```python
# !pip install vorbin

from mpdaf.obj import Cube
from spectral_cube import SpectralCube
from astropy.io import fits
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import transforms
from astropy.stats import sigma_clip, mad_std
import pandas as pd
from gofish import imagecube
import astropy.units as u
import matplotlib as mpl
from matplotlib.ticker import LogLocator, LogFormatter,
ScalarFormatter, LogFormatterSciNotation
import vorbin
from vorbin.voronoi_2d_binning import voronoi_2d_binning
from scipy.optimize import curve_fit
from astropy.visualization import AsinhStretch, ImageNormalize
from astropy.visualization import simple_norm
import math
from astropy.cosmology import Planck18 as cosmo
import extinction

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import transforms
from matplotlib.ticker import ScalarFormatter
import astropy.units as u
from spectral_cube import SpectralCube
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.wcs import WCS
from matplotlib.ticker import ScalarFormatter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

hdu = fits.open("teacup.fits")
hdu.info()
print(repr(hdu[1].header))
```

```
Filename: teacup.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1546   ()
  1  DATA          1 ImageHDU        43   (322, 318, 3802)   float32

  2  STAT          1 ImageHDU        43   (322, 318, 3802)   float32
```

```
XTENSION= 'IMAGE   '           / IMAGE extension

BITPIX  =                  -32 / number of bits per data pixel

NAXIS   =                    3 / number of data axes

NAXIS1  =                  322 / length of data axis 1

NAXIS2  =                  318 / length of data axis 2

NAXIS3  =                 3802 / length of data axis 3

PCOUNT  =                    0 / required keyword; must = 0

GCOUNT  =                    1 / required keyword; must = 1

EXTNAME = 'DATA    '           / This extension contains data values

DATASUM = '1605300420'         / data unit checksum updated 2020-03-
10T12:23:59
HDUCLASS= 'ESO     '           / class name (ESO format)

HDUDOC  = 'DICD    '           / document with class description

HDUVERS = 'DICD version 6'     / version number (according to spec
v2.5.1)
HDUCLAS1= 'IMAGE   '           / Image data format

HDUCLAS2= 'DATA    '           / this extension contains the data
itself
ERRDATA = 'STAT    '           / pointer to the variance extension

OBJECT  = 'Teacup (DATA)'

BUNIT   = '10**(-20)*erg/s/cm**2/Angstrom'

CRPIX1  =     163.938487646673 / Pixel coordinate of reference point

CRPIX2  =     155.344040098697 / Pixel coordinate of reference point

CD1_1   = -5.55555555555556E-05 / Coordinate transformation matrix
element
CD1_2   =                   0. / Coordinate transformation matrix
element
CD2_1   =                   0. / Coordinate transformation matrix
element
CD2_2   = 5.55555555555556E-05 / Coordinate transformation matrix
element
CUNIT1  = 'deg     '           / Units of coordinate increment and
value
```

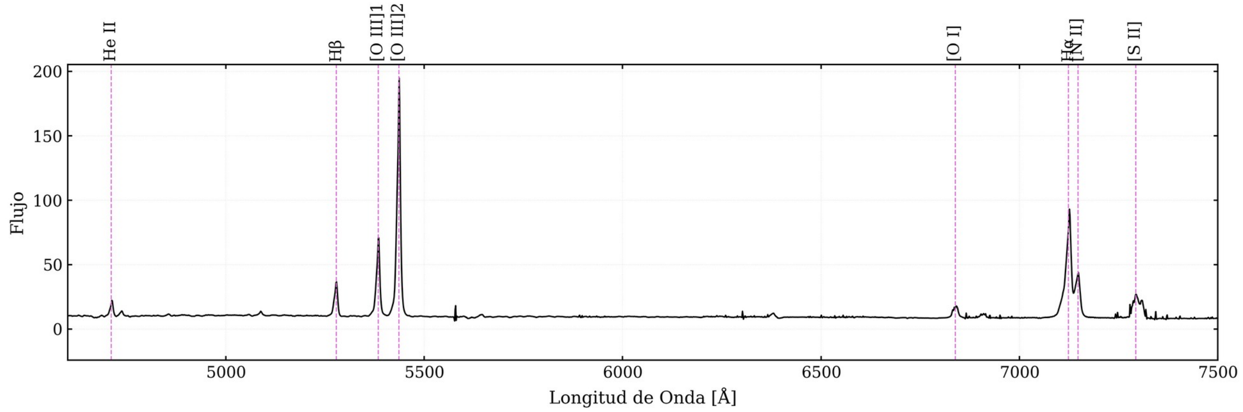
```
CUNIT2  = 'deg     '           / Units of coordinate increment and
value
CTYPE1  = 'RA---TAN'           / Right ascension, gnomonic projection

CTYPE2  = 'DEC--TAN'           / Declination, gnomonic projection

CSYER1  =     1.3680091695E-05 / [deg] Systematic error in coordinate

CSYER2  =    7.77995869722E-06 / [deg] Systematic error in coordinate

CRVAL1  =           217.624397

CRVAL2  =             13.65424

CTYPE3  = 'AWAV    '

CUNIT3  = 'Angstrom'

CD3_3   =                 1.25

CRPIX3  =                   1.

CRVAL3  =     4600.28173828125

CRDER3  =                0.026 / [Angstrom] Random error in spectral
coordinate
CD1_3   =                   0.

CD2_3   =                   0.

CD3_1   =                   0.

CD3_2   =                   0.

TITLE   = 'Teacup_2081147_2019-03-03T08:07:29.370_WFM-NOAO-E_OBJ'
```

```
cube = hdu[1].data
header = hdu[1].header
```

```
print(cube.shape)
```

```
(3802, 318, 322)
```

```
df = pd.read_csv("teacup.fits-Z-profile-galax_true.tsv",
                 sep="\t")

fig, ax = plt.subplots(figsize=(12,4), constrained_layout=True)

ax.plot(df["x"], df["y"], color='black', lw=1)
ax.set(xlabel=r'Longitud de Onda [Å]',
       ylabel=r'Flujo',
      xlim=(df["x"].min(), 7500))

trans = transforms.blended_transform_factory(ax.transData,
ax.transAxes)
def vline_outside(ax, x, label, color='orchid', dy_axes=0.01, rot=90):
    ax.axvline(x=x, color=color, ls='--', lw=0.8)
    ax.text(x, 1.0 + dy_axes, label,
            transform=trans, rotation=rot,
            ha='center', va='bottom', clip_on=False)

# Líneas de emisión
vline_outside(ax, 4711, 'He II', )
vline_outside(ax, 5278, 'Hβ',    )
vline_outside(ax, 5384, '[O III]1', )
vline_outside(ax, 5436, '[O III]2', )
vline_outside(ax, 6838, '[O I]', )
vline_outside(ax, 7123, 'Hα',    )
vline_outside(ax, 7147, '[N II]', )
vline_outside(ax, 7293, '[S II]', )

ax.xaxis.set_major_formatter(ScalarFormatter(useMathText=True))
fig.subplots_adjust(top=0.86)
plt.show()
```



```
rest = u.AA * np.array([4861.33, 4958.91, 5006.84, 6300.30, 6562.80,
6583.45, 6716.44])
obs  = u.AA * np.array([5278.00, 5384.00, 5436.00, 6838.00, 7123.00,
7147.00, 7293.00])

z = (obs/rest - 1).value
zc = sigma_clip(z, sigma=2).compressed()
z_med, z_err = np.median(zc), mad_std(zc)

A = np.vstack([rest.value, np.ones_like(rest.value)]).T
slope, offset = np.linalg.lstsq(A, obs.value, rcond=None)[0]
z_fit = slope - 1
```

```
print(f"z = {z_med:.5f} ± {z_err:.5f}")
```

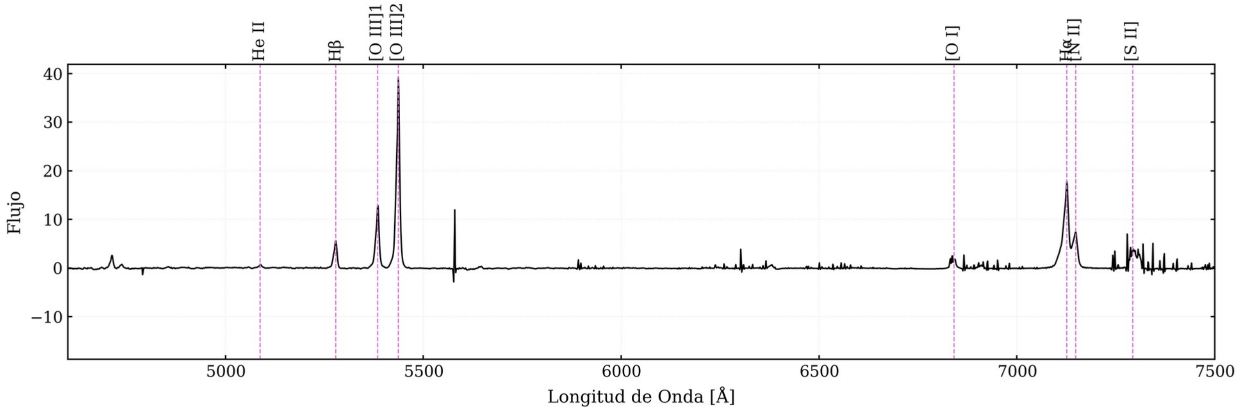
```
z = 0.08571 ± 0.00001
```

```
cube = Cube(filename='teacup.fits',ext=(1,2))

# cube[:,150,160].plot(unit=u.angstrom)

# cube[:,156,155].plot(unit=u.angstrom,lmin=5405,lmax=5460) # OIII
# cube[:,156,155].plot(unit=u.angstrom,lmin=5005,lmax=6000)

# OIII =
cube[:,156,155].gauss_fit(unit=u.angstrom,lmin=5405,lmax=5460,
plot=True)
# OIII.print_param()

# 5278
# cube[:,156,155].plot(unit=u.angstrom,lmin=5265,lmax=5290) # Hb

# Hb = cube[:,156,155].gauss_fit(unit=u.angstrom,lmin=5265,lmax=5290,
plot=True)
# Hb.print_param()

# cube[:,156,155].plot(unit=u.angstrom,lmin=7085,lmax=7170) # Ha y NII
```

Correcciones

Resta del Continuo

```
# cube_rest = SpectralCube.read('teacup.fits',
hdu=1).with_spectral_unit(u.AA)
# lam  = cube_rest.spectral_axis.to_value(u.AA).astype(np.float32)
# N, Ny, Nx = cube_rest.shape

# m = (lam >= 5708) & (lam <= 6758)
# k = int(m.sum())
# Xg = np.vstack([np.ones(k, dtype=np.float32), lam[m]]).T   # [k x 2]
# X  = np.vstack([np.ones(N, dtype=np.float32), lam]).T      # [N x 2]

# data_sub = np.empty((N, Ny, Nx), dtype=np.float32)

# ty, tx = 64, 64

# for y0 in range(0, Ny, ty):
#     y1 = min(Ny, y0+ty)
#     for x0 in range(0, Nx, tx):
#         x1 = min(Nx, x0+tx)
#         blk = cube_rest.filled_data[:, y0:y1,
x0:x1].value.astype(np.float32)
#         Dg = blk[m].reshape(k, -1)
#         ok = np.all(np.isfinite(Dg), axis=0)
#         cont_blk = np.full((N, Dg.shape[1]), np.nan,
dtype=np.float32)

#         if np.any(ok):
#             coef, *_ = np.linalg.lstsq(Xg, Dg[:, ok], rcond=None)
#             cont_blk[:, ok] = X @ coef

#         sub_blk = blk.reshape(N, -1) - cont_blk
#         data_sub[:, y0:y1, x0:x1] = sub_blk.reshape(N, y1-y0, x1-x0)

# cube_sub = SpectralCube(data_sub * cube_rest.unit,
wcs=cube_rest.wcs)
# cube_sub.write('teacup_continuum_sub.fits', overwrite=True)

# cube[:,156,150].plot(unit=u.angstrom,lmin=5000,lmax=7400)

# cubee = Cube(filename='teacup_continuum_sub.fits',ext=(1,2))
# cubee[:,156,150].plot(unit=u.angstrom,lmin=5000,lmax=7400)

cube2 = SpectralCube.read("teacup_continuum_sub.fits", hdu=1)

wave = cube2.spectral_axis.to(u.AA).value
nchan = cube2.shape[0]

do_sum = False

flux = np.empty(nchan, dtype=float)
ny, nx = cube2.shape[1], cube2.shape[2]

for i in range(nchan):
    sli = cube2.filled_data[i, :, :]
    arr = sli.value
    if do_sum:
        flux[i] = np.nansum(arr)
    else:
        flux[i] = np.nanmean(arr)

fig, ax = plt.subplots(figsize=(12, 4), constrained_layout=True)
ax.plot(wave, flux, color='black', lw=1)

ax.set(
    xlabel=r'Longitud de Onda [Å]',
    ylabel=('Flujo (suma)' if do_sum else 'Flujo'),
    xlim=(wave.min(), 7500)
)
trans = transforms.blended_transform_factory(ax.transData,
ax.transAxes)
def vline_outside(ax, x, label, color='orchid', dy_axes=0.01, rot=90):
    ax.axvline(x=x, color=color, ls='--', lw=0.8)
    ax.text(x, 1.0 + dy_axes, label, transform=trans, rotation=rot,
            ha='center', va='bottom', clip_on=False)

z = 0.08571
rest = {
    "He II": 4685.7,
    "Hβ": 4861.33,
    "[O III]1": 4958.92,
    "[O III]2": 5006.84,
    "[O I]": 6300.30,
    "Hα": 6562.80,
    "[N II]": 6583.45,
    "[S II]": 6716.44,
}
for lab, lam0 in rest.items():
    vline_outside(ax, lam0*(1+z), lab)

ax.xaxis.set_major_formatter(ScalarFormatter(useMathText=True))
plt.show()
```



Momentos

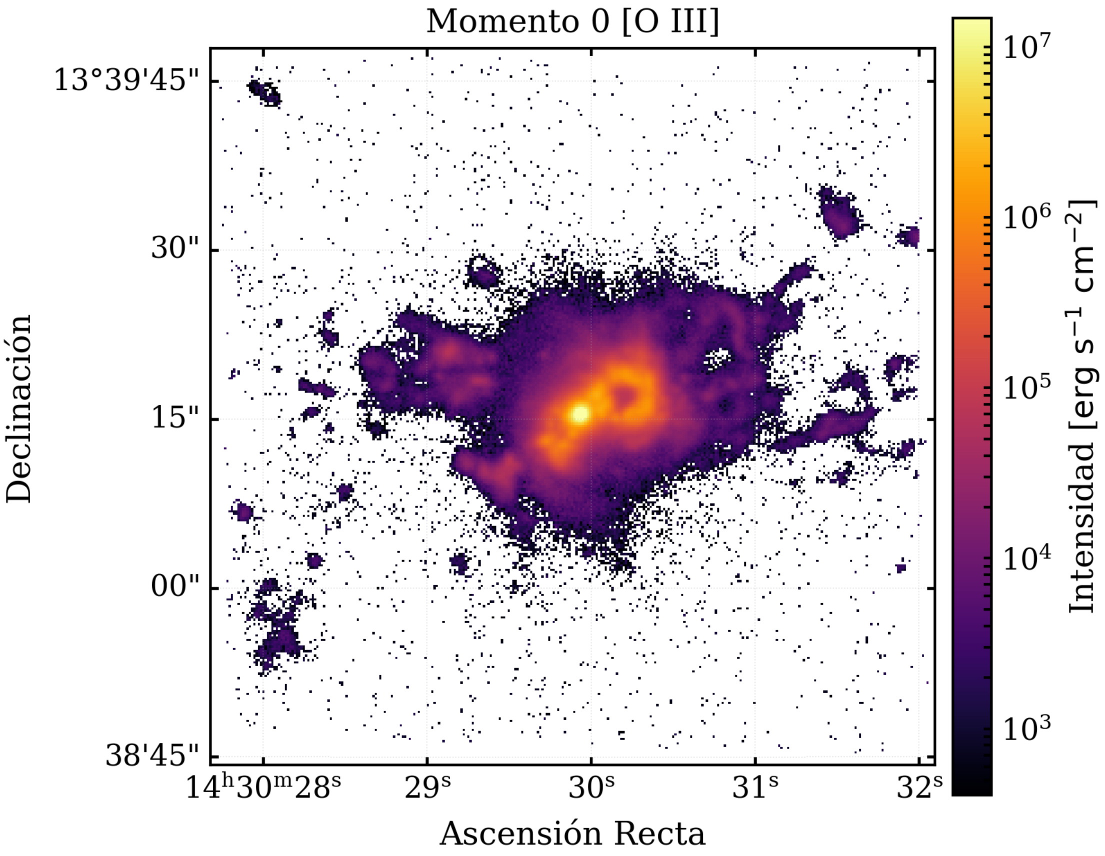
```
fits.open('teacup_continuum_sub.fits').info()
```

```
Filename: teacup_continuum_sub.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU      31   (322, 318, 3802)   float32
```

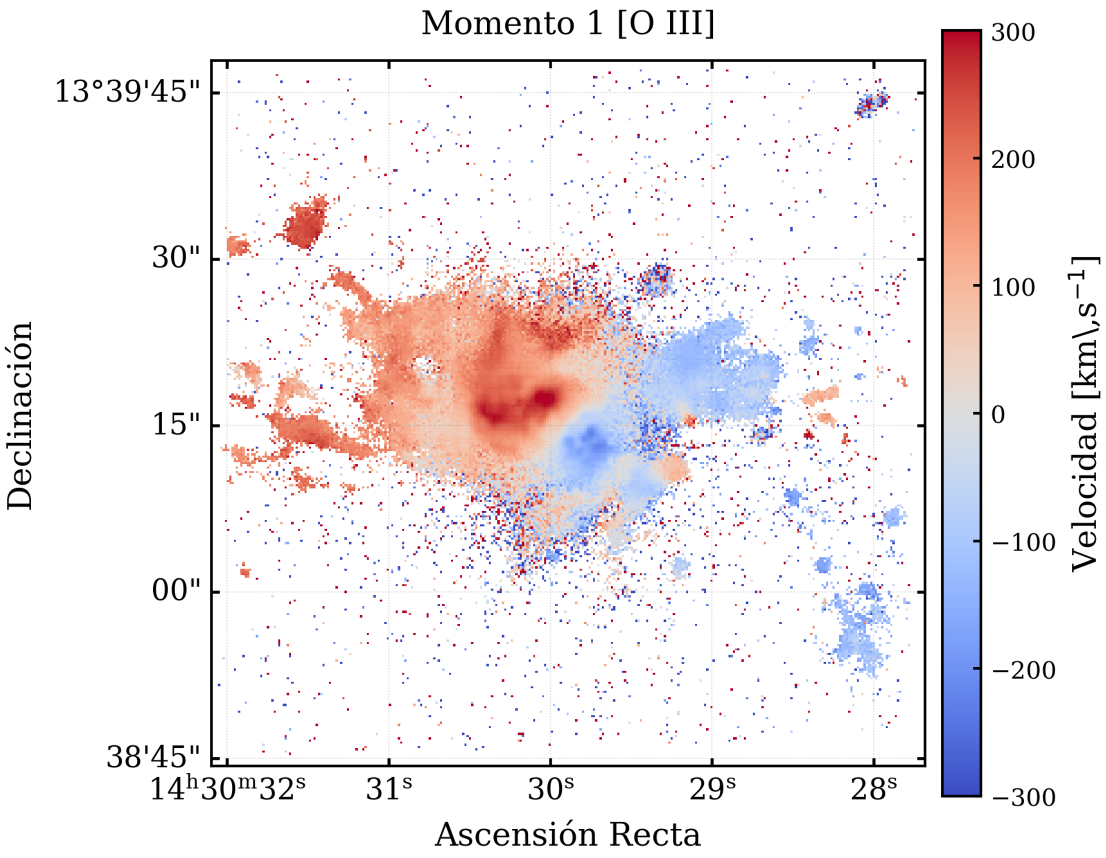
```
fits.open('teacup.fits').info()
```

```
Filename: teacup.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1546   ()
  1  DATA          1 ImageHDU        43   (322, 318, 3802)   float32

  2  STAT          1 ImageHDU        43   (322, 318, 3802)   float32
```

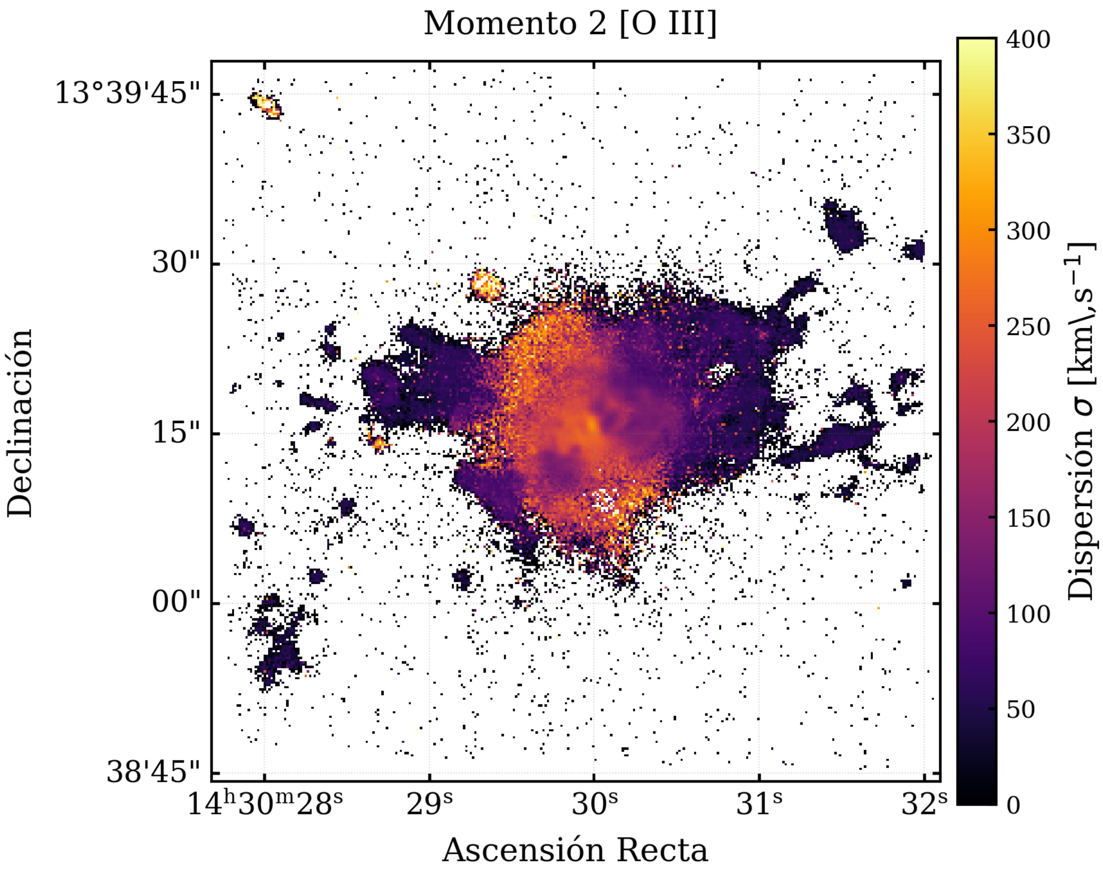
```python
# cube2 = SpectralCube.read('teacup_continuum_sub.fits',hdu=1)

OIII_cube = fits.open('teacup.fits')
cube2_sn =
SpectralCube(data=OIII_cube[1].data/np.sqrt(OIII_cube[2].data),wcs=cub
e2.wcs)

cube2_masked = cube2.with_mask(cube2_sn>3.5)

OIII_cube_masked =
cube2_masked.with_spectral_unit(u.km/u.s,velocity_convention='optical'
,rest_value=5433.61*u.AA)
OIII_subcube_masked = OIII_cube_masked.spectral_slab(-
600*u.km/u.s,600*u.km/u.s)

from matplotlib.colors import LogNorm
from matplotlib.ticker import LogFormatterMathtext

moment_0_masked = OIII_subcube_masked.moment(order=0)

fig = plt.figure(figsize=(6,5))
ax = plt.subplot(projection=cube2.wcs.celestial)
data = moment_0_masked.value
vmin = np.nanpercentile(data[data > 0], 1)
vmax = np.nanpercentile(data, 99.9)

im = ax.imshow(data, origin='lower', cmap='inferno',
               norm=LogNorm(vmin=vmin, vmax=vmax))
cbar = plt.colorbar(im, ax=ax, pad=0.02)
cbar.set_label(r'Intensidad [$\mathrm{erg\ s^{-1}\ cm^{-2}}$]',
fontsize=12)
ax.set_xlabel(r'Ascensión Recta', fontsize=12)
ax.set_ylabel(r'Declinación', fontsize=12)
ax.set_title(r'Momento 0 [O III]', fontsize=12)
ax.invert_xaxis()
plt.tight_layout()
plt.show()
```



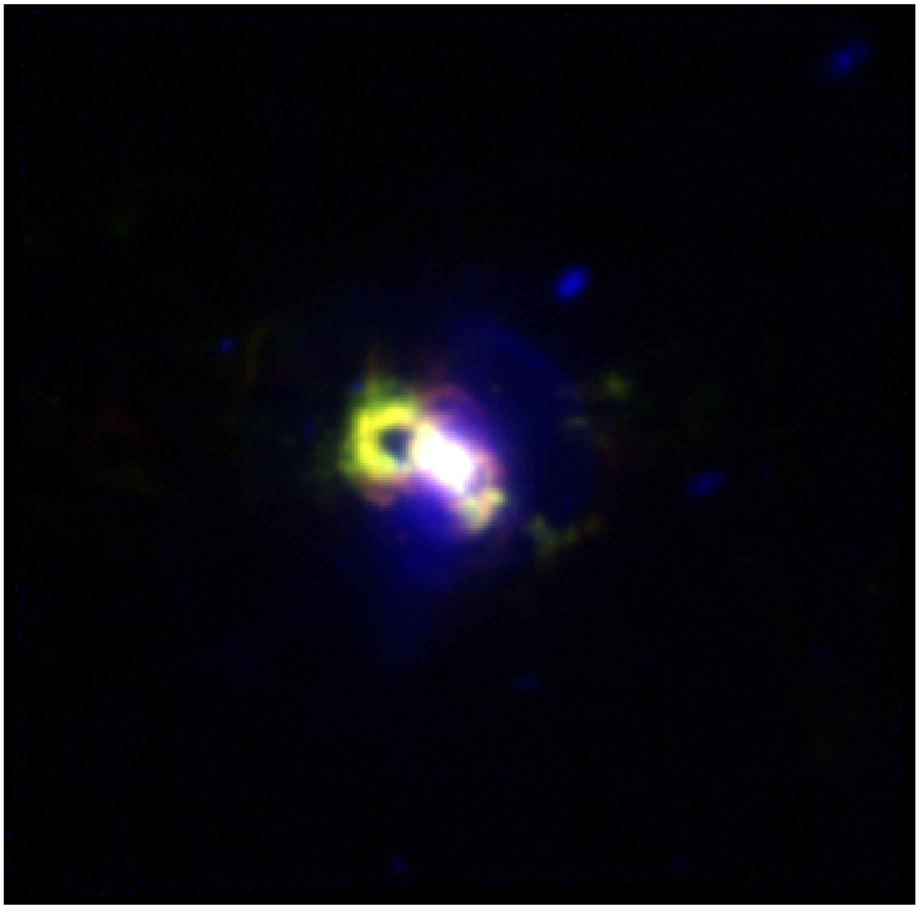
```
moment_1_masked = OIII_subcube_masked.moment(order=1)

wcs = cube2.wcs.sub(['longitude', 'latitude'])
norm = simple_norm(moment_1_masked.value, 'linear', vmin=-300,
vmax=300)

fig = plt.figure(figsize=(6, 5))
ax = plt.subplot(projection=wcs)
im = ax.imshow(moment_1_masked.value, origin='lower',
               cmap='coolwarm', norm=norm)
cbar = plt.colorbar(im, ax=ax, pad=0.02)
cbar.ax.yaxis.set_major_formatter(ScalarFormatter(useMathText=True))
cbar.ax.tick_params(labelsize=9)
cbar.set_label(r"Velocidad [km\,s$^{-1}$]", fontsize=12)
ax.set_xlabel(r"Ascensión Recta", fontsize=12)
ax.set_ylabel(r"Declinación", fontsize=12)
ax.set_title("Momento 1 [O III]", fontsize=12, pad=10)
ax.coords[0].set_format_unit('hour')
ax.coords[1].set_format_unit('degree')
ax.coords.grid(color='gray', ls=':', lw=0.4)
plt.tight_layout()
plt.show()
```



```
moment_2_masked = OIII_subcube_masked.moment(order=2)
sigma_map = np.sqrt(moment_2_masked.value)

wcs = cube2.wcs.celestial
norm = simple_norm(sigma_map, 'linear', vmin=0, vmax=400)

fig = plt.figure(figsize=(6,5))
ax = plt.subplot(projection=wcs)
im = ax.imshow(sigma_map, origin='lower', cmap='inferno', norm=norm)
cbar = plt.colorbar(im, ax=ax, pad=0.02)
cbar.ax.yaxis.set_major_formatter(ScalarFormatter(useMathText=True))
cbar.ax.tick_params(labelsize=9)
cbar.set_label(r"Dispersión $ {\sigma}$ [km\,s$^{-1}$]", fontsize=12)
ax.set_xlabel(r"Ascensión Recta", fontsize=12)
ax.set_ylabel(r"Declinación", fontsize=12)
ax.set_title(r"Momento 2 [O III] ", fontsize=12, pad=10)
ax.invert_xaxis()
ax.coords.grid(color='gray', ls=':', lw=0.4)
plt.tight_layout()
plt.show()
```



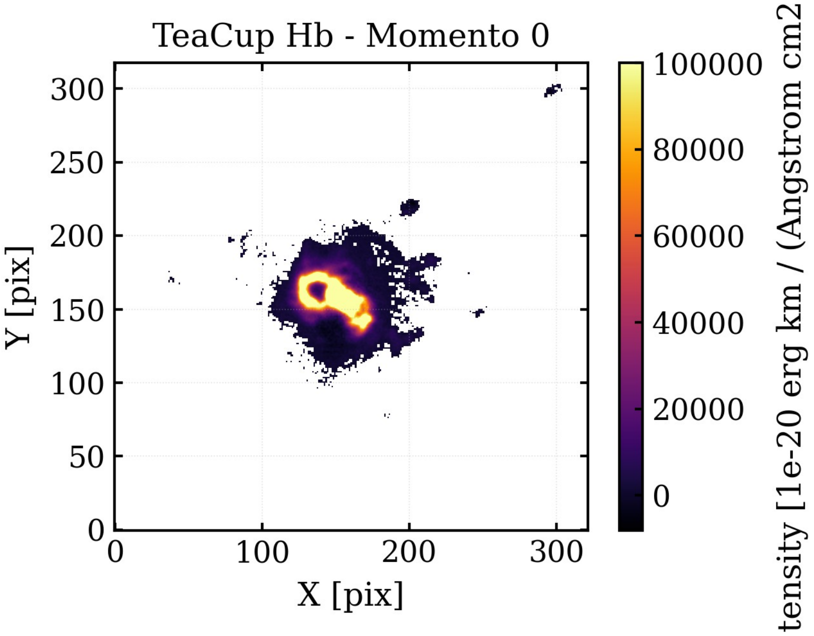
```
z = 0.08571
half_line = 10.0
cont_win = (6000.0, 6200.0)

cube_cont_path = "teacup.fits"
cube_lines_path = "teacup_continuum_sub.fits"
hdu_idx = 1

lam0_oiii = 5006.84
lam0_ha   = 6562.80
lam_oiii = lam0_oiii * (1.0 + z)
lam_ha   = lam0_ha   * (1.0 + z)

def collapse_window_sum(cube, lam_center, half_width):
    w = cube.spectral_axis.to(u.AA).value
    sel = (w >= lam_center - half_width) & (w <= lam_center +
half_width)
    idx = np.nonzero(sel)[0]
    if idx.size == 0:
        raise ValueError(f"No hay canales en {lam_center:.1f}±
{half_width:.1f} Å")
    img = np.zeros((cube.shape[1], cube.shape[2]), dtype=float)
    for i in idx:
        sli = cube.filled_data[i, :, :]
        arr = sli.value

        np.nan_to_num(arr, copy=False, nan=0.0)
        img += arr
    return img

def collapse_window_median(cube, win):
    w1, w2 = win
    w = cube.spectral_axis.to(u.AA).value
    sel = (w >= w1) & (w <= w2)
    idx = np.nonzero(sel)[0]
    if idx.size == 0:
        raise ValueError(f"No hay canales en ventana continuo 
{w1:.1f}-{w2:.1f} Å")
    stack = []
    for i in idx:
        arr = cube.filled_data[i, :, :].value
        stack.append(arr)
    stack = np.stack(stack, axis=0)
    return np.nanmedian(stack, axis=0)

cube_cont  = SpectralCube.read(cube_cont_path,  hdu=hdu_idx)
cube_lines = SpectralCube.read(cube_lines_path, hdu=hdu_idx)

img_B = collapse_window_median(cube_cont, cont_win)
img_G = collapse_window_sum(cube_lines, lam_oiii, half_line)
img_R = collapse_window_sum(cube_lines, lam_ha, half_line)

def stretch_asinh(img, pmin=5, pmax=99.7, soft=0.1):
    lo, hi = np.nanpercentile(img, [pmin, pmax])
    if hi <= lo:
        hi = lo + (np.nanstd(img) or 1.0)
    x = np.clip((img - lo) / (hi - lo), 0, 1)
    return np.arcsinh(x / soft) / np.arcsinh(1/soft)

R = stretch_asinh(img_R)
G = stretch_asinh(img_G)
B = stretch_asinh(img_B)

rgb = np.dstack([R, G, B])
rgb = np.clip(rgb, 0, 1)

plt.figure(figsize=(6,5))
plt.imshow(rgb, origin='lower')
plt.axis('off')
plt.tight_layout()
plt.show()
```



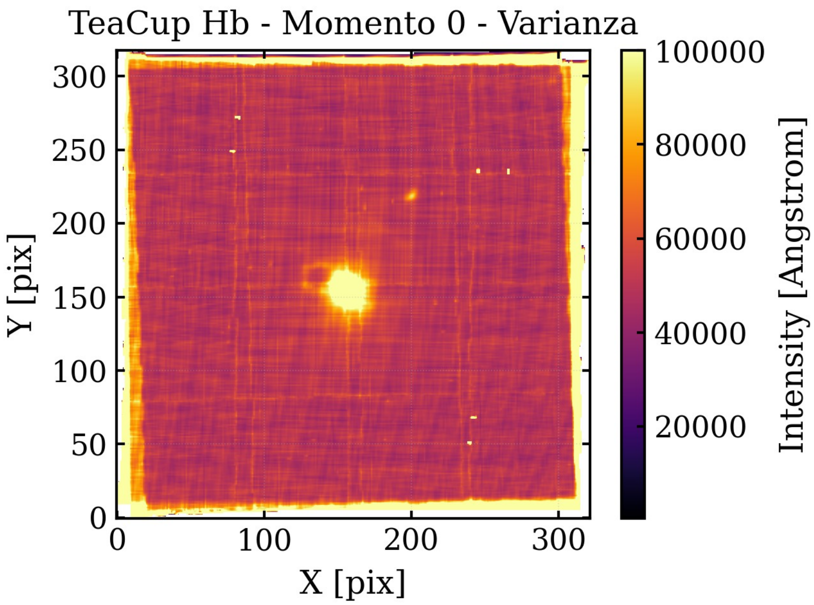
```
□ Guardado: rgb_Ha_OIII_continuum.png
```

```
cube3 = SpectralCube.read('teacup_continuum_sub.fits',hdu=1)

Hb_cube = fits.open('teacup.fits')
cube3_sn =
SpectralCube(data=Hb_cube[1].data/np.sqrt(Hb_cube[2].data),wcs=cube3.w
cs)

cube3_masked = cube3.with_mask(cube3_sn>3.5)

Hb_cube_masked =
cube3_masked.with_spectral_unit(u.km/u.s,velocity_convention='optical'
,rest_value=5277.03*u.AA)
Hb_subcube_masked = Hb_cube_masked.spectral_slab(-
600*u.km/u.s,600*u.km/u.s)

moment_0_masked_Hb = Hb_subcube_masked.moment(order=0)

plt.imshow(moment_0_masked_Hb.value, origin='lower', cmap='inferno',
vmax=1e5)
cbar = plt.colorbar()
```

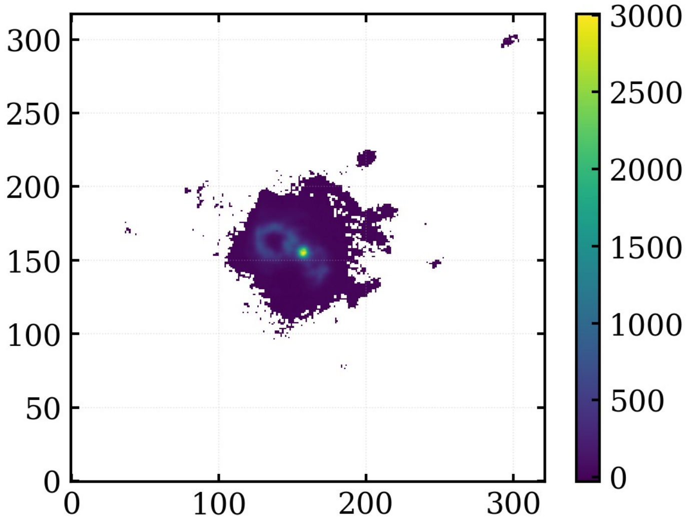
```
cbar.set_label(f"Intensity [{moment_0_masked_Hb.unit}]")
plt.xlabel("X [pix]")
plt.ylabel("Y [pix]")
plt.title("TeaCup Hb - Momento 0")
plt.show()
```



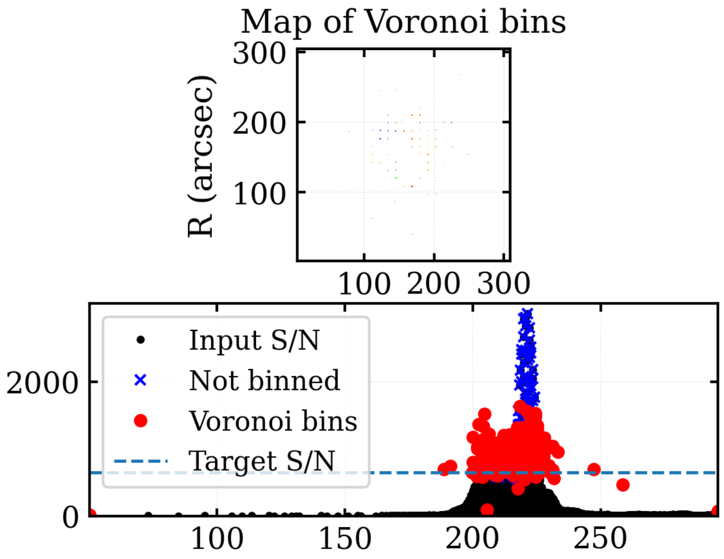
```
cube4 = SpectralCube.read('teacup.fits',hdu=2)

# Hb_cube = fits.open('teacup.fits')
# cube4_sn =
SpectralCube(data=Hb_cube[1].data/np.sqrt(Hb_cube[2].data))

# cube4_masked = cube3.with_mask(cube4_sn>3.5)

# Hb_cube_masked =
cube4_masked.with_spectral_unit(u.km/u.s,velocity_convention='optical'
,rest_value=5277.03*u.AA)
# Hb_subcube_masked =
Hb_cube_masked.spectral_slab(-600*u.km/u.s,600*u.km/u.s)

moment_0_masked_Hb_var = cube4.moment(order=0)

plt.imshow(moment_0_masked_Hb_var.value, origin='lower',
cmap='inferno', vmax=1e5)
cbar = plt.colorbar()
cbar.set_label(f"Intensity [{moment_0_masked_Hb_var.unit}]")
plt.xlabel("X [pix]")
plt.ylabel("Y [pix]")
plt.title("TeaCup Hb - Momento 0 - Varianza")
plt.show()
```



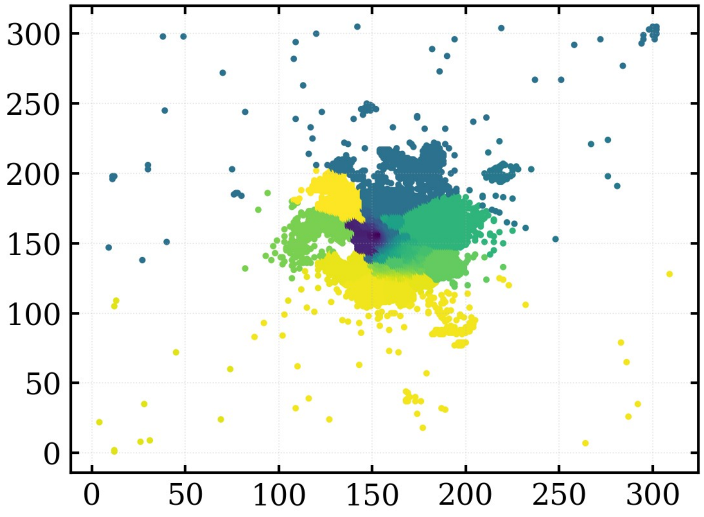
```
signal = moment_0_masked_Hb.value
noise = np.sqrt(moment_0_masked_Hb_var).value
sn = signal/noise

x,y = np.nonzero(sn>5)

plt.imshow(sn,origin='lower')
plt.colorbar()

<matplotlib.colorbar.Colorbar at 0x7f73f18d7580>
```



```
binNum, x_gen, y_gen, x_bar, y_bar, sn, nPixels, scale = 
voronoi_2d_binning(x, y, signal[x,y], noise[x,y], 650, plot=1, 
quiet=1,pixelsize=0.2)
```



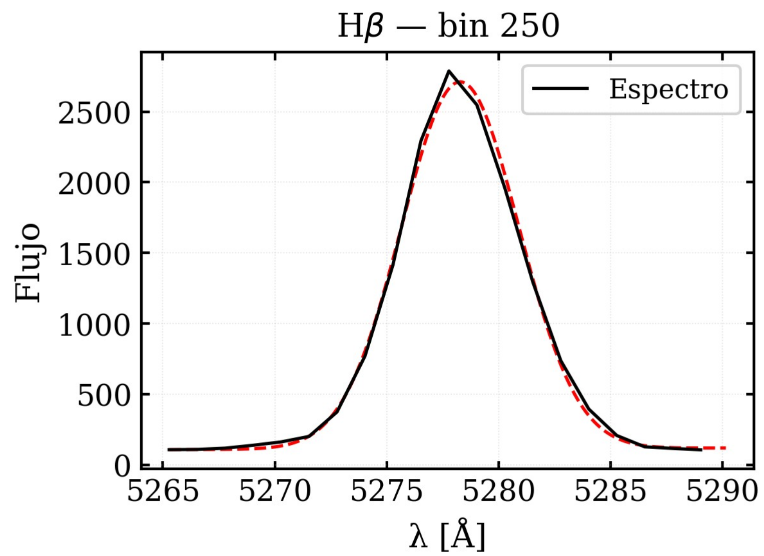
```
scatter=plt.scatter(x, y, c=binNum, s=2, cmap='viridis')
```

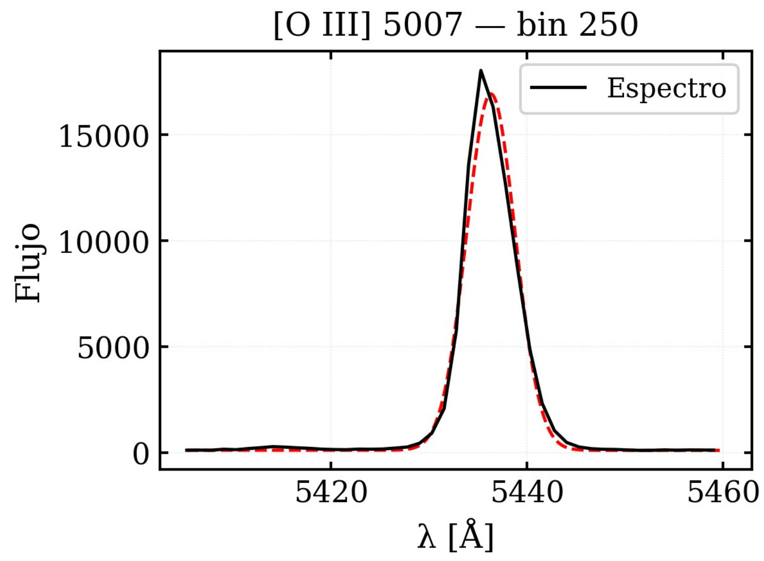


```
len(np.unique(binNum))
```

```
305
```

```
# Flujo_OIII = []
# Flujo_Hb   = []

# cube2 = cube.copy()
# for b in np.unique(binNum):
#     x_b = x[binNum == b]
#     y_b = y[binNum == b]
#     if x_b.size == 0:
#         continue

#     cube2.mask[...] = True
#     cube2.mask[:, x_b, y_b] = False

#     spec_bin = cube2.sum(axis=(1, 2))

#     g_oiii = spec_bin.gauss_fit(unit=u.angstrom, lmin=5405.0,
lmax=5460.0, plot=False)
#     g_hb   = spec_bin.gauss_fit(unit=u.angstrom, lmin=5265.0,
lmax=5290.0, plot=False)

#     F_oiii = float(g_oiii.flux)
#     F_hb   = float(g_hb.flux)

#     Flujo_OIII.append(F_oiii)
#     Flujo_Hb.append(F_hb)

# Flujo_OIII = np.array(Flujo_OIII)
# Flujo_Hb   = np.array(Flujo_Hb)

b = 250

x_b = x[binNum == b]
y_b = y[binNum == b]
assert x_b.size > 0, f"Bin {b} sin spaxels."

cube2 = cube.copy()
cube2.mask[...] = True
cube2.mask[:, x_b, y_b] = False
spec_bin = cube2.sum(axis=(1, 2))

w_full = spec_bin.wave.coord()
f_full = spec_bin.data

def fit(lmin, lmax, titulo):
    g = spec_bin.gauss_fit(unit=u.angstrom, lmin=lmin, lmax=lmax,
plot=True)
    ax = plt.gca()
    m = (w_full >= lmin) & (w_full <= lmax)
    ax.plot(w_full[m], f_full[m], 'k-', lw=1.2, label='Espectro')
    ax.legend(loc='best', frameon=True, framealpha=0.85)
    ax.set_title(f"{titulo} – bin {b}")
    plt.xlabel("λ [Å]")
    plt.ylabel("Flujo")
    plt.tight_layout()
    plt.show()
    F = float(getattr(g, 'flux', np.nan))
    return F

F_hb = fit(5265.0, 5290.0, r'H$\beta$')
F_o3 = fit(5405.0, 5460.0, r'[O III] 5007')

print(f"bin {b}:  F(Hβ) = {F_hb:.3g},   F([O III]) = {F_o3:.3g}")
```

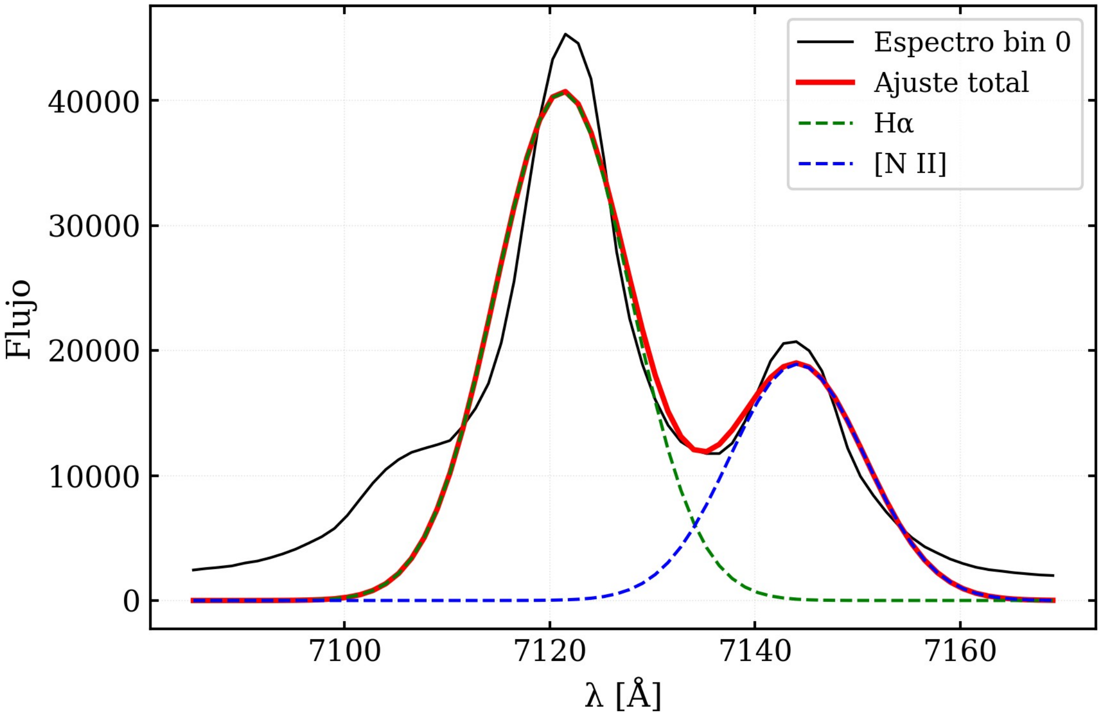


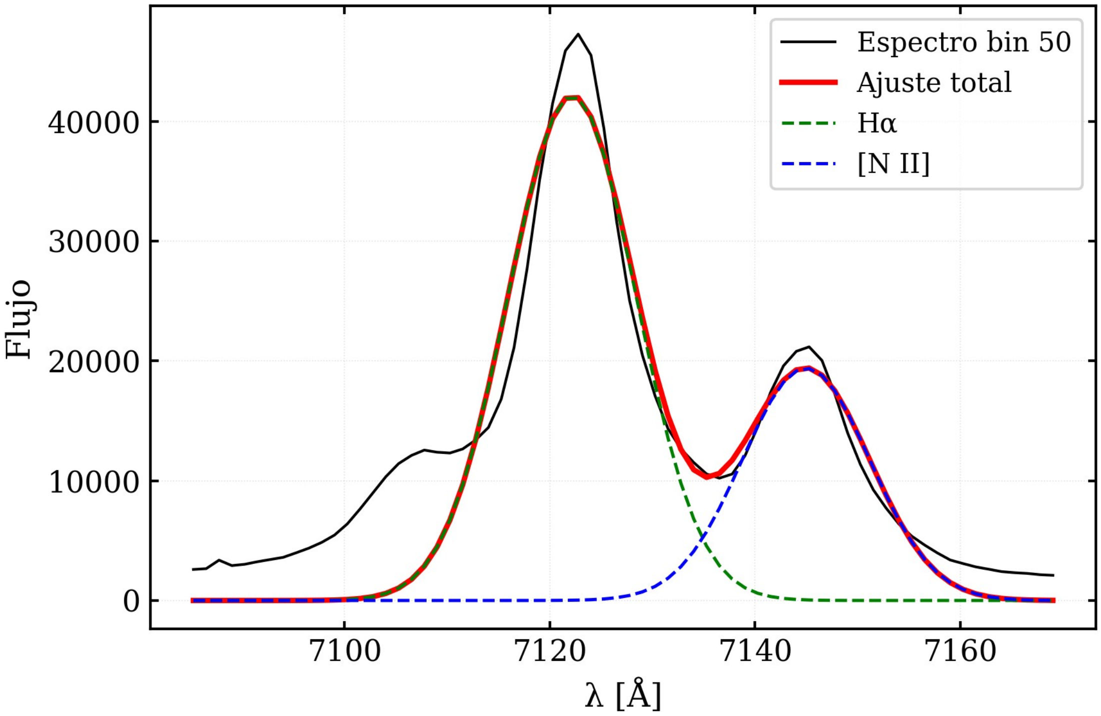


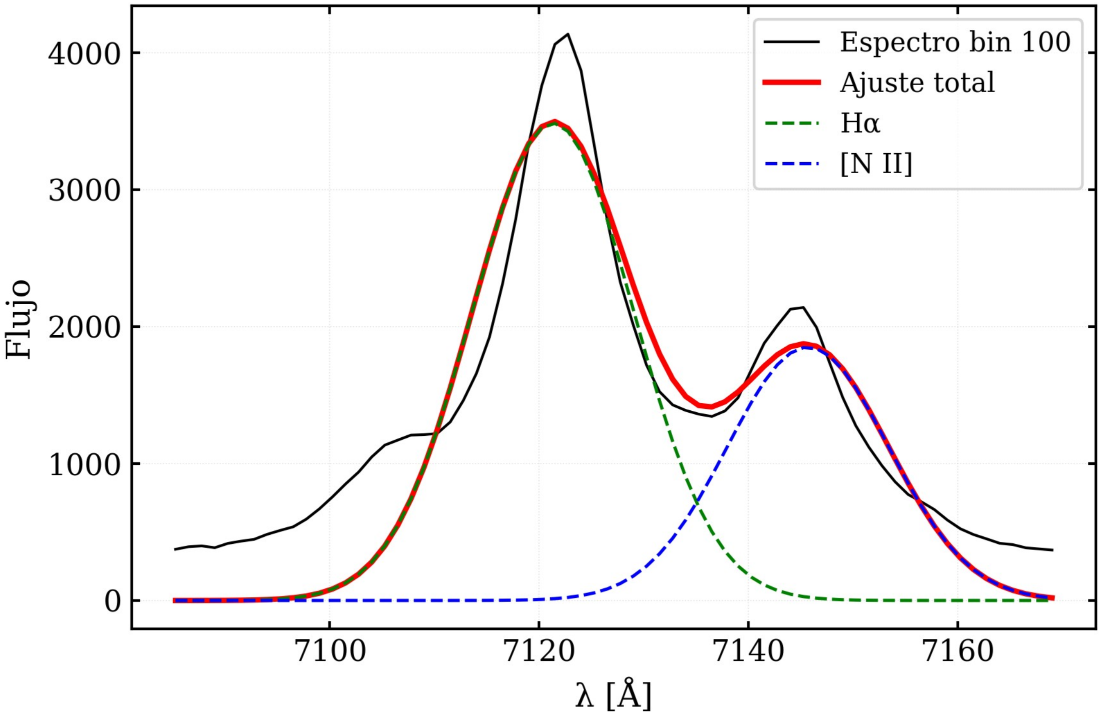
```
bin 250:  F(Hβ) = 1.7e+04,   F([O III]) = 1.05e+05
```

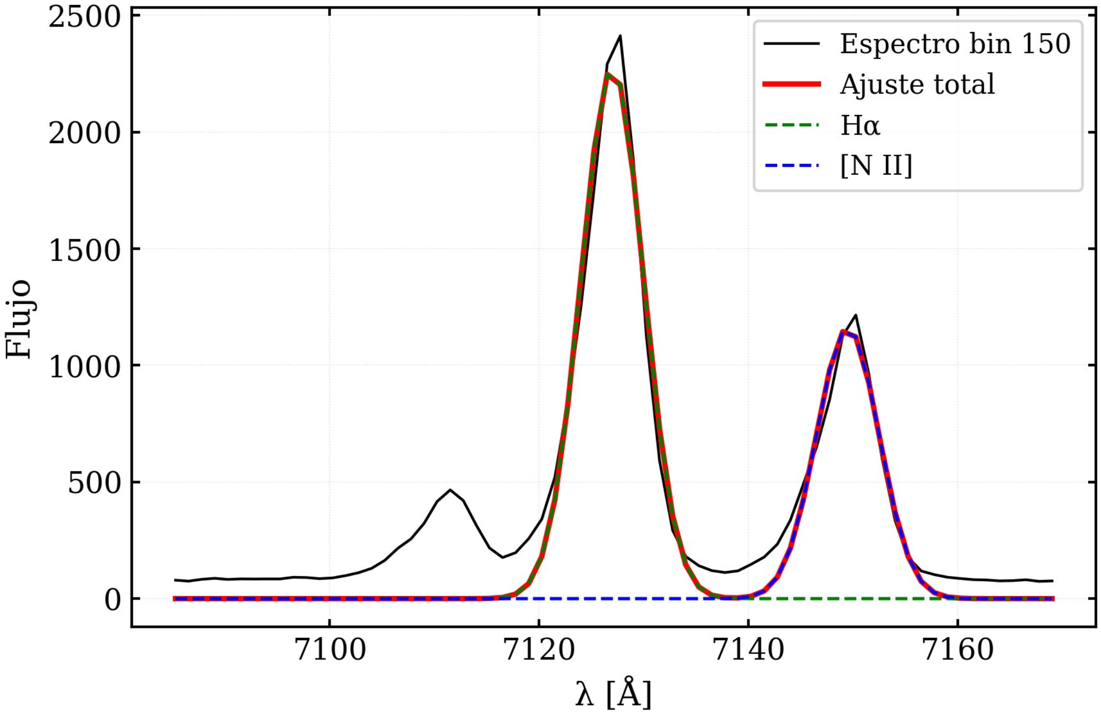
```
def two_gauss(w, Aha, muha, Ani, muni, sigma):
    return (Aha*np.exp(-(w-muha)**2/(2*sigma**2)) +
            Ani*np.exp(-(w-muni)**2/(2*sigma**2)))
```

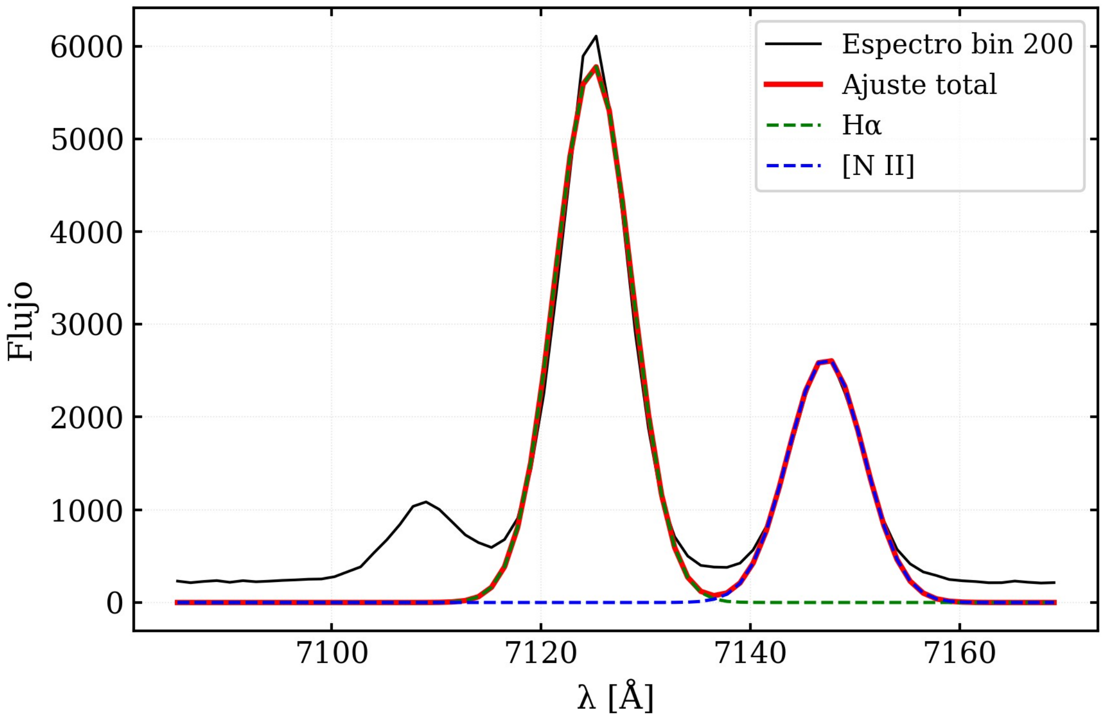
```
WMIN, WMAX = 7085.0, 7170.0
p0 = [15000, 7122.0, 6000, 7145.0, 2.5]
bounds = (
    [0, 7115.0, 0, 7135.0, 0.6],
    [np.inf, 7128.0, np.inf, 7155.0, 8.0]
)
Flux_Ha  = []
Flux_NII = []
cube2 = cube.copy()

for b in np.unique(binNum):
    x_b = x[binNum == b]
    y_b = y[binNum == b]
    if x_b.size == 0:
        continue
    cube2.mask[...] = True
    cube2.mask[:, x_b, y_b] = False
    spec_bin = cube2.sum(axis=(1,2))
    w = spec_bin.wave.coord()
    f = spec_bin.data
    m = (w >= WMIN) & (w <= WMAX)
    w, f = w[m], f[m]

    popt, _ = curve_fit(two_gauss, w, f, p0=p0, bounds=bounds,
maxfev=20000)
    Aha, muha, Ani, muni, sigma = popt
    Fha  = Aha * sigma * math.sqrt(2*np.pi)
    Fnii = Ani * sigma * math.sqrt(2*np.pi)
    Flux_Ha.append(Fha)
    Flux_NII.append(Fnii)

    if b % 50 == 0:
        ft = two_gauss(w, *popt)
        plt.figure(figsize=(6,4))
        plt.plot(w, f, 'k', lw=1, label='Espectro bin %d' % b)
        plt.plot(w, ft, 'r-', lw=2, label='Ajuste total')
        plt.plot(w, Aha*np.exp(-(w-muha)**2/(2*sigma**2)), 'g--',
label='Hα')
        plt.plot(w, Ani*np.exp(-(w-muni)**2/(2*sigma**2)), 'b--',
label='[N II]')
        plt.xlabel("λ [Å]")
        plt.ylabel("Flujo")
        plt.legend()
        plt.tight_layout()
        plt.show()

Flux_Ha  = np.array(Flux_Ha)
Flux_NII = np.array(Flux_NII)
```

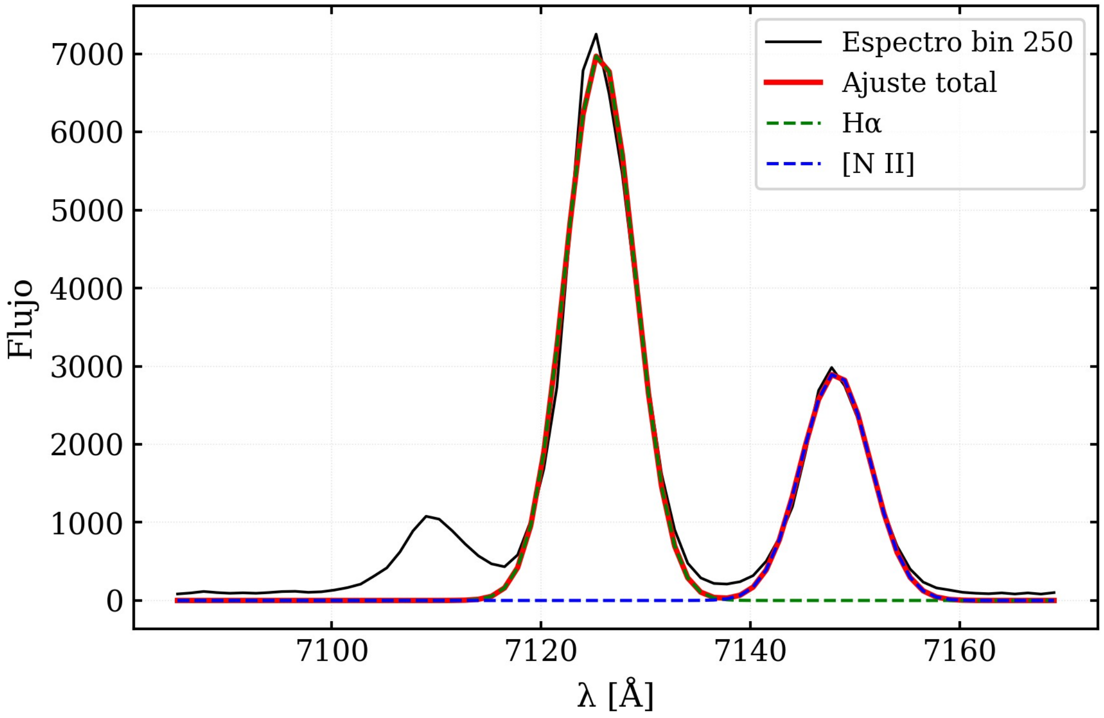

Espectro bin 0
Ajuste total
Hα
[N II]
Flujo
0
10000
20000
30000
40000
7100
7120
7140
7160
λ [Å]

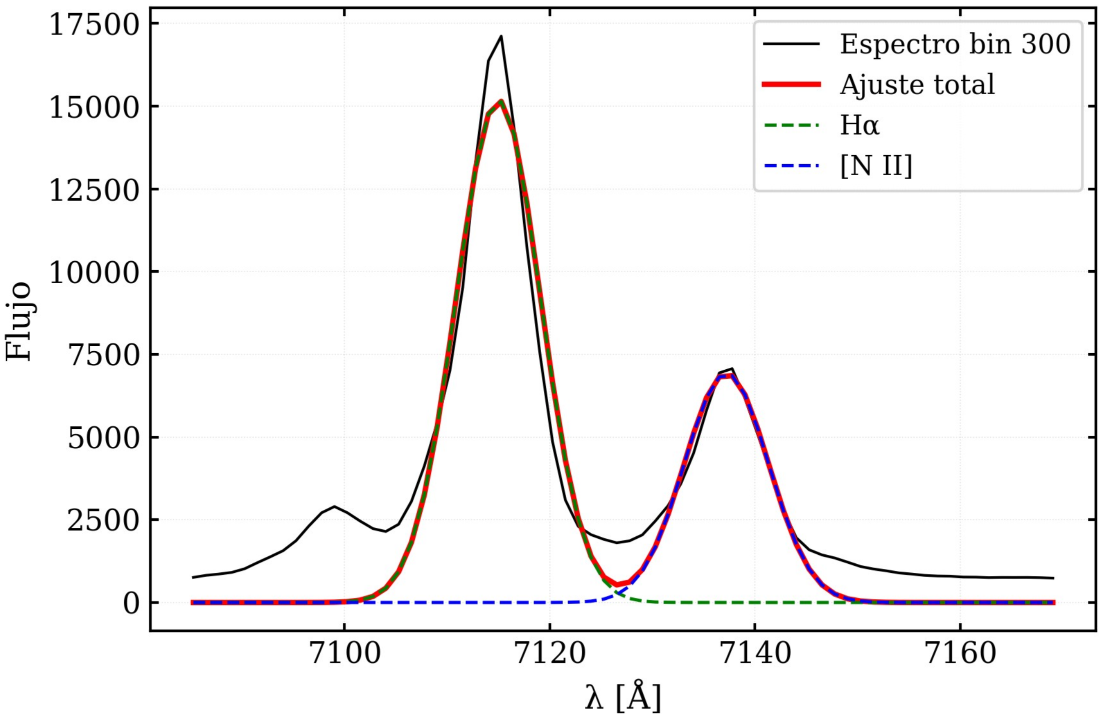

Espectro bin 50
Ajuste total
Hα
[N II]
Flujo
0
10000
20000
30000
40000
7100
7120
7140
7160
λ [Å]


Espectro bin 100
Ajuste total
Hα
[N II]
Flujo
λ [Å]
0
1000
2000
3000
4000
7100
7120
7140
7160


Espectro bin 150
Ajuste total
Hα
[N II]
Flujo
λ [Å]
0
500
1000
1500
2000
2500
7100
7120
7140
7160

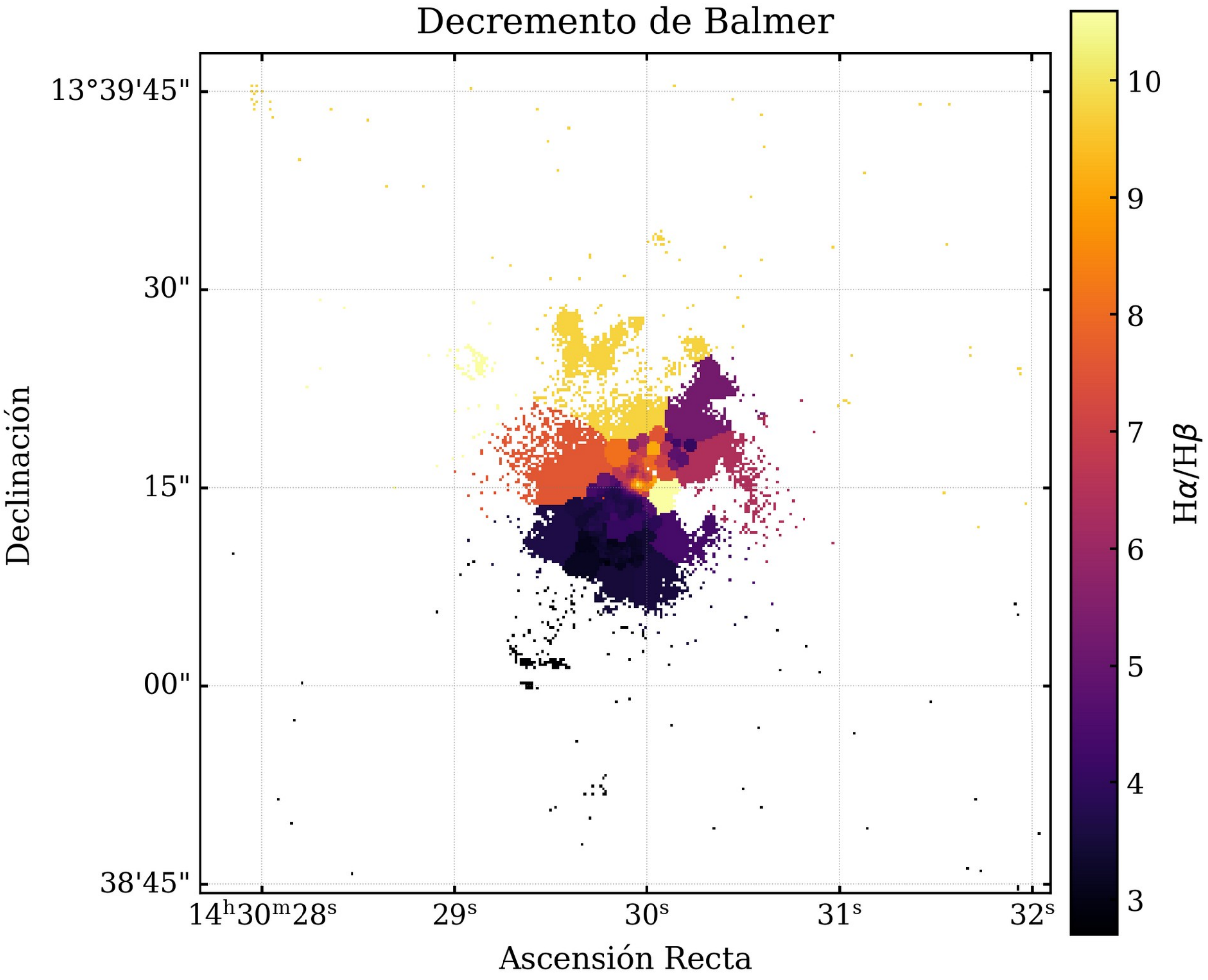

Espectro bin 200
Ajuste total
Hα
[N II]
Flujo
λ [Å]
0
1000
2000
3000
4000
5000
6000
7100
7120
7140
7160


Espectro bin 250
Ajuste total
Hα
[N II]
Flujo
λ [Å]
0
1000
2000
3000
4000
5000
6000
7000
7100
7120
7140
7160



```
df = pd.read_csv("flujos_por_bin.csv")
bins = df["bin"].to_numpy(int)
nbins = bins.max() + 1

Flux_Ha = np.full(nbins, np.nan, float)
Flux_Hb = np.full(nbins, np.nan, float)
Flux_Ha[bins] = df["Flux_Ha"].to_numpy(float)
Flux_Hb[bins] = df["Flux_Hb"].to_numpy(float)

ratio = Flux_Ha / Flux_Hb

ny, nx = cube.shape[1], cube.shape[2]
map_ratio = np.full((ny, nx), np.nan, float)

map_ratio[y, x] = ratio[binNum]

try:
    wcs2d = cube2.wcs.celestial
except Exception:
    hdr = fits.getheader("teacup_continuum_sub.fits", 1)
    wcs2d = WCS(hdr).celestial

vals = map_ratio[np.isfinite(map_ratio)]
vmin, vmax = np.nanpercentile(vals, [2, 98]) if vals.size else (1.5,
5.0)
if vmax <= vmin:
    vmax = vmin + (np.nanstd(vals) or 1.0)
```

```
fig = plt.figure(figsize=(7,6))
ax = plt.subplot(projection=wcs2d)

im = ax.imshow(map_ratio, origin='lower', cmap='inferno', vmin=vmin,
vmax=vmax)
cbar = plt.colorbar(im, ax=ax, pad=0.02)
cbar.ax.yaxis.set_major_formatter(ScalarFormatter(useMathText=True))
cbar.set_label(r'H$\alpha$/H$\beta$', fontsize=12)

ax.set_xlabel('Ascensión Recta', fontsize=12)
ax.set_ylabel('Declinación', fontsize=12)
ax.set_title(r'Decremento de Balmer', fontsize=14, pad=8)

ax.invert_xaxis()
ax.coords.grid(color='gray', ls=':', lw=0.4, alpha=0.6)

plt.tight_layout()
plt.show()
```

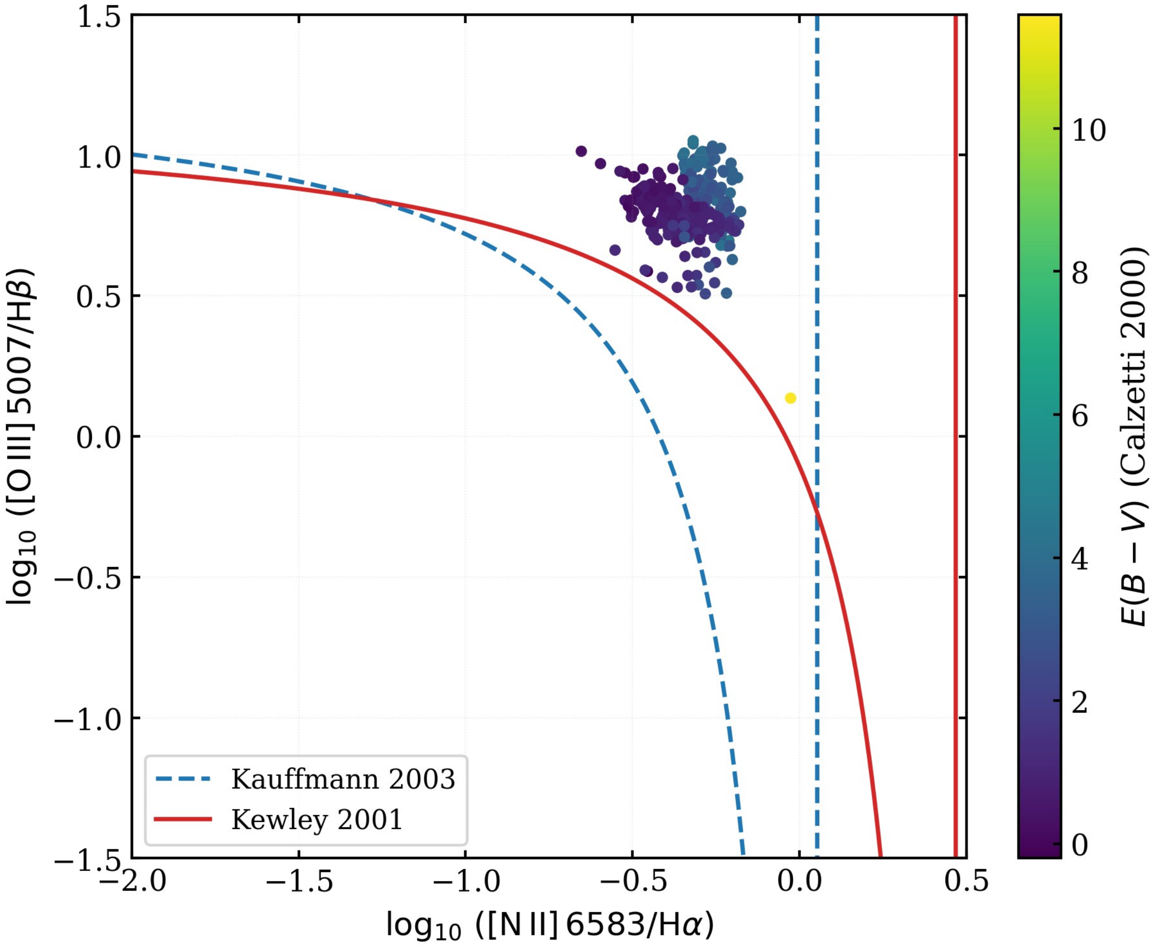


```
df = pd.read_csv("flujos_por_bin_corrected.csv")
bins = df["bin"].astype(int).to_numpy()

x = np.log10(df["NII_corr"] / df["Ha_corr"])
y = np.log10(df["OIII_corr"] / df["Hb_corr"])

def bpt_classify(xi, yi):
    if not np.isfinite(xi) or not np.isfinite(yi):
        return "NaN"
    kauff  = 0.61/(xi - 0.05) + 1.3
    kewley = 0.61/(xi - 0.47) + 1.19
    if yi > kewley:
        return "Seyfert"
    elif yi > kauff:
        return "Composite"
    else:
        return "SF"

df["BPT_class"] = [bpt_classify(xi, yi) for xi, yi in zip(x, y)]
df.to_csv("flujos_por_bin_BPT.csv", index=False)
```

```
OK: columna BPT_class agregada
```

```
df = pd.read_csv("flujos_por_bin.csv").astype(float)

HAB_HB_teor = 2.86
kHb = extinction.calzetti00(np.array([4861.0]), 1.0, 4.05)[0]
kHa = extinction.calzetti00(np.array([6563.0]), 1.0, 4.05)[0]
kO3 = extinction.calzetti00(np.array([5007.0]), 1.0, 4.05)[0]
kN2 = extinction.calzetti00(np.array([6583.0]), 1.0, 4.05)[0]

ratio_obs = (df["Flux_Ha"] / df["Flux_Hb"]).replace([np.inf, -np.inf],
np.nan)
ratio_obs = ratio_obs.where(ratio_obs > 0)
EBV = (2.5 / (kHb - kHa)) * np.log10(ratio_obs / HAB_HB_teor)

def corr_factor(k, ebv):
    return np.power(10.0, 0.4 * k * ebv)

Ha   = df["Flux_Ha"]   * corr_factor(kHa, EBV)
Hb   = df["Flux_Hb"]   * corr_factor(kHb, EBV)
OIII = df["Flux_OIII"] * corr_factor(kO3, EBV)
NII  = df["Flux_NII"]  * corr_factor(kN2, EBV)

x = np.log10(NII / Ha)
y = np.log10(OIII / Hb)
finite = np.isfinite(x) & np.isfinite(y)

xx = np.linspace(-2.0, 0.5, 600)
kauff03 = 0.61/(xx - 0.05) + 1.3      # Kauffmann (2003): límite
empírico SF
```

```python
kewley01 = 0.61/(xx - 0.47) + 1.19    # Kewley (2001): límite teórico SF

plt.figure(figsize=(6.4, 5.2))
plt.plot(xx, kauff03, '--', lw=1.6, color='#1f77b4', label='Kauffmann 2003')
plt.plot(xx, kewley01, '-',  lw=1.6, color='#d62728', label='Kewley 2001')

sc = plt.scatter(x[finite], y[finite], c=EBV[finite], s=18, cmap='viridis', edgecolor='none')
cbar = plt.colorbar(sc); cbar.set_label(r'$E(B-V)$ (Calzetti 2000)')

plt.xlim(-2.0, 0.5)
plt.ylim(-1.5, 1.5)
plt.xlabel(r'$\log_{10}\,([\mathrm{N\,II}]\,6583/\mathrm{H}\alpha)$')
plt.ylabel(r'$\log_{10}\,([\mathrm{O\,III}]\,5007/\mathrm{H}\beta)$')
plt.grid(alpha=0.25, ls=':')
plt.legend(loc='lower left')
plt.tight_layout()
plt.show()
```



```
z_sys = 0.08571

df = pd.read_csv("flujos_por_bin_corrected.csv")
dL = cosmo.luminosity_distance(z_sys).to(u.cm).value
fac = 4.0 * np.pi * dL**2
df["L_Ha"]   = fac * df["Ha_corr"].to_numpy()
df["L_Hb"]   = fac * df["Hb_corr"].to_numpy()
df["L_OIII"] = fac * df["OIII_corr"].to_numpy()
df["L_NII"]  = fac * df["NII_corr"].to_numpy()
df["log_NII_Ha"]  = np.log10(df["L_NII"]  / df["L_Ha"])
df["log_OIII_Hb"] = np.log10(df["L_OIII"] / df["L_Hb"])

df.to_csv("flujos_y_luminosidades_por_bin.csv", index=False)
```