# 20250610 M and SFR pipeline

---

## Stellar mass $M_*$

`Mass.py` extends a MAUVE galaxy's Voronoi-binning file by adding photometric R-band flux, extinction-corrected magnitudes, $M/L_R$, stellar mass, and mass–surface-density layers, all in a single automated pass. The workflow evoloves MUSE cubes, existing bin maps, BaSTI+Chabrier SSP weights, and filter curves from `speclite`, then writes the new products back into `{GAL}_SPATIAL_BINNING_maps_extended.fits`.

### 1 Command-line interface

Simply use `python Mass.py` to run all galaxies by default, or specifc galaxy's ID with `-g` or `--galaxy`.

### 2 Key inputs

- **MUSE cube** (`*_DATACUBE_FINAL_WCS_Pall_mad_red_v3.fits`) — 3-D array in units of ($10^{-20}$ erg s$^{-1}$ cm$^{-2}$ Å$^{-1}$).
- **Voronoi maps** — BINID & FLUX layers produced with the Cappellari & Copin (2003) adaptive binning.
- **SFH weights** (`*_sfh-weights.fits`) — $4077 \times 477$ grid giving the light-fraction of each BaSTI SSP in every Voronoi bin.
- **Photometry table** (`BaSTI+Chabrier.dat`) — R-band $M/L_R$ lookup keyed by age (Gyr) & metallicity (dex) for a Chabrier IMF.

### 3 M/L lookup & per-bin weighting

Rows with `IMF == 'Ch'` are filtered out of the merged BaSTI archive; ages & metallicities are rounded to 0.01 dex/Gyr to build a hash map → $M/L_R$. Weighted averages

$$M/L_R^{\rm bin} = \sum_i w_i \, (M/L_R)_i \tag{1}$$

are computed via matrix multiplication. Every bin is verified finite & positive.

## 4 Bessell R-band magnitudes and flux per spaxel

The Bessell R-band magnitude of each bin is obtained by using `speclite`'s `load_filter('bessell-R')`.

## 5 Galactic-extinction correction

Stellar $E(B-V)$ from the SFH file is scaled to $A_r$ with

**Bessell R**: $A_r = 2.32\,E(B-V)$ (Fitzpatrick 1999).

Corrected magnitudes are back-converted to **FLUX_R_corr** in the unit of nanomaggies for comparison with Legacy data.

## 6 Stellar luminosity & mass

Adopting Virgo Cluster distance at $D = 16.5$ Mpc and solar R-band absolute magnitude $M_{r,\odot} = 4.64$, we can compute R-band luminosity $L_R$ in solar unit. Then stellar mass per spaxel is $M_\star = L_R \times M/L_{R_{\rm bin}}$, then take $\log_{10}$ and logged as **LOGMSTAR**.

## 7 Surface-density map

Pixel scales from the WCS yield the physical area; dividing by $A_{\rm pix}$ converts mass to $\Sigma_\star [M_\odot \,{\rm kpc}^{-2}]$ and then I take $\log_{10}$ stored it as **LOGMASS_SURFACE_DENSITY**.

## 8 Output FITS structure

The script clones every original HDU, then appends:

1. **FLUX_R_corr** (nanomaggies)
2. **ML_R** $(M/L_R)$

3. **LOGMSTAR** (log (M_\odot))
4. **LOGMASS_SURFACE_DENSITY** $(\log\left(M_{\odot}\,\mathrm{kpc}^{-2}\right))$
5. **magnitude_r** and **magnitude_r_uncorrected** (mag_AB)

# SFR

## 1 Command-line interface & initial setup

Simply use `python SFR.py` to run all galaxies by default, or specify galaxy's ID with `-g` or `--galaxy`.

```
def cli():
    p = argparse.ArgumentParser(description="Generate SFR maps for
a MAUVE galaxy")
    p.add_argument("-g", "--galaxy", default="IC3392",
                   help="Galaxy identifier (default IC3392)")
    p.add_argument("--root", default="/arc/projects/mauve",
                   help="Root of MAUVE directory tree")
    p.add_argument("-v", "--verbose", action="store_true",
                   help="Verbose logging")
    return p.parse_args()

args = cli()
loglvl = logging.INFO if args.verbose else logging.WARNING
logging.basicConfig(level=loglvl,
                    format="%(asctime)s  %(levelname)-8s  %
(message)s",
                    datefmt="%H:%M:%S")
```

## 2 Files & sanity checks

All input/output paths are constructed relative to `--root/products/v0.6/{GAL}`, with default root address `root="/arc/projects/mauve"`; execution halts if mandatory FITS files are missing.

```
galaxy = args.galaxy.upper()
root   = Path(args.root).expanduser().resolve()

gas_path = root / "products/v0.6" / galaxy / f"
{galaxy}_gas_BIN_maps.fits"
SFH_path = root / "products/v0.6" / galaxy / f"
{galaxy}_SFH_maps.fits"
out_path = Path(f"{galaxy}_gas_BIN_maps_extended.fits")

missing = [p for p in (gas_path, SFH_path) if not p.is_file()]
if missing:
    for p in missing:
        logging.error("Missing file: %s", p)
    sys.exit(1)
```

## 3 Load data

**Gas cube** – getting flux and corresponding error maps for all key optical lines.

```
with fits.open(gas_path) as hdul:
    HB4861_FLUX, HB4861_FLUX_ERR = hdul['HB4861_FLUX'].data,
hdul['HB4861_FLUX_ERR'].data
    HA6562_FLUX, HA6562_FLUX_ERR = hdul['HA6562_FLUX'].data,
hdul['HA6562_FLUX_ERR'].data
    OIII5006_FLUX, OIII5006_FLUX_ERR = hdul['OIII5006_FLUX'].data,
hdul['OIII5006_FLUX_ERR'].data
    NII6583_FLUX, NII6583_FLUX_ERR = hdul['NII6583_FLUX'].data,
hdul['NII6583_FLUX_ERR'].data
    SII6716_FLUX, SII6716_FLUX_ERR = hdul['SII6716_FLUX'].data,
hdul['SII6716_FLUX_ERR'].data
    SII6730_FLUX, SII6730_FLUX_ERR = hdul['SII6730_FLUX'].data,
hdul['SII6730_FLUX_ERR'].data
```

## 4 Quality cut & Balmer decrement

Only spaxels with $Flux/Flux_{err} \geq 15$ in both H$\beta$ and H$\alpha$ are retained to calculate Balmer decrement (BD).

```python
cut = 15
mask = (HA6562_FLUX / HA6562_FLUX_ERR >= cut) & \
       (HB4861_FLUX / HB4861_FLUX_ERR >= cut)

HA6562_FLUX_cut = np.where(mask, HA6562_FLUX, np.nan)
HB4861_FLUX_cut = np.where(mask, HB4861_FLUX, np.nan)

BD = HA6562_FLUX_cut / HB4861_FLUX_cut
```

The decrement map is appended as a new HDU:

```python
BD_hdu = fits.ImageHDU(data=BD.astype(np.float64),
name="Balmer_Decrement")
BD_hdu.header.update(gas_header);  BD_hdu.header["BUNIT"] = ""
new_hdul = fits.HDUList([hdu.copy() for hdu in
fits.open(gas_path)])
new_hdul.append(BD_hdu)
new_hdul.writeto(out_path, overwrite=True)
```

# 5 Gas $E(B-V)$

Using Calzetti et al. (2000) coefficients: $k_{\mathrm{H}\beta} = 4.598$, $k_{\mathrm{H}\alpha} = 3.325$ and $R_{\mathrm{int}} = 2.86$:

```python
k_Hb, k_Ha, R_int = 4.598, 3.325, 2.86
E_BV_BD = 2.5 / (k_Hb - k_Ha) * np.log10(BD / R_int)

with fits.open(out_path, mode="append") as hdul:
    hdul.append(fits.ImageHDU(data=E_BV_BD.astype(np.float64),
                              header=gas_header, name="E(B-V)_BD"))
    hdul.flush()
```

# 6 Quality cut again for emission lines

[O III], [N II] and [S II] are also required to have $Flux/Flux_{err} \geq 15$ simultaneously to identify H II region later.

```
logN2   = np.log10(NII6583_FLUX / HA6562_FLUX)        # [N II]/Hα
logS2   = np.log10((SII6716_FLUX+SII6730_FLUX) / HA6562_FLUX)   #
Σ[S II]/Hα
logO3   = np.log10(OIII5006_FLUX / HB4861_FLUX)       # [O III]/Hβ



mask_N2 = NII6583_FLUX / NII6583_FLUX_ERR >= cut
mask_S2 = (SII6716_FLUX + SII6730_FLUX) / (SII6716_FLUX_ERR +
SII6730_FLUX_ERR) >= cut
mask_O3 = OIII5006_FLUX / OIII5006_FLUX_ERR >= cut
mask_combinedd = mask_combined & mask_N2 & mask_S2 & mask_O3

logN2_cut = np.where(mask_combinedd, logN2, np.nan)
logS2_cut = np.where(mask_combinedd, logS2, np.nan)
logO3_cut = np.where(mask_combinedd, logO3, np.nan)
```

## 7 BPT classification & pure-SF mask

Classical Kauffmann (2003) / Kewley (2001, 2006) demarcations isolate star-forming spaxels **in both** [N II]– and [S II]–based diagrams.

```
logN2  = np.log10(NII6583_FLUX / HA6562_FLUX)
logS2  = np.log10((SII6716_FLUX + SII6730_FLUX) / HA6562_FLUX)
logO3  = np.log10(OIII5006_FLUX / HB4861_FLUX)

def kauff03_N2(x):  return 0.61 / (x - 0.05) + 1.30
def kewley01_N2(x): return 0.61 / (x - 0.47) + 1.19
def kewley01_S2(x): return 0.72 / (x - 0.32) + 1.30
def kewley06_SyLIN(x): return 1.89 * x + 0.76

N2_SF  = logO3 <= kauff03_N2(logN2)
S2_SF  = logO3 <= kewley01_S2(logS2)
mask_SF = mask & N2_SF & S2_SF

HA6562_FLUX_SF = np.where(mask_SF, HA6562_FLUX_cut, np.nan)
```

## 8 Extinction-corrected H$\alpha$ luminosity

Still assuming 16.5 Mpc and caclurating the corrected H$\alpha$ luminosity

```
HA6562_FLUX_corr = HA6562_FLUX_cut * 10**(0.4 * k_Ha * E_BV_BD)
HA6562_FLUX_SF_corr = HA6562_FLUX_SF * 10**(0.4 * k_Ha * E_BV_BD)

def flux_to_luminosity(flux, distance=16.5):
    return (flux*1e-20*u.erg/u.s/u.cm**2 * 4*np.pi*
(distance*u.Mpc)**2).cgs

HA6562_Luminosity_corr = flux_to_luminosity(HA6562_FLUX_corr)
HA6562_Luminosity = flux_to_luminosity(HA6562_FLUX)
HA6562_Luminosity_SF_corr = flux_to_luminosity(HA6562_FLUX_SF_corr)
```

The SF-only luminosity map is stored:

```
with fits.open(out_path, mode="append") as hdul:          # open
existing file
    HA6562_Luminosity_SF_corr_hdu = fits.ImageHDU(
        data=HA6562_Luminosity_SF_corr.value.astype(np.float64),  #
like others
        header=gas_header, name="Halpha_Luminosity_SF_corr")  # new
HDU name
    HA6562_Luminosity_SF_corr_hdu.header["BUNIT"] = "erg/s"   #
units keyword
    hdul.append(HA6562_Luminosity_SF_corr_hdu)
    hdul.flush()
```

# 9 Star-formation rate

Calzetti (2007) → Chabrier IMF scaling (i.e. $\frac{0.63}{0.67}$):

```
def calzetti_sfr(luminosity):
    return 5.3e-42 * luminosity.cgs.value / 0.67 * 0.63 # SFR in
M_sun/yr
HA6562_SFR_corr = calzetti_sfr(HA6562_Luminosity_corr)
HA6562_SFR_SF_corr = calzetti_sfr(HA6562_Luminosity_SF_corr)
```

The SF-only SFR map is stored:

```
with fits.open(out_path, mode="append") as hdul:             # open
existing file
    HA6562_SFR_SF_corr_hdu = fits.ImageHDU(
        data=HA6562_SFR_SF_corr.astype(np.float64),  # like others
        header=gas_header, name="Halpha_SFR_SF_corr")  # new HDU
name
    HA6562_SFR_SF_corr_hdu.header["BUNIT"] = "M_sun/yr"  # units
keyword
    hdul.append(HA6562_SFR_SF_corr_hdu)

    hdul.flush()
```

# 10 Surface-density $\log\left(\Sigma_{SFR}[M_\odot/yr/kpc^2]\right)$

Extract pixel scale and convert to unit of kpc, then calculate the SFR surface density.

```
pix_scale =
(proj_plane_pixel_scales(WCS(gas_header).celestial)*u.deg).to(u.arc
sec)
pix_area  = (pix_scale[0].to(u.rad).value * 16.5*u.Mpc) *
(pix_scale[1].to(u.rad).value * 16.5*u.Mpc)
pix_area  = pix_area.to(u.kpc**2)

Sigma_SFR = SFR_map_SF / pix_area
log_Sigma = np.log10(Sigma_SFR.value)
```

The SF-only surface-density map is stored:

```
with fits.open(out_path, mode="append") as hdul:            # open
existing file
    log_SFR_surface_density_hdu = fits.ImageHDU(
        data=log_SFR_surface_density.astype(np.float64),  # like
others
        header=gas_header, name="LOGSFR_SURFACE_DENSITY")  # new
HDU name
    log_SFR_surface_density_hdu.header["BUNIT"] =
"log(M_sun/yr/kpc2)"  # units keyword
    hdul.append(log_SFR_surface_density_hdu)

    hdul.flush()
```