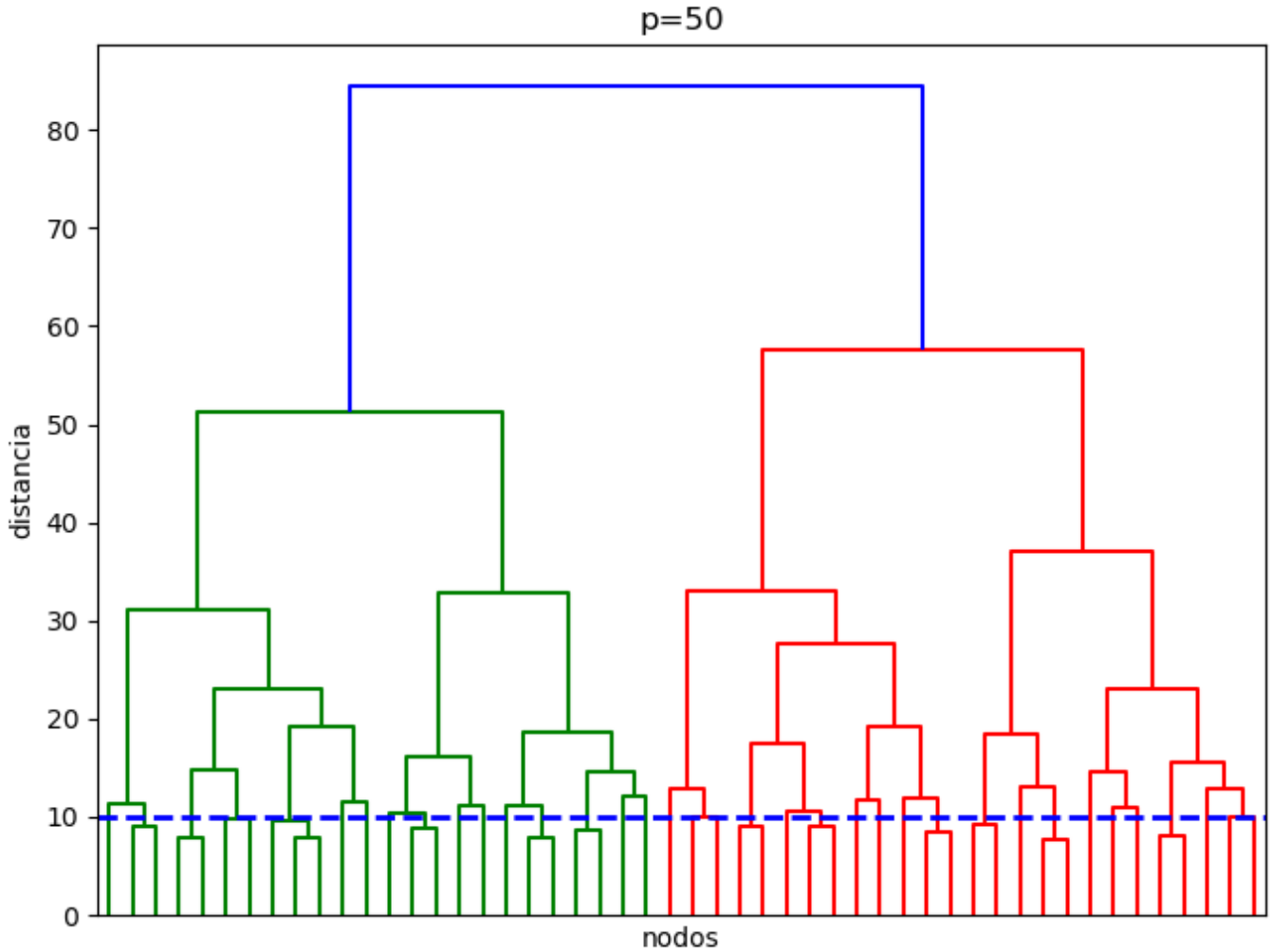
Actividades:



Actividades:



Actividades:



Actividades:

- Una distribución con una ley de potencia del tipo $\rho(> f) \propto f^{-\gamma}$ es una distribución acumulativa porque $\rho(> f)$ corresponde a todos los elementos con valores mayores o iguales a f.
- Supongamos una distribución de luminosidad con valores en el rango $[1, 10]$ y $\gamma = 2$. Voy a usar un histograma de 0,25 para cada bin.
- La constante de proporcionalidad se determina asumiendo que para $\rho(> 10) = 1$.

```
In [2]: lum=np.linspace(10,1,37)

In [3]: kk=1./10**(-2)

In [4]: rho=kk*lum**(-2)

In [5]: rho=rho/rho[-1]
```

Actividades:

- Ahora extraigo al azar las luminosidades de una población simulada de 20000 objetos. En el array **val** voy a guardar en cada columna luminosidad, distancia y flujo observado:

```
In [6]: rr=np.random.random(20000)

In [7]: val=np.zeros((len(rr),3))

In [8]: for ii in range(len(rr)):
   ...:     jj=0
   ...:     while(rr[ii] > rho[jj]):
   ...:         jj+=1
   ...:     val[ii,0]=lum[jj]+(np.random.random()-0.5)*0.25
   ...:
```

Actividades:

- Asumo un volumen total de 1000 unidades cúbicas, lo divido en 100 cáscaras de igual volumen y distribuyo los objetos con probabilidad uniforme:

```
In [9]: rad=np.zeros(100)

In [10]: rad[0]=10.**(1/3)

In [11]: for ii in range(1,100):
    ...:     rad[ii]=(10.+rad[ii-1]**3)**(1/3)
    ...:

In [12]: bb=np.random.randint(0,100,size=20000)

In [13]: for ii in range(20000):
    ...:     if(bb[ii] != 0):
    ...:         val[:,1]=(rad[bb]-rad[bb-1])*np.random.random()+rad[bb-1]
    ...:     else:
    ...:         val[:,1]=rad[bb]*np.random.random()
    ...:
```

Actividades:

- Calculo el flujo observable, fijo un límite inferior de detección y extraigo la muestra:

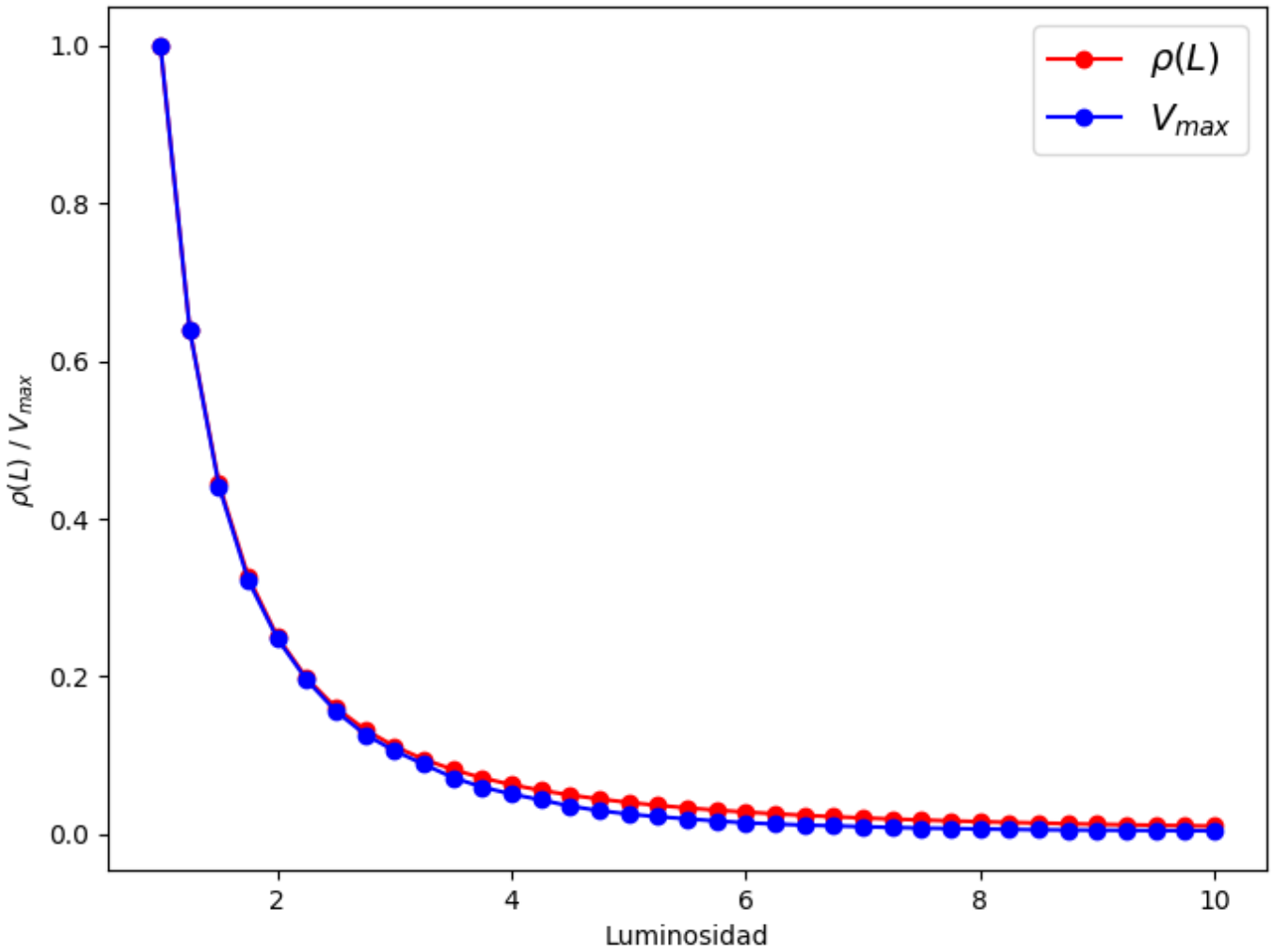
```
In [14]: val[:,2]=val[:,0]/val[:,1]**2

In [15]: np.min(val[:,2]),np.max(val[:,2])
Out[15]: (0.008825323755358806, 1.347497375858959)

In [16]: inx=np.where(val[:,2] > 0.05)

In [17]: np.shape(inx)
Out[17]: (1, 4315)

In [18]: mue=val[inx[0],:]
```

Actividades:

Aplicar el método V_{max} y recuperar la distribución utilizada:

```
In [3]: inx=np.where(val[:,2] > 0.05)

In [4]: np.shape(inx)
Out[4]: (1, 4238)

In [5]: mue=val[inx[0],:]

In [6]: lum0=lum-0.125

In [7]: bb=np.zeros(37)

In [8]: for ii in range(4238):
   ...:     jj=0
   ...:     while(mue[ii,0] < lum0[jj]):
   ...:         jj+=1
   ...:     vmax=(mue[ii,0]/0.05)**1.5
   ...:     bb[jj]+=1./vmax
   ...:

In [9]: acu=np.cumsum(bb)

In [10]: acu=acu/acu[-1]
```

Actividades:

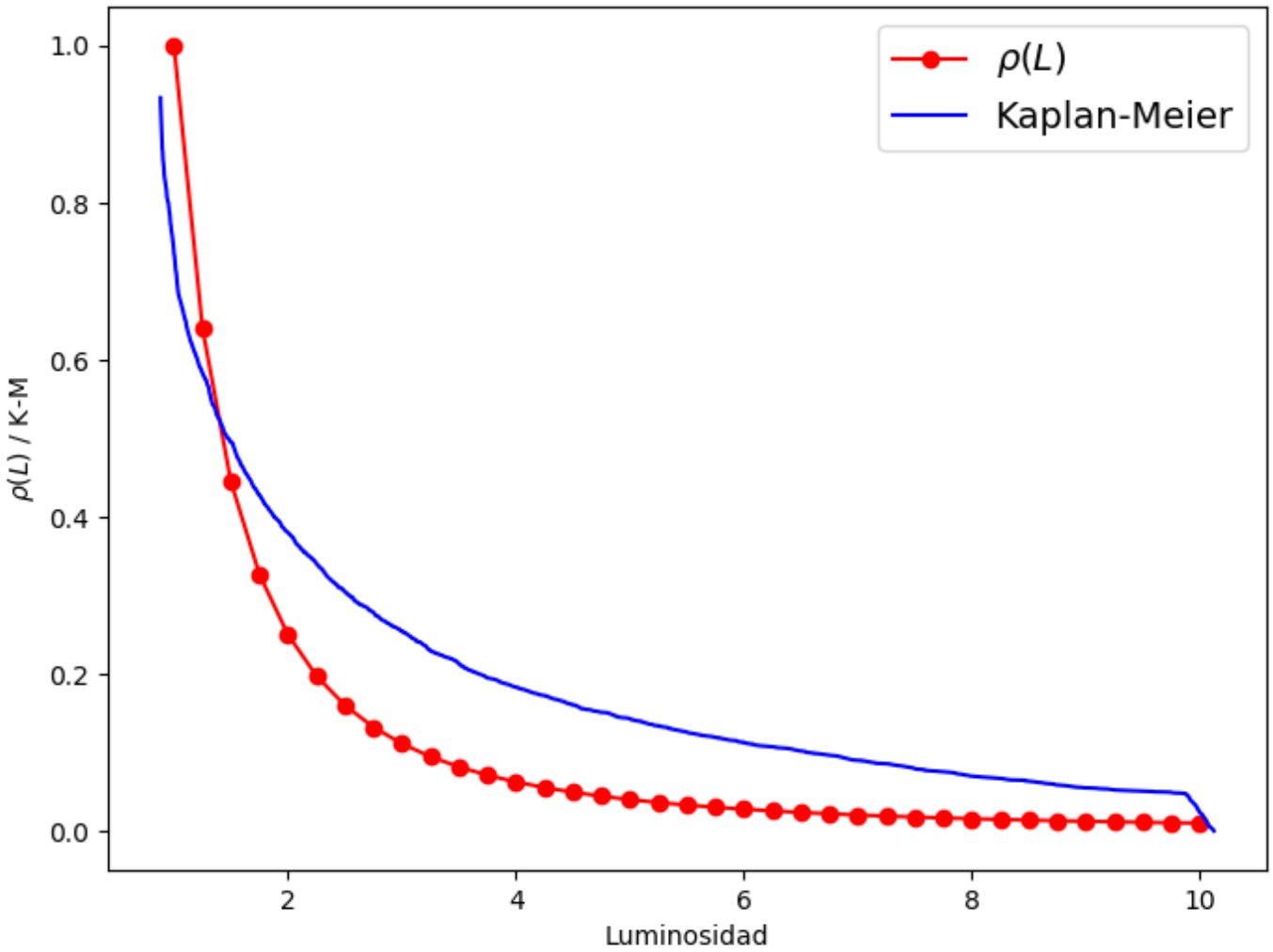


Actividades:

Asignar a cada objeto dos flujos diferentes, detecte y utilice el método de Kaplan-Meier para normalizar la distribución:

```
In [11]: flg=np.zeros(len(mue[:,0]))

In [12]: inx=np.where(mue[:,2] > 0.08)

In [13]: flg[inx[0]]=1

In [14]: inx=np.argsort(mue[:,0])

In [15]: hh=np.flip(mue[inx,0])

In [16]: ff=np.flip(flg[inx])

In [17]: ac=np.flip(np.arange(1,len(hh)+1,1))

In [18]: kap=[]

In [19]: ll=[]

In [20]: for ii in range(1,len(hh)):
    ...:     vv=1.
    ...:     if(ff[ii] == 1):
    ...:       for jj in range(ii):
    ...:           vv=vv*(1.-1./ac[jj])**(ff[jj])
    ...:       kap.append((1.-vv))
    ...:       ll.append(hh[ii])
    ...:
```

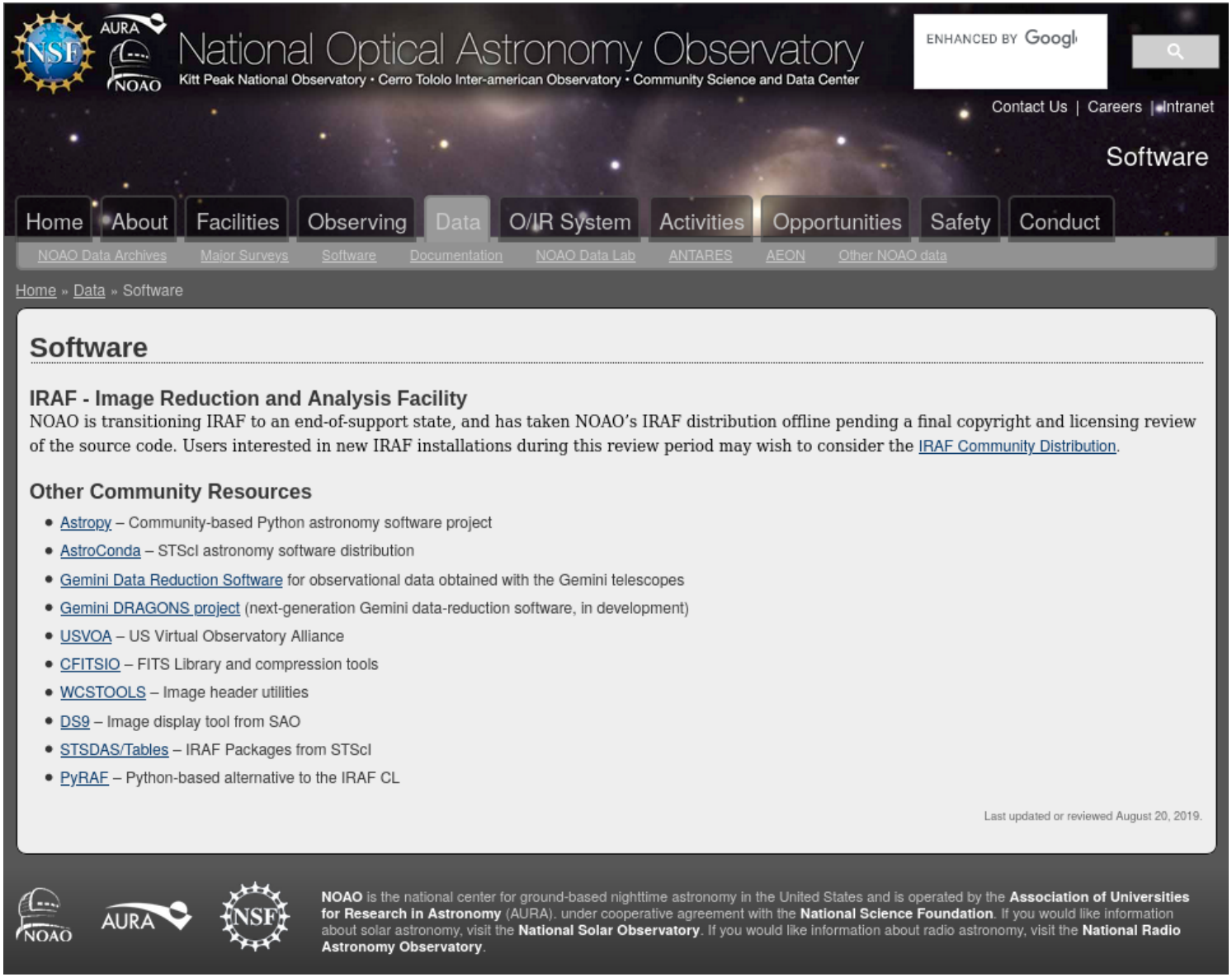
Actividades:



IRAF:

- La colección de programas IRAF (Image Reduction and Analysis Facility) fue creado en 1986 para procesar imágenes astronómicas.
- Para tener capacidad gráfica debía correr en una terminal gráfica llamada xgterm.
- Originalmente esplegaba imágenes en IMTOOL, pero podía usar diferentes herramientas como SAOImage y DS9.
- Dispone de cientos de paquetes específicos para diferentes tareas.
- La última versión disponible es la 2.16 de 2012.
- En 2013 ya estaban disponibles paquetes RPM y deb para Linux.

IRAF:

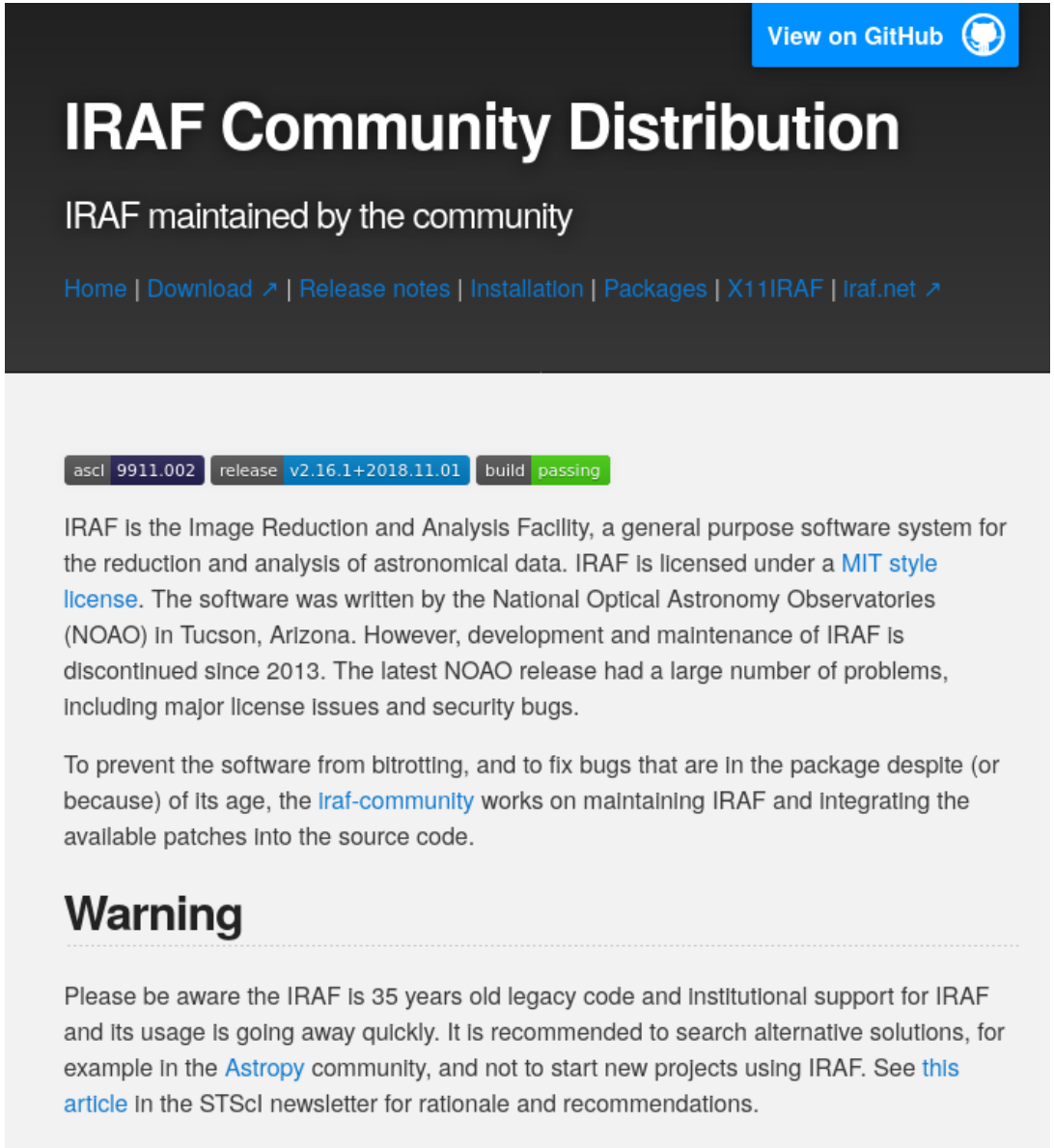

National Optical Astronomy Observatory
Kitt Peak National Observatory • Cerro Tololo Inter-american Observatory • Community Science and Data Center
ENHANCED BY Google
Contact Us | Careers | Intranet
Software
Home
About
Facilities
Observing
Data
O/IR System
Activities
Opportunities
Safety
Conduct
NOAO Data Archives
Major Surveys
Software
Documentation
NOAO Data Lab
ANTARES
AEON
Other NOAO data
Home » Data » Software
Software
IRAF - Image Reduction and Analysis Facility
NOAO is transitioning IRAF to an end-of-support state, and has taken NOAO's IRAF distribution offline pending a final copyright and licensing review of the source code. Users interested in new IRAF installations during this review period may wish to consider the IRAF Community Distribution.
Other Community Resources
Astropy – Community-based Python astronomy software project
AstroConda – STScI astronomy software distribution
Gemini Data Reduction Software for observational data obtained with the Gemini telescopes
Gemini DRAGONS project (next-generation Gemini data-reduction software, in development)
USVOA – US Virtual Observatory Alliance
CFITSIO – FITS Library and compression tools
WCSTOOLS – Image header utilities
DS9 – Image display tool from SAO
STSDAS/Tables – IRAF Packages from STScI
PyRAF – Python-based alternative to the IRAF CL
Last updated or reviewed August 20, 2019.
NOAO is the national center for ground-based nighttime astronomy in the United States and is operated by the Association of Universities for Research in Astronomy (AURA), under cooperative agreement with the National Science Foundation. If you would like information about solar astronomy, visit the National Solar Observatory. If you would like information about radio astronomy, visit the National Radio Astronomy Observatory.

IRAF:

- PyRAF es un paquete de **Python** que opera con programas de IRAF pero también fue discontinuado.
- Otras opciones son **MIDAS**, **IDL** y **C/C++**.
- En **Python** el paquete de procesamiento astronómico principal es astropy el cual incluye algunos paquetes asociados.
- El problema principal con astropy es la mala documentación, falta de tutoriales y la homogeneidad entre los diferentes sub-paquetes y paquetes asociados.
- Se pueden encontrar Jupyter Notebooks de ayuda para procesamiento de imágenes en **Python** en:

  https://github.com/spacetelescope/stak-notebooks

IRAF:



View on GitHub

IRAF Community Distribution

IRAF maintained by the community

Home | Download ↗ | Release notes | Installation | Packages | X11IRAF | Iraf.net ↗

ascl 9911.002 | release v2.16.1+2018.11.01 | build passing

IRAF is the Image Reduction and Analysis Facility, a general purpose software system for the reduction and analysis of astronomical data. IRAF is licensed under a MIT style license. The software was written by the National Optical Astronomy Observatories (NOAO) in Tucson, Arizona. However, development and maintenance of IRAF is discontinued since 2013. The latest NOAO release had a large number of problems, including major license issues and security bugs.

To prevent the software from bitrotting, and to fix bugs that are in the package despite (or because) of its age, the Iraf-community works on maintaining IRAF and integrating the available patches into the source code.

Warning

Please be aware the IRAF is 35 years old legacy code and institutional support for IRAF and its usage is going away quickly. It is recommended to search alternative solutions, for example in the Astropy community, and not to start new projects using IRAF. See this article in the STScI newsletter for rationale and recommendations.

Astropy:

Es una colección de paquetes escritos en **Python** y **C** para usar en astronomía. La última versión es la 4.0.1 (Abril 2020). Algunos paquetes disponibles:

- Constantes: astropy.constants
- Tablas de datos: astropy.table
- Datos n-dimensionales: astropy.nddata
- Coordenadas: astropy.coordinates
- Imágenes FITS: astropy.io.fits
- Calculos cosmológicos: astropy.cosmology
- Herramientas estadísticas: astropy.stats
- Observatorios virtuales: astropy.io.votable

Astropy:

No todos los sub-paquetes están suficientemente desarrollados:

The classification is as follows:

- Planned
- Actively developed, be prepared for possible significant changes.
- Reasonably stable, any significant changes/additions will generally include backwards-compatiblity.
- Mature. Additions/improvements possible, but no major changes planned.
- Pending deprecation. Might be deprecated in a future version.
- Deprecated. Might be removed in a future version.

The current planned and existing sub-packages are:

Sub-Package		Comments
astropy.config	●	Configuration received major overhaul in v0.4. Since then on, the package has been stable.
astropy.constants	●	The package has been stable except for the occasional additions of new constants. Since v3.0, it includes the ability to use sets of constants from previous versions.
astropy.convolution	●	New top-level package in v0.3 (was previously part of `astropy.nddata`). A major consistency improvement between fft/non-fft convolution, which is not fully backward-compatible, was added in 2.0.
astropy.coordinates	●	New in v0.2, major changes in v0.4. Subsequent versions should maintain a stable/backwards-compatible API, following the plan of APE 5. Further major additions/enhancements likely, but with basic framework unchanged.
astropy.cosmology	●	Incremental improvements since v0.1, stable API last several versions.
astropy.io.ascii	●	Originally developed as `asciitable`, and has maintained a stable API.
astropy.io.fits	●	Originally developed as `pyfits`, and retains an API consistent with the standalone version.
astropy.io.misc	●	The functionality that is currently present is stable, but this sub-package will likely see major additions in future.
astropy.io.votable	●	Originally developed as `vo.table`, and has a stable API.
astropy.modeling	●	New in v0.3. Major changes in v1.0, significant additions planned. Backwards-compatibility likely to be maintained, but not guaranteed.

Astropy:

No todos los sub-paquetes están suficientemente desarrollados:

astropy.nddata	🟡	Significantly revised in v1.0 to implement APE 7. Major changes in the API are not anticipated, broader use may reveal flaws that require API changes.
astropy.samp	🟢	Virtual Observatory service access: SAMP. This was renamed from astropy.vo.samp to astropy.samp in 2.0.
astropy.stats	🟡	Likely to maintain backwards-compatibility, but functionality continually being expanded, so significant additions likely in the future.
astropy.table	🟢	Incremental improvements since v0.1, mostly stable API with backwards compatibility an explicit goal.
astropy.time	🟢	Incremental improvements since v0.1, API likely to remain stable for the foreseeable future.
astropy.timeseries	🟡	New in v3.2, in heavy development.
astropy.units	🟢	Incremental improvements since v0.4. Functionality mature and unlikely to change. Efforts focused on performance and increased interoperability with Numpy functions.
astropy.utils	🟡	Contains mostly utilities destined for internal use with other parts of Astropy. Existing functionality generally stable, but regular additions and occasional changes.
astropy.uncertainty	🟡	New in v3.1, in development.
astropy.visualization	🟡	New in v1.0, and in development.
astropy.visualization.wcsaxes	🔵	New in v1.3. Originally developed as `wcsaxes` and has maintained a stable API.
astropy.wcs	🔵	Originally developed as `pywcs`, and has a stable API for now. However, there are plans to generalize the WCS interface to accommodate non-FITS WCS transformations, and this may lead to small changes in the user interface.

Astropy:

Una parte importante del proyecto son los paquetes coordinados y los paquetes afiliados, aunque no son parte integral de astropy. Los paquetes coordinados son mantenidos por el proyecto. Por ejemplo:

astroquery Website Repository PyPI

Tools for querying online astronomical data sources.

Maintainer(s): Adam Ginsburg and Brigitta Sipőcz

Functionality: General package | Astropy integration: Good | Docs: Good | Tests: Partial | Development: Good | Python 3: Yes

photutils Website Repository PyPI

Photometry and related image-processing tools.

Maintainer(s): Larry Bradley and Brigitta Sipőcz

Functionality: General package | Astropy integration: Good | Docs: Good | Tests: Good | Development: Good | Python 3: Yes

specutils Website Repository PyPI

Affiliated package for analysis tools and basic data types of astronomical spectra.

Maintainer(s): Nicholas Earl, Adam Ginsburg, Steve Crawford, and Erik Tollerud

Functionality: General package | Astropy integration: Good | Docs: Partial | Tests: Partial | Development: Heavy development | Python 3: Yes

Astropy:

Los paquetes afiliados integran el proyecto pero son mantenidos por otros grupos. Por ejemplo:

astroalign Website Repository PyPI

Astrometric registration of images when no WCS info is available

Maintainer(s): TOROS Dev Team

Functionality: Specialized package | Astropy integration: Partial | Docs: Good | Tests: Good | Development: Good | Python 3: Yes

astroML Website Repository PyPI

Tools for machine learning and data mining in Astronomy.

Maintainer(s): Jake Vanderplas and Brigitta Sipőcz

Functionality: General package | Astropy integration: Partial | Docs: Good | Tests: Partial | Development: Good | Python 3: Yes

ginga Website Repository PyPI

Ginga is a Python toolkit for building viewers for astronomical and scientific images stored in numpy data arrays and displaying them on a variety of different display platforms. The toolkit provides many features for building a viewer, including a flexible plugin framework that is provided as a customizable reference viewer for FITS images.

Maintainer(s): Eric Jeschke and Pey-Lian Lim

Functionality: General package | Astropy integration: Partial | Docs: Good | Tests: Partial | Development: Good | Python 3: Yes

statmorph Website Repository PyPI

A Python package for calculating non-parametric morphological diagnostics of galaxy images, as well as fitting 2D Sérsic profiles.

Maintainer(s): Vicente Rodriguez-Gomez

Functionality: Specialized package | Astropy integration: Partial | Docs: Partial | Tests: Partial | Development: Functional but low activity | Python 3: Yes

Astropy:

- Cuando se trabaja con astropy hay que tener cuidado con funciones y clases que en diferentes sub-paquetes tienen el mismo nombre pero operan de manera diferente.
- En particular, NUNCA importen algo de manera general como:

```
from astropy.io.fits import *  # NOT recommended
```

Siempre importen sub-paquetes individualmente:

```
from astropy.io import fits
hdulist = fits.open('data.fits')
```

```
from astropy import units as u
from astropy import coordinates as coord
```

Leer imágenes FITS:

```
import astropy.io.fits as ft

def leefits(arch,uint=True,plano=0):
    """
    Funcion que permite leer una imagen FITS y su cabecera desde su
archivo.

    Parameters
    ----------
    arch     : Archivo a leer. (string)
    uint     : Si la cabecera tiene los keywords BSCALE y BZERO puede
considerar la imagen como de enteros sin signo. El default es uint=True
(bool)
    plano    : En un cubo multidimensional indica cual es el plano a
leer. El default es plano=0. (int)

    Returns
    -------
    img      : El array con la imagen.
    hdr      : La cabecera.
    """
    f=ft.open(arch,uint=uint)

# esto es para evitar problemas con keywords no standard en la cabecera
#
    f.verify('silentfix')

    img=f[0].data
    hdr=f[0].header
    f.close()
    return img,hdr
```

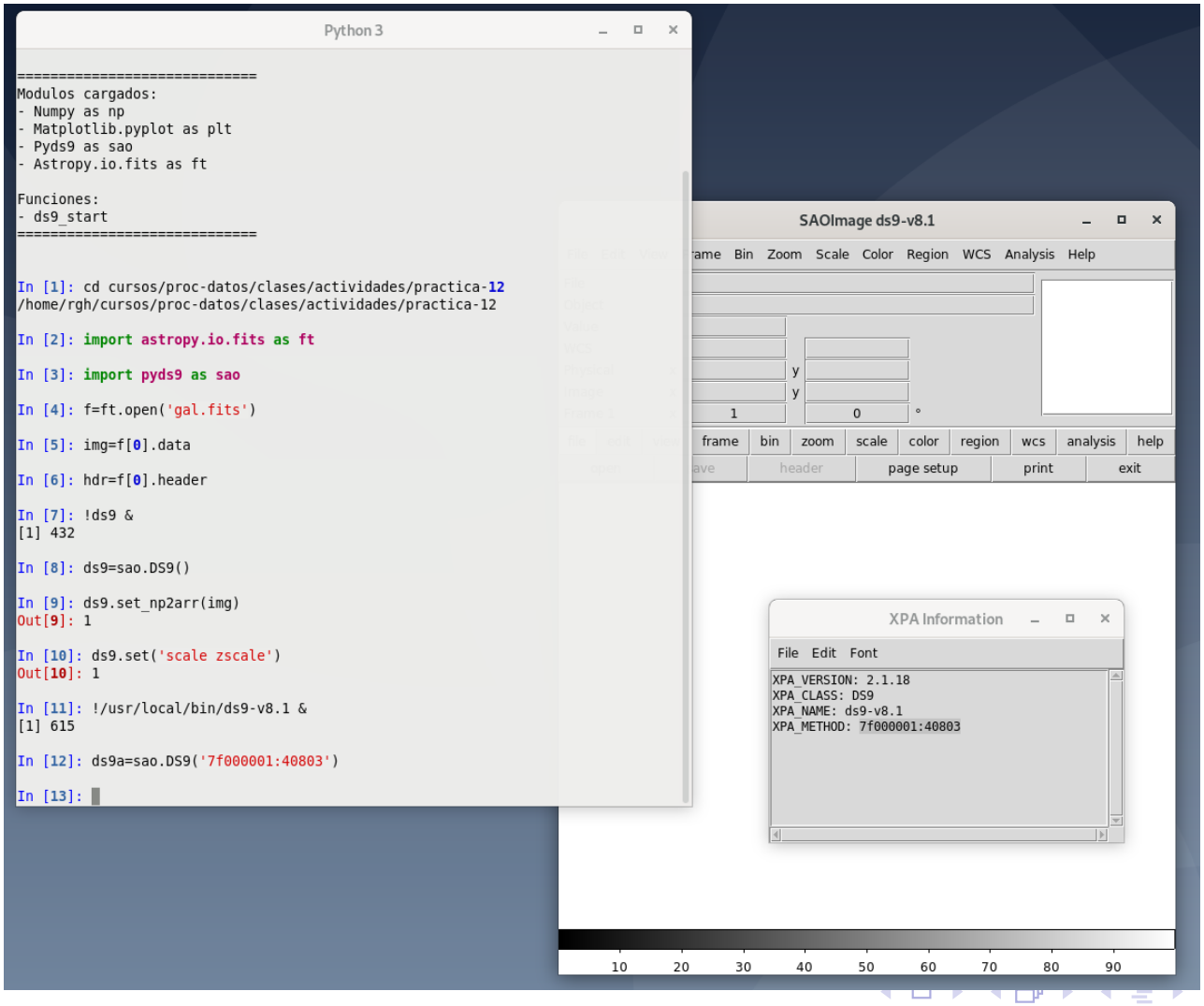
Usando SAOImage DS9:

- Hay varios paquetes que permiten utilizar SAOImage DS9 como el visualizador standard de imágenes en **Python**.
- El más utilizado es pyds9 que se conecta con el DS9 mediante el intercambio de mensajes XPA.

```
In [1]: cd cursos/proc-datos/clases/actividades/practica-12
/home/rgh/cursos/proc-datos/clases/actividades/practica-12

In [2]: import astropy.io.fits as ft

In [3]: import pyds9 as sao

In [4]: !ds9 &
[1] 31533

In [5]: f=ft.open('gal.fits')

In [6]: img=f[0].data

In [7]: hdr=f[0].header

In [8]: ds9=sao.DS9()

In [9]: ds9.set_np2arr(img)
Out[9]: 1

In [10]: ds9.set('scale zscale')
Out[10]: 1
```

Usando SAOImage DS9:

Pyds9 permite usar más de un SAOImage DS9 usando el código en File $->$ XPA $->$ Information...:



Usando SAOImage DS9:

Pyds9 tiene 4 funciones principales para operar sobre SAOImage DS9:

- pyds9.set_np2arr() lee un array y lo despliega en DS9.
- pyds9.get_np2arr() lee la imagen desplegada en DS9 y la copia en un array.
- pyds9.set() envía un comando XPA a DS9.
- pyds9.get() obtiene el valor devuelto por un comando XPA operando sobre DS9.

Recuerde que SAOImage DS9 numera los pixels desde (1,1) comenzando desde el rincón inferior izquierdo.
Un listado de los mensajes XPA aceptados se encuentra en http://ds9.si.edu/doc/ref/xpa.html

Usando SAOImage DS9:

Ejemplos:

- pyds9.set('export jpeg prueba.jpg 75') guarda la imagen desplegada como jpg.
- pyds9.set('frame 2') cambia al segundo frame.
- pyds9.set('frame center') centra la imagen en el frame.
- pyds9.set('frame next') y pyds9.set('frame prev') cambian de frame secuencialmente.
- pyds9.set('header save 1 header.txt') guarda en un archivo el header de la imagen desplegada en el frame 1.
- pyds9.set('orient x'), pyds9.set('orient y') y pyds9.set('orient xy') invierte la imagen en la dirección indicada.

Usando SAOImage DS9:

- pyds9.set('scale zscale') ajusta la paleta a un cierto modo.
- pyds9.set('cmap Heat') cambia la paleta de colores.
- pyds9.set('cmap invert yes') invierte la paleta de colores.
- pyds9.set('colorbar yes') y pyds9.set('colorbar no') muestra o no la barra de colores.
- pyds9.set('blink') y pyds9.set('single') inicia o para el blinkeo de imágenes.
- pyds9.set('regions command {circle 300 220 20 # color=blue}') dibuja un círculo azul de radio 20 px en (X,Y)=(300,220).
- pyds9.set('regions command {box 300 220 80 60 # color=cyan width=3}') dibuja un rectángulo cyan de 80 x 60 en (X,Y)=(300,220).

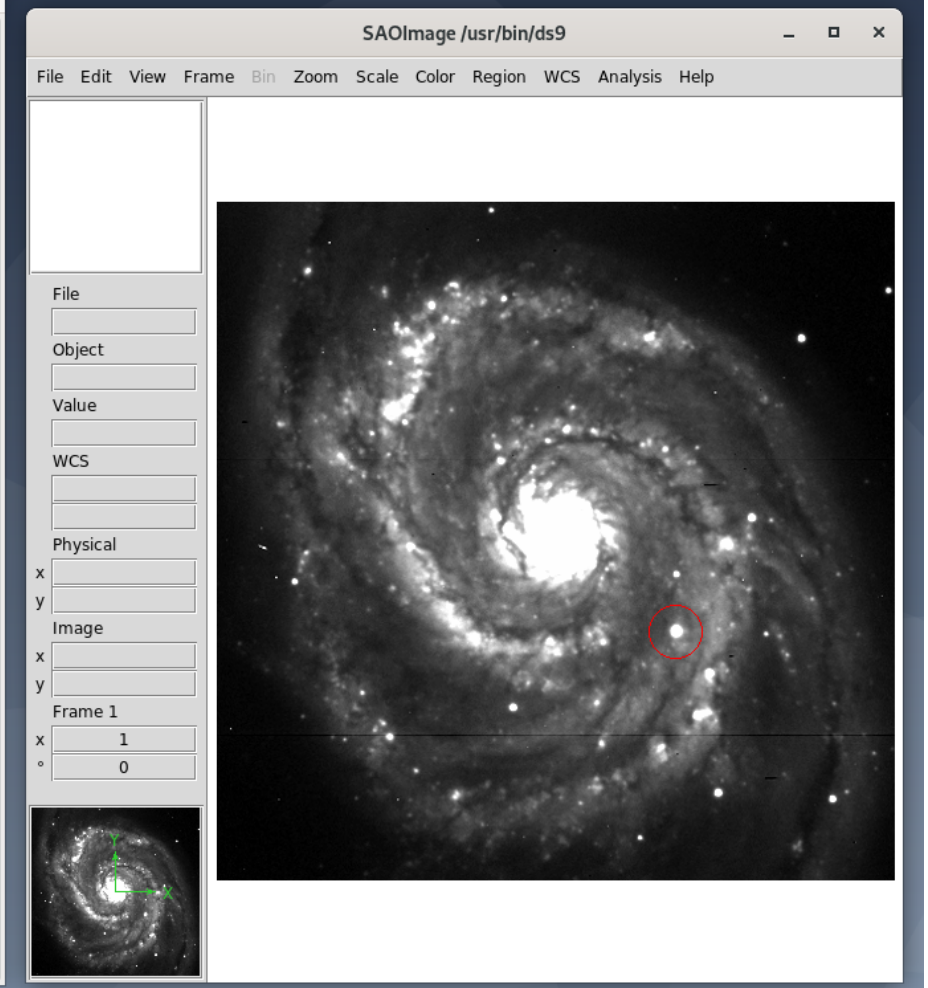
Usando SAOImage DS9:

- kkk=pyds9.get("fits header keyword 'object'") obtiene el valor del keyword 'object' de la cabecera.
- aaa=pyds9.get('data physical 250 250 50 50 yes') devuelve un bloque de datos de 50 x 50 desde (X,Y)=(250,250).
- coo=pyds9.get('imexam coordinate image') devuelve coordenadas donde se presiona el mouse.
- coo=pyds9.get('imexam key coordinate image') devuelve coordenadas donde está el mouse y la tecla presionada.
- val=pyds9.get('imexam data') devuelve valor del pixel donde se presiona el mouse.
- val=pyds9.get('imexam key data') devuelve valor del pixel donde se presiona una tecla.

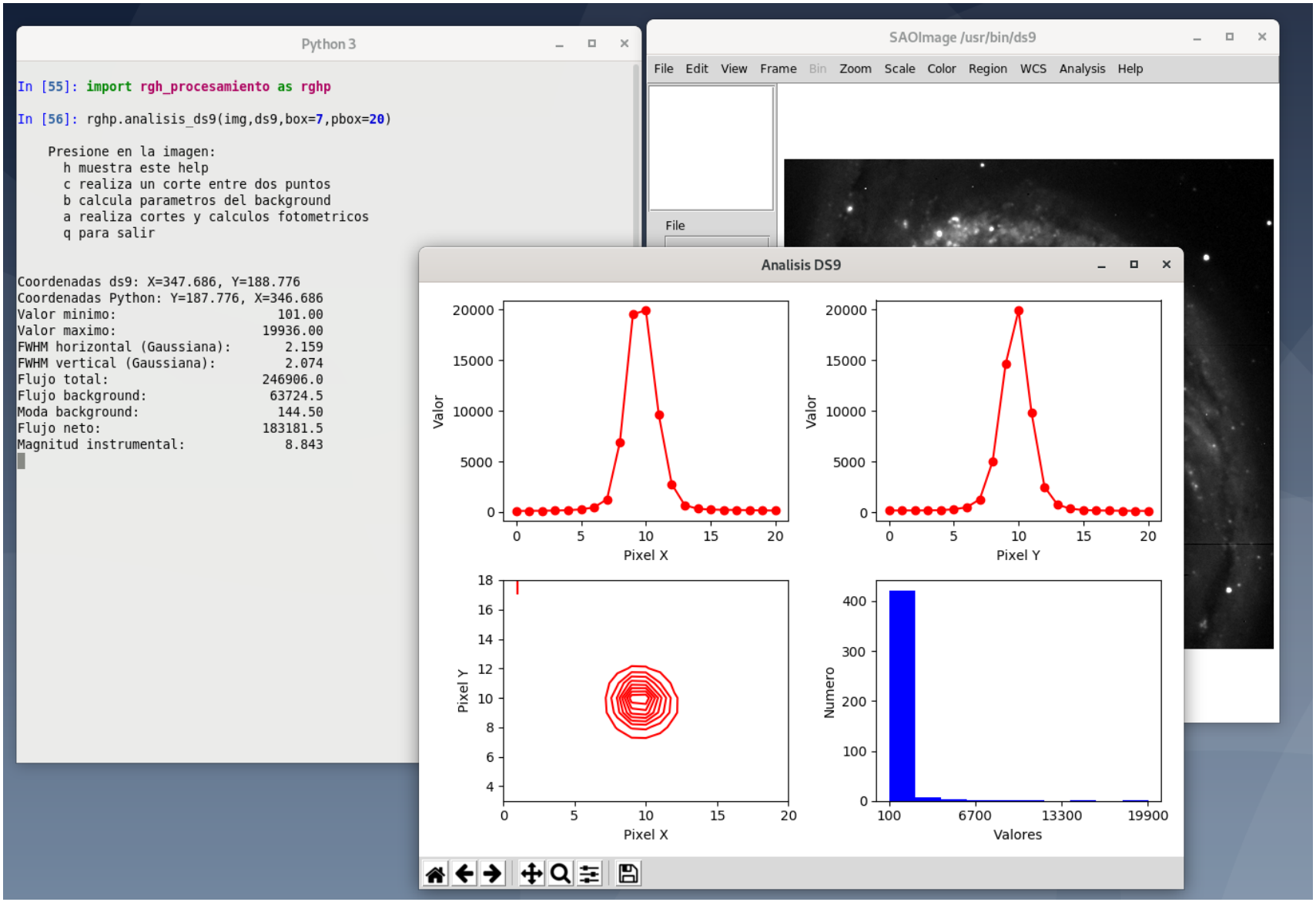
Usando SAOImage DS9:

Python 3

```
In [42]: ds9.set_np2arr(img)
Out[42]: 1

In [43]: pos=ds9.get('imexam key coordinate image')

In [44]: pos
Out[44]: 'a 347 188'

In [45]: coo=pos.split()

In [46]: cx=int(coo[1])-1

In [47]: cy=int(coo[2])-1

In [48]: val=img[cy,cx]

In [49]: print('El valor en el pixel({:d},{:d}) es: {:f}'.format(cy,cx,val))
El valor en el pixel(187,346) es: 13988.000000

In [50]: lin="regions command {circle "+coo[1]+" "+coo[2]+" 20 # color=red}"

In [51]: ds9.set(lin)
Out[51]: 1

In [52]:
```



Usando SAOImage DS9:



Actividades:

Trabajo final del curso:

Busque algún paper no reciente de su especialidad donde se realice un análisis y/o un procesamiento de datos.
Trate de repetir el proceso con el mismo conjunto de datos que usaron los autores o alguno similar que tengan. Comenten las diferencias entre sus resultados y los que obtuvieron los autores.

Entrega

Última quincena de Noviembre 2020

Por consultas: ricardo.gil-hutton@conicet.gov.ar
Grupo de Ciencias Planetarias - CUIM 2