20250620 Quality Control Cut

This time I use NGC4298 as the example.

0. No Foreground Stars

I use a simple mask (based on median$\pm 5\sigma$ outliters in velocities or velocity dispersions in `*KIN_map.fits`) to remove the foreground stars in NGC4064, NGC4298 and NGC4694.

1. Line to SFR Flowchart

Now I rewrite the extraction of SFR into a flowchart with multiple masks that can be applied at any time.

1.1 Flowchart

1. Data. The input is the FLUX and $\mathrm{FLUX}_{\mathrm{ERR}}$ of Hα, Hβ, [O III], [NII] and [SII] doublets.
2. Balmer Decrement (BD). Only those spaxels with $BD < 2.86$ will be set to 2.86.
3. $E(B-V)_{\mathrm{BD}}$. Get gas galactic extinction.
4. $\mathrm{FLUX}_{\mathrm{corr}}$. Use $E(B-V)_{\mathrm{BD}}$ to get corrected fluxes of each lines.
5. L_{corr}. Get corrected luminosity by assuming distance at 16.5 Mpc.
6. SFR. Get SFR by Calzetti 2000, with a convertion from Kroupa to Chabrier IMF.
7. Σ_{SFR}. Get SFR surface density.

1.2 Masks

There are two types of masks.

1.2.1 S/N cut

The quality control is applied to observed flux of each line. The value of this cut any be any positive value. Then for each line, I have masks `above` or `below` for $\mathrm{FLUX}/\mathrm{FLUX}_{\mathrm{ERR}}$ higher than that value.

```python
def apply_cut(cut=0):
    cut_above = {
        'HB4861': (HB4861_FLUX / HB4861_FLUX_ERR) >= cut,
        'HA6562': (HA6562_FLUX / HA6562_FLUX_ERR) >= cut,
        'OIII5006': (OIII5006_FLUX / OIII5006_FLUX_ERR) >= cut,
        'NII6583': (NII6583_FLUX / NII6583_FLUX_ERR) >= cut,
        'SII6716': (SII6716_FLUX / SII6716_FLUX_ERR) >= cut,
        'SII6730': (SII6730_FLUX / SII6730_FLUX_ERR) >= cut
    }
    cut_below = {
        'HB4861': (HB4861_FLUX / HB4861_FLUX_ERR) < cut,
        'HA6562': (HA6562_FLUX / HA6562_FLUX_ERR) < cut,
        'OIII5006': (OIII5006_FLUX / OIII5006_FLUX_ERR) < cut,
        'NII6583': (NII6583_FLUX / NII6583_FLUX_ERR) < cut,
        'SII6716': (SII6716_FLUX / SII6716_FLUX_ERR) < cut,
        'SII6730': (SII6730_FLUX / SII6730_FLUX_ERR) < cut
    }
    return cut_above, cut_below
```

Hence, I have $2 \times 6 = 12$ basic masks in quality control.

```python
cut_above, cut_below = apply_cut(cut)

# Extract individual masks
HB4861_FLUX_above = cut_above['HB4861']
HB4861_FLUX_below = cut_below['HB4861']
HA6562_FLUX_above = cut_above['HA6562']
HA6562_FLUX_below = cut_below['HA6562']
OIII5006_FLUX_above = cut_above['OIII5006']
OIII5006_FLUX_below = cut_below['OIII5006']
NII6583_FLUX_above = cut_above['NII6583']
NII6583_FLUX_below = cut_below['NII6583']
```

```
SII6716_FLUX_above = cut_above['SII6716']
SII6716_FLUX_below = cut_below['SII6716']
SII6730_FLUX_above = cut_above['SII6730']
SII6730_FLUX_below = cut_below['SII6730']
```

Then I can combine them to have:

```
# create a mask for Balmer Decrement (BD), i.e. cut at HB4861_FLUX
and HA6562_FLUX, call it "mask_BD  "
mask_BD = HB4861_FLUX_above & HA6562_FLUX_above
# create a mask for BPT diagram, i.e., "mask_BD" as well as
OIII5006_FLUX, NII6583_FLUX, SII6716_FLUX, SII6730_FLUX, call it
"mask_BPT"
mask_BPT = mask_BD & OIII5006_FLUX_above & NII6583_FLUX_above &
SII6716_FLUX_above & SII6730_FLUX_above
# create a mask for NII BPT, i.e., "mask_BD" as well as
OIII5006_FLUX, NII6583_FLUX, call it "mask_NII_BPT"
mask_NII_BPT = mask_BD & OIII5006_FLUX_above & NII6583_FLUX_above
# create a mask for SII BPT, i.e., "mask_BD" as well as
OIII5006_FLUX, SII6716_FLUX, SII6730_FLUX, call it "mask_SII_BPT"
mask_SII_BPT = mask_BD & OIII5006_FLUX_above & SII6716_FLUX_above &
SII6730_FLUX_above
# create a mask for BPT diagram, but exclude the OIII5006_FLUX,
i.e., "mask_BD" as well as NII6583_FLUX, SII6716_FLUX,
SII6730_FLUX, call it "mask_BPT_no_OIII"
mask_BPT_no_OIII = mask_BD & NII6583_FLUX_above &
SII6716_FLUX_above & SII6730_FLUX_above
# create a mask for BPT diagram, but exclude the NII6583_FLUX,
i.e., "mask_BD" as well as OIII5006_FLUX, SII6716_FLUX,
SII6730_FLUX, call it "mask_BPT_no_NII"
mask_BPT_no_NII = mask_BD & OIII5006_FLUX_above &
SII6716_FLUX_above & SII6730_FLUX_above
# create a mask for BPT diagram, but exclude both SII6716_FLUX and
SII6730_FLUX, i.e., "mask_BD" as well as OIII5006_FLUX,
NII6583_FLUX, call it "mask_BPT_no_SII"
mask_BPT_no_SII = mask_BD & OIII5006_FLUX_above &
NII6583_FLUX_above
```

1.2.2 SF selection

Based on corrected flux of each line, I can use [N II] and [SII] BPT diagrams to select the SF spaxels.

```
# ---- line ratios ---------------------------------------------------
--
logN2  = np.log10(NII6583_FLUX_corr / HA6562_FLUX_corr)        # [N
II]/Hα
logS2  = np.log10((SII6716_FLUX_corr+SII6730_FLUX_corr) /
HA6562_FLUX_corr)   # Σ[S II]/Hα
logO3  = np.log10(OIII5006_FLUX_corr / HB4861_FLUX_corr)       # [O
III]/Hβ

#  N II BPT -----------------------------------------
def kewley01_N2(x):   # max-starburst
    return 0.61/(x-0.47) + 1.19
def kauff03_N2(x):    # empirical SF upper envelope
    return 0.61/(x-0.05) + 1.30

#  S II BPT -----------------------------------------
def kewley01_S2(x):
    return 0.72/(x-0.32) + 1.30
def kewley06_Sy_LIN(x):   # Seyfert/LINER division
    return 1.89*x + 0.76

# Define a function to apply the BPT masks,
# the BPT masks are to find the HII, Comp, and AGN regions in NII
BPT,
# while the HII, LINER, and Seyfert regions in SII BPT,
respectively.
def apply_bpt_masks(logN2, logS2):
    # NII BPT masks
    mask_N2_HII = logN2 < kewley01_N2(logS2)
    mask_N2_Comp = (logN2 >= kewley01_N2(logS2)) & (logN2 <
kauff03_N2(logS2))
    mask_N2_AGN = logN2 >= kauff03_N2(logS2)
```

```
    # SII BPT masks
    mask_S2_HII = logS2 < kewley01_S2(logN2)
    mask_S2_LINER = (logS2 >= kewley01_S2(logN2)) & (logS2 <
kewley06_Sy_LIN(logN2))
    mask_S2_Seyfert = logS2 >= kewley06_Sy_LIN(logN2)

    return (mask_N2_HII, mask_N2_Comp, mask_N2_AGN), (mask_S2_HII,
mask_S2_LINER, mask_S2_Seyfert)

# Apply the BPT masks
masks_N2, masks_S2 = apply_bpt_masks(logN2, logS2)
mask_N2_HII, mask_N2_Comp, mask_N2_AGN = masks_N2
mask_S2_HII, mask_S2_LINER, mask_S2_Seyfert = masks_S2
```

Similarly, I can combine basic masks of these 6 regions to have different definitions of SF:

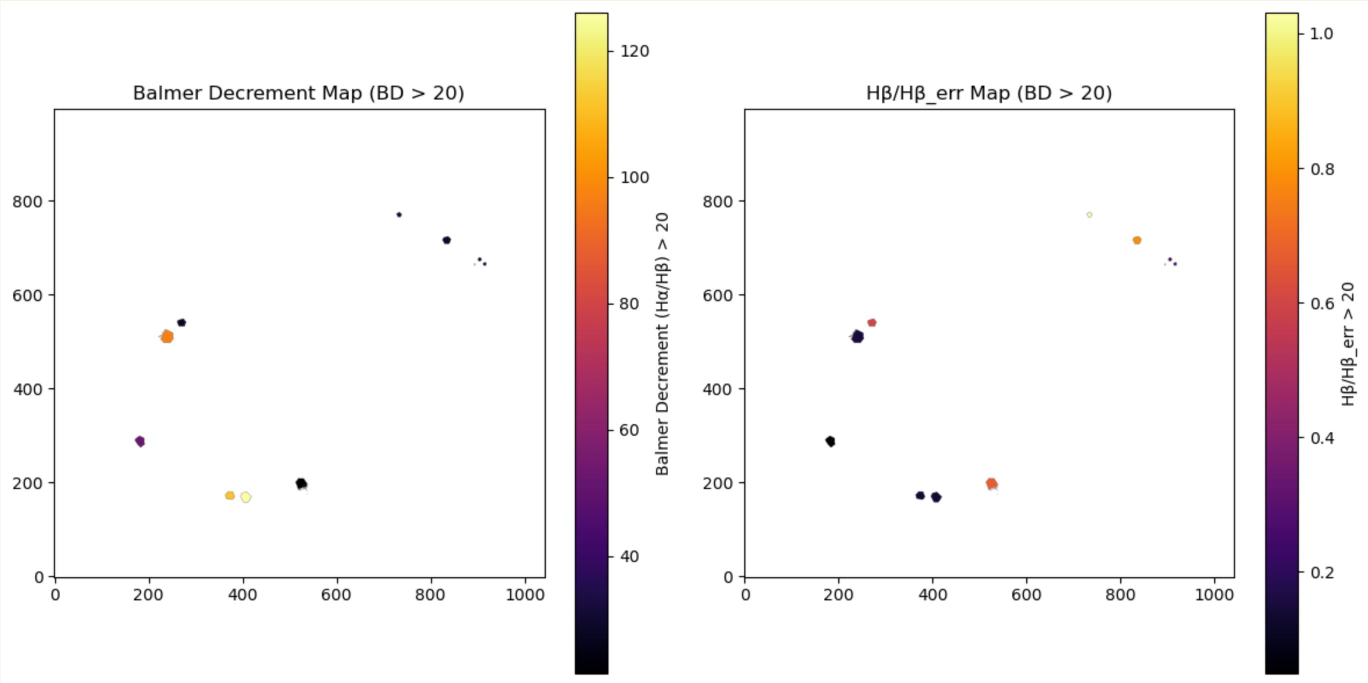
```
# Define a function to get SF mask under different conditions:
# 'both': (mask_N2_HII+mask_N2_Comp)*mask_S2_HII;
# 'either': (mask_N2_HII+mask_N2_Comp) | mask_S2_HII;
# 'NII': (mask_N2_HII+mask_N2_Comp);
# 'SII': mask_S2_HII.
# and also for and non-SF mask, i.e., 'non-SF', which is the
opposite of SF mask.
def get_sf_mask(mask_N2_HII = mask_N2_HII, mask_S2_HII =
mask_S2_HII, mask_N2_Comp = mask_N2_Comp,
                mask_S2_LINER = mask_S2_LINER, mask_N2_AGN =
mask_N2_AGN, mask_S2_Seyfert = mask_S2_Seyfert,
                condition='both'):
    if condition == 'both':
        SF = (mask_N2_HII | mask_N2_Comp) & (mask_S2_HII)
        non_SF = ~(SF)
        return SF, non_SF
    elif condition == 'either':
        SF = (mask_N2_HII | mask_N2_Comp) | mask_S2_HII
        non_SF = ~(SF)
        return SF, non_SF
    elif condition == 'NII':
        SF = mask_N2_HII | mask_N2_Comp
```

```
        non_SF = ~(SF)
        return SF, non_SF
    elif condition == 'SII':
        SF = mask_S2_HII
        non_SF = ~(SF)
        return SF, non_SF

# Get the SF mask under different conditions
SF_mask_both, non_SF_mask_both = get_sf_mask(condition='both')
SF_mask_either, non_SF_mask_either =
get_sf_mask(condition='either')
SF_mask_NII, non_SF_mask_NII = get_sf_mask(condition='NII')
SF_mask_SII, non_SF_mask_SII = get_sf_mask(condition='SII')
```

2. Some unrealistic BD values

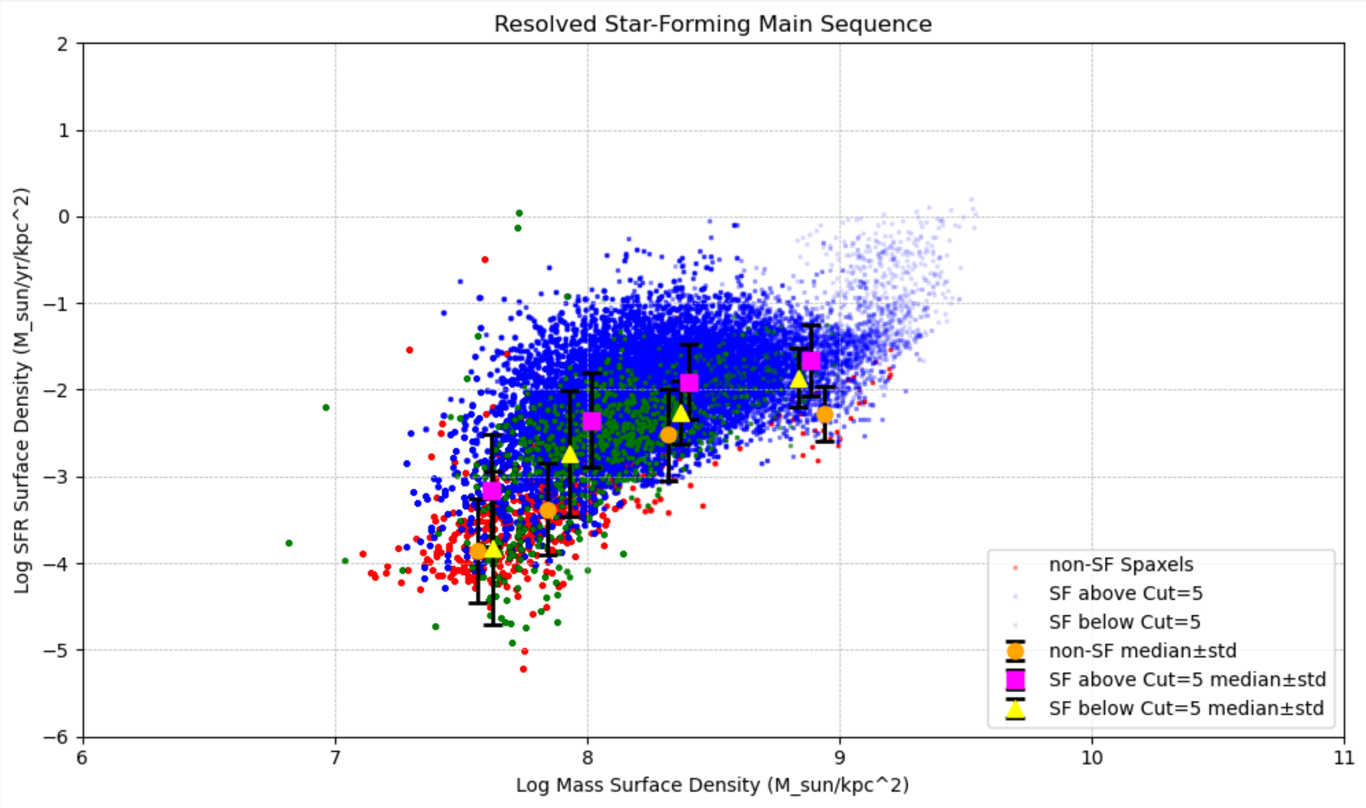
I notice that some BD values can go up to ~100. Probably they are because the `pPXF` fails to fit Hβ line properly. And I have confirmed that they will be excluded by the cut at Hβ line (see figure below).

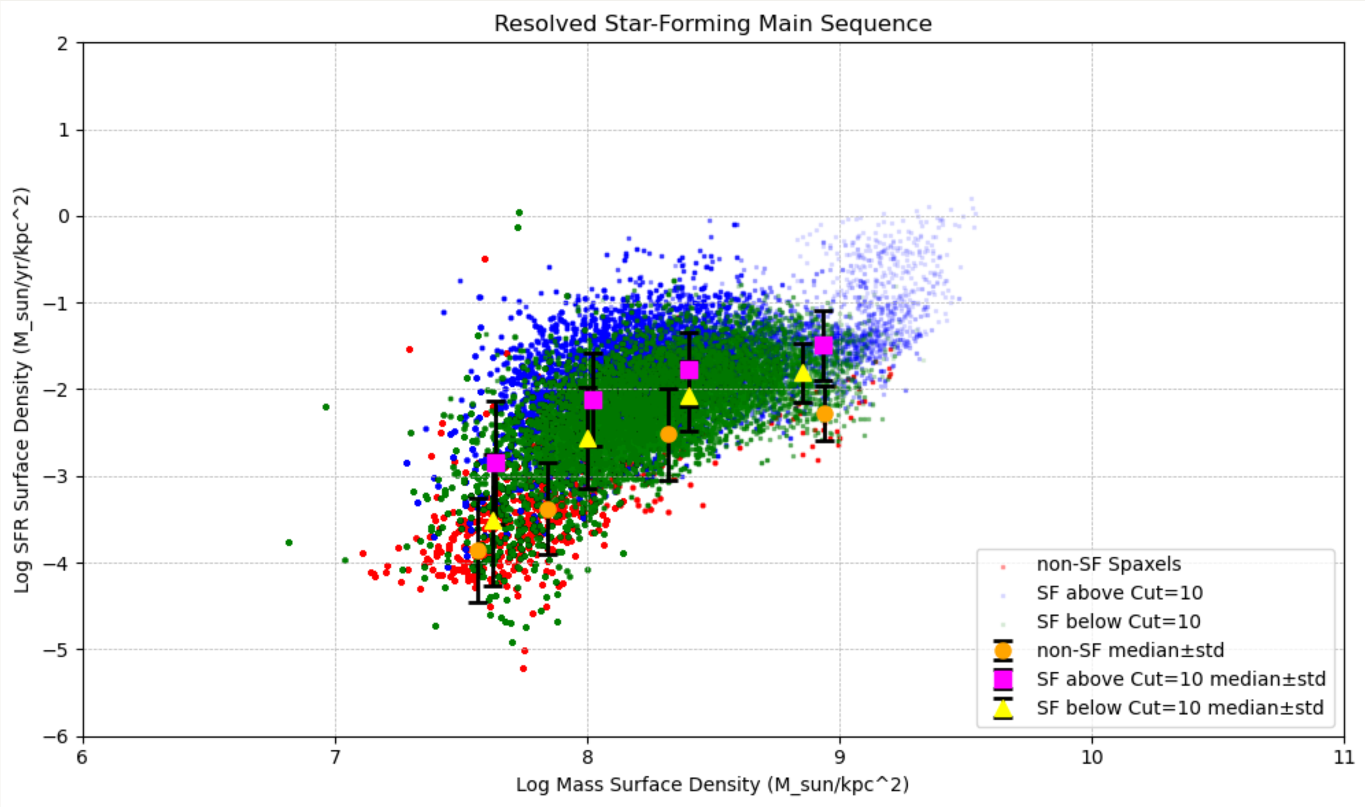


For example, those BD > 20 are actually $\mathrm{S/N} \lesssim 1$.

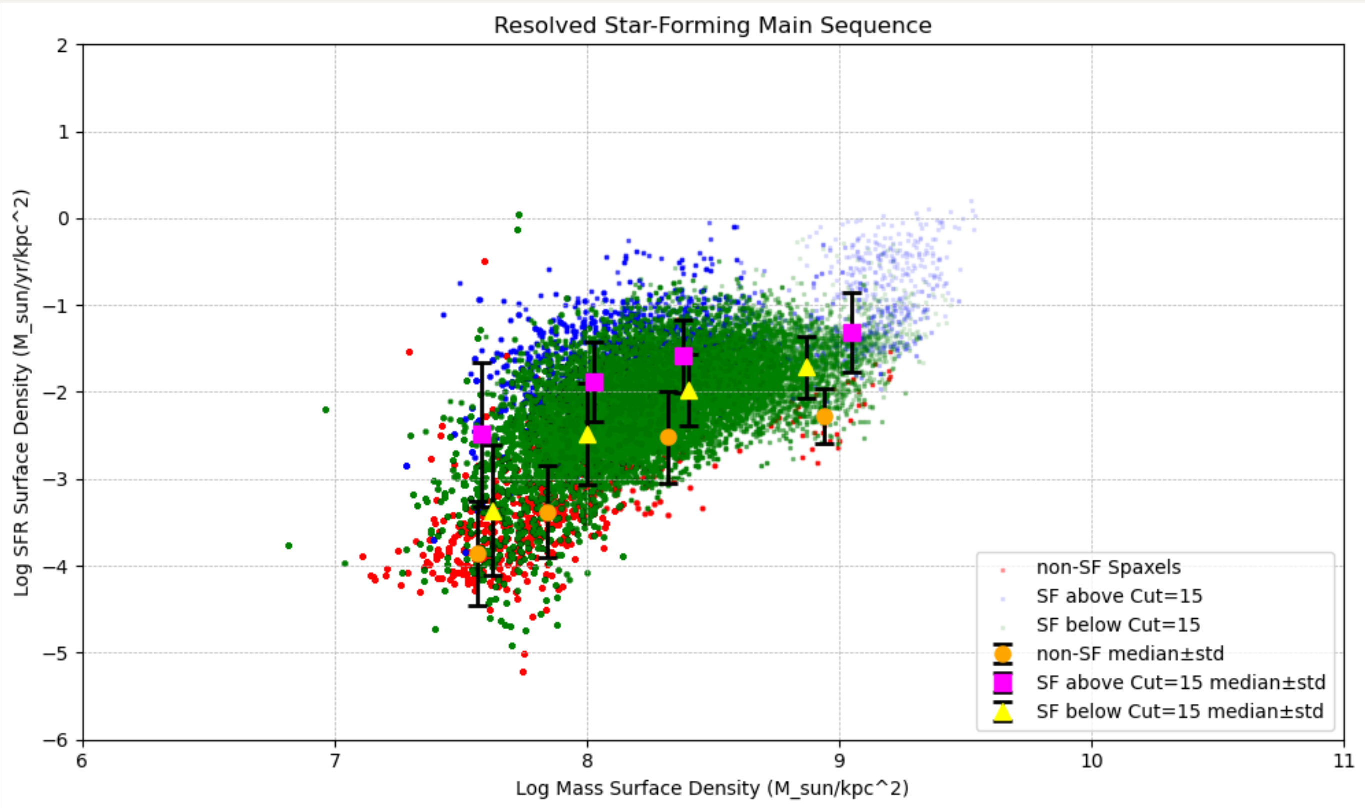
3. Different cut values

Below I show the resolved SFMS under different cut values: 5, 10, 15, 20, 25. Blue dots are "true" SF spaxels (applying the masks `SF_mask_both` & `mask_BPT`), green dots are "potential" SF spaxels (applying the masks `SF_mask_both` & `~mask_BPT`), and red dots are other spaxels that are definitely not SF, respectively.

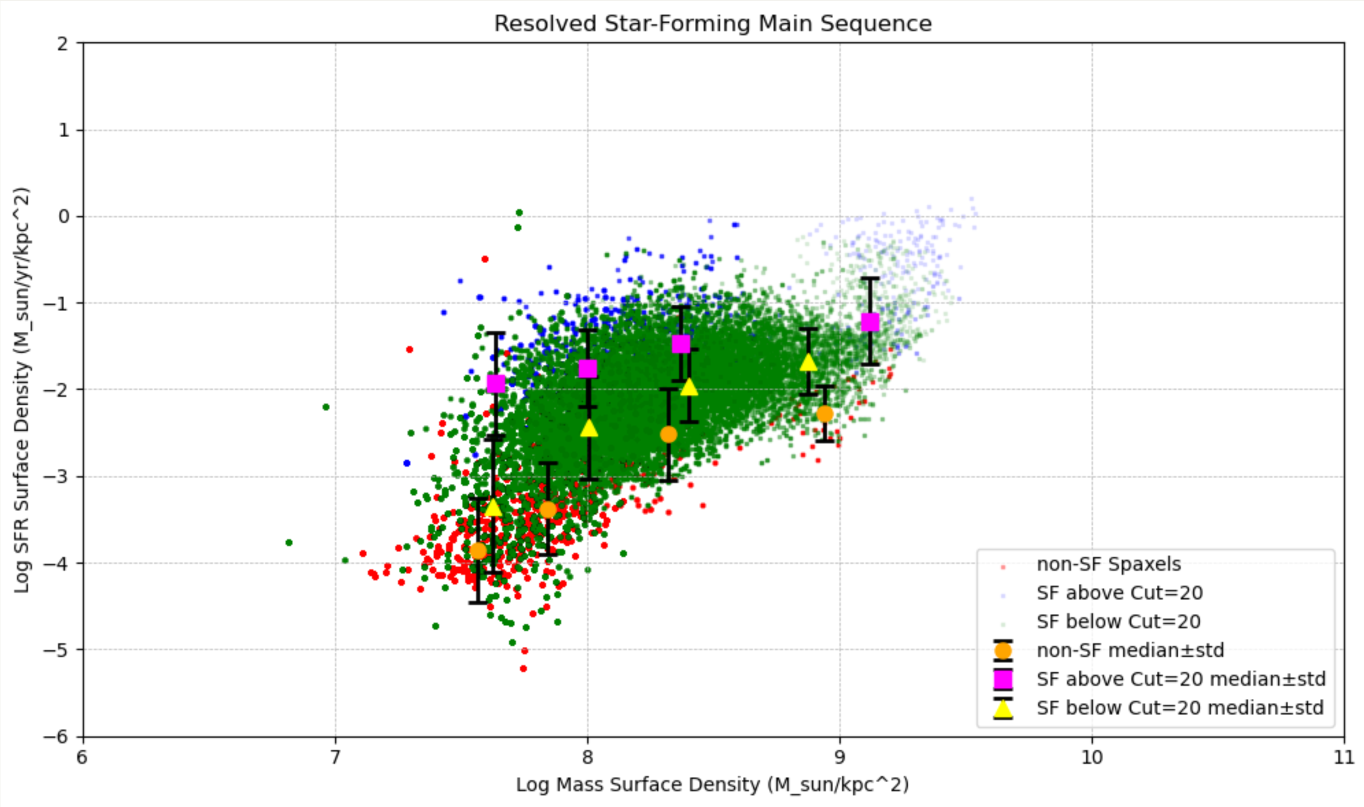


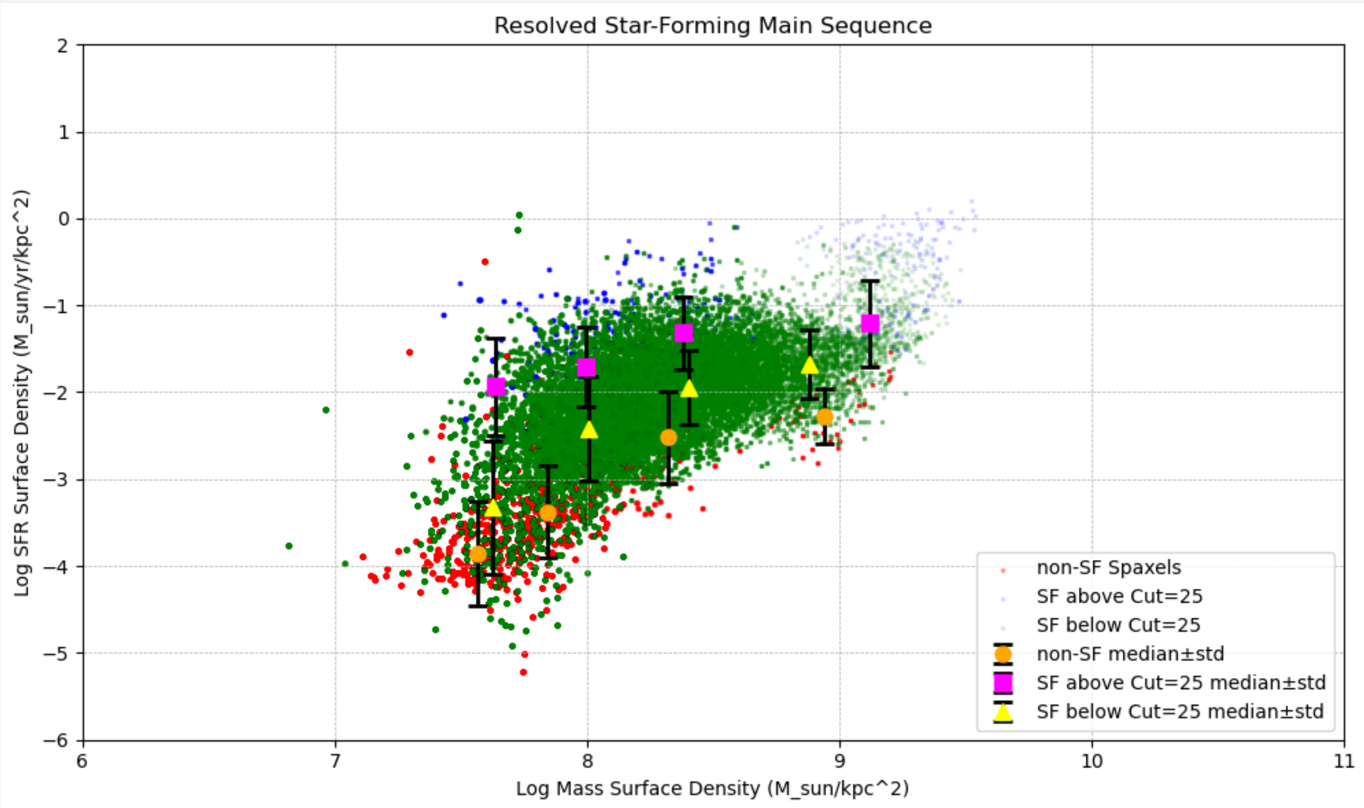


Resolved Star-Forming Main Sequence
Log SFR Surface Density (M_sun/yr/kpc^2)
Log Mass Surface Density (M_sun/kpc^2)
non-SF Spaxels
SF above Cut=10
SF below Cut=10
non-SF median±std
SF above Cut=10 median±std
SF below Cut=10 median±std



Resolved Star-Forming Main Sequence
Log SFR Surface Density (M_sun/yr/kpc^2)
Log Mass Surface Density (M_sun/kpc^2)
non-SF Spaxels
SF above Cut=15
SF below Cut=15
non-SF median±std
SF above Cut=15 median±std
SF below Cut=15 median±std

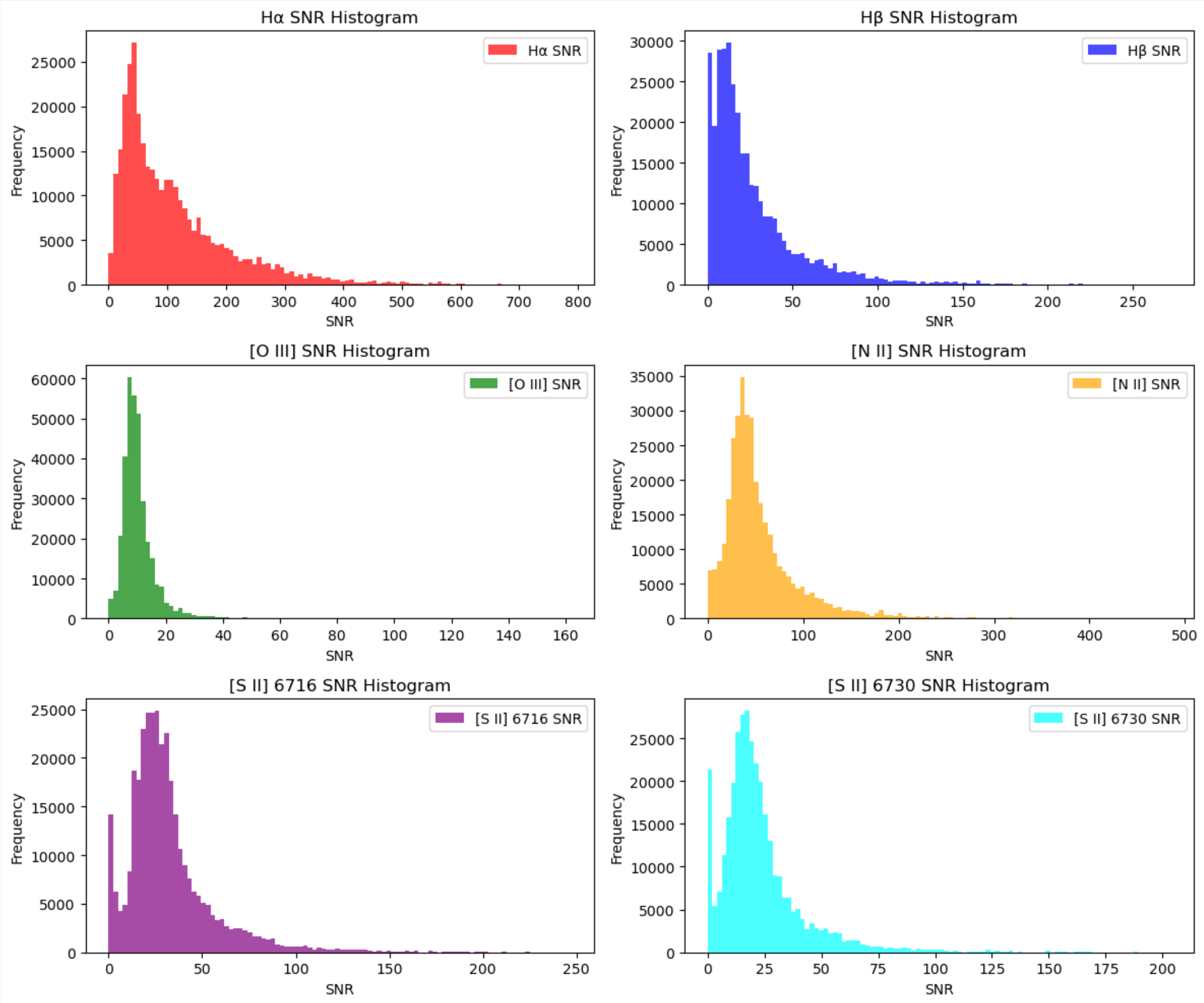




The purpose of this plot is to show what happened when applying the $\mathrm{Flux}/\mathrm{Flux}_{\mathrm{err}}$ cut and see if this cut biases our resolved SFMS. It seems that the green dots eroses the blue dots as the value increases. And my answer is that the quality control cut will make our results reflect the upper limit if the cut is "high" (probably higher than 10 in this galaxy).

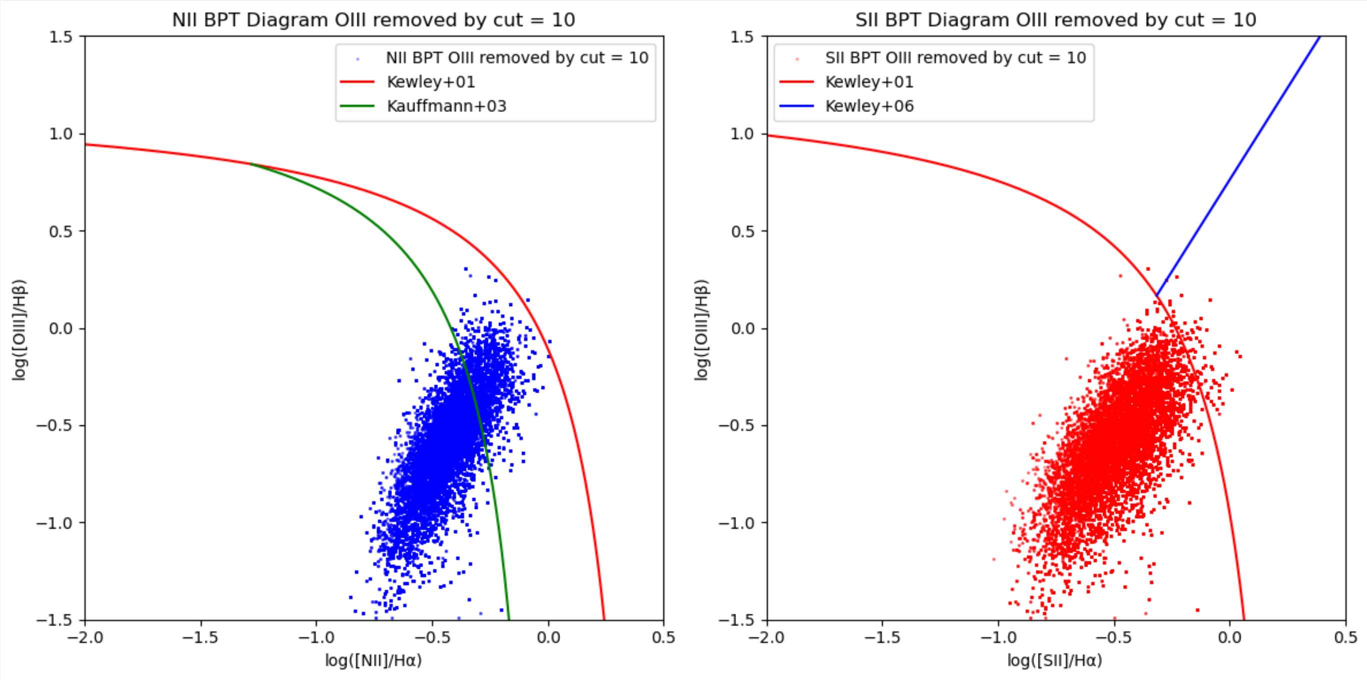
5. Cut on [O III]

Another question is that if uniformly applying same cut on each line is too conservative.

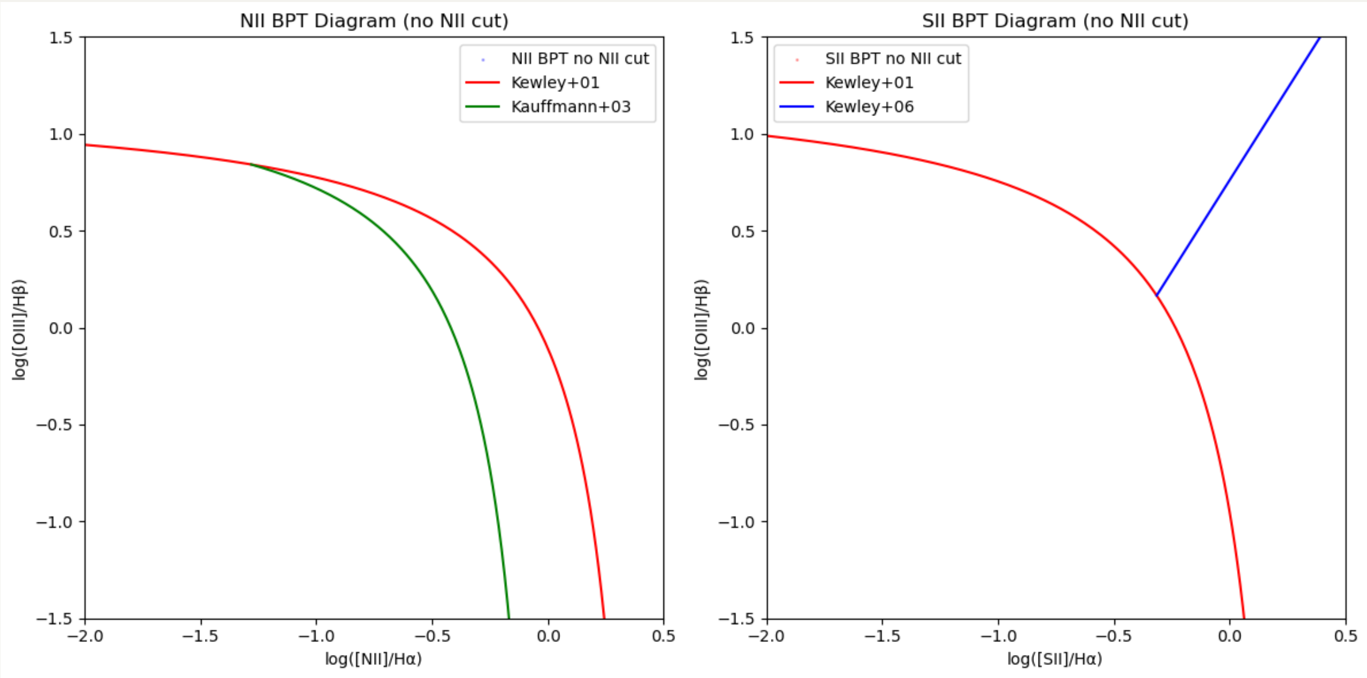


Based on the histogram of S/N of each line, [OIII] is clearly the worst.

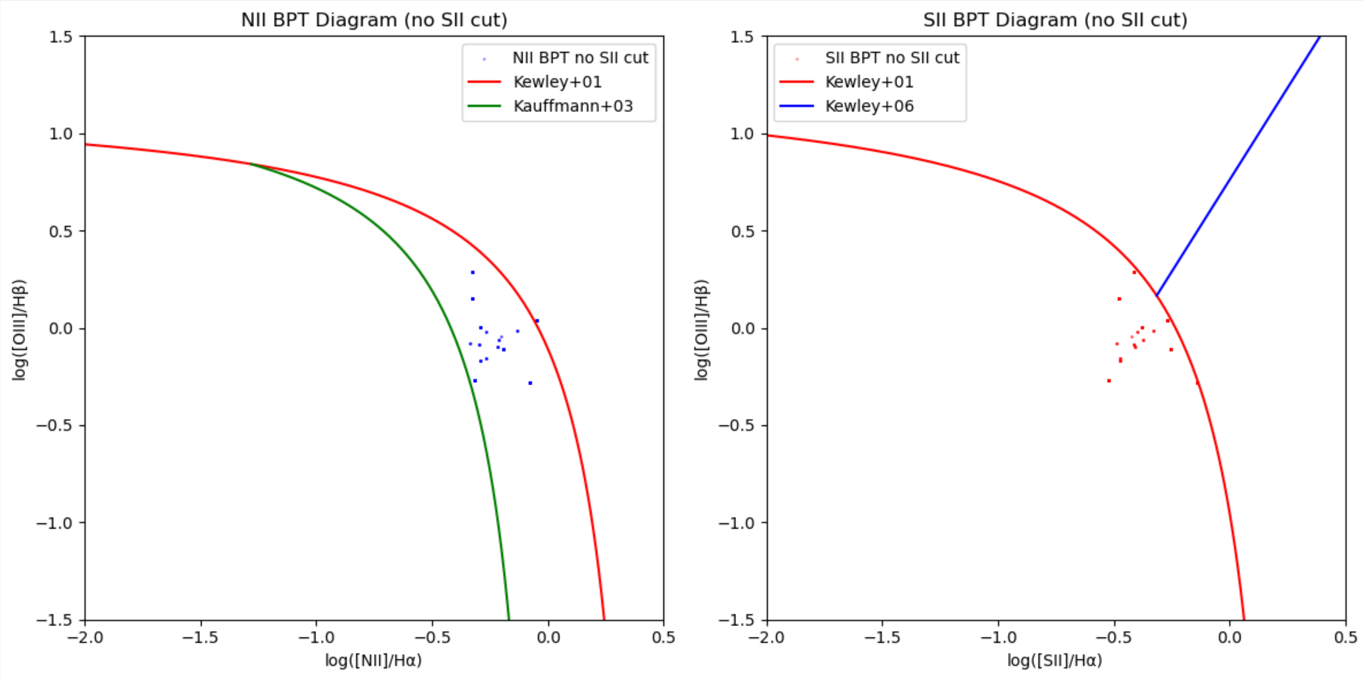
And in fact, if I fix the cut at 10 and plot all the spaxels applying the cut except the [O III] line (i.e., those removed by cut due to low S/N in [OIII]), most of them are actually SF:



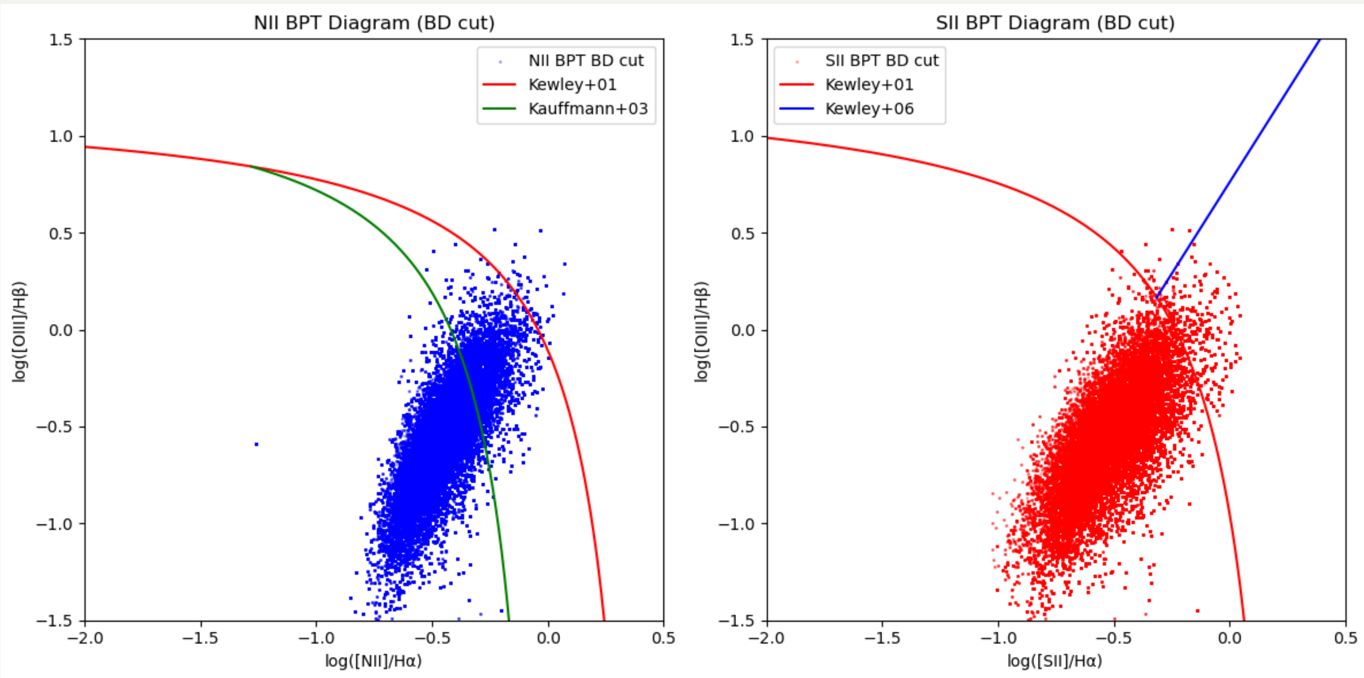
Similarly, I also show the spaxels that are removed by cut due to low S/N in [NII]. And it turns out there are zero, so that means when quality control cut is already appllied to other lines, they already removed those low S/N in [NII].



Simiarly, here is for [S II] and they are near the edge of the SF boundaries:



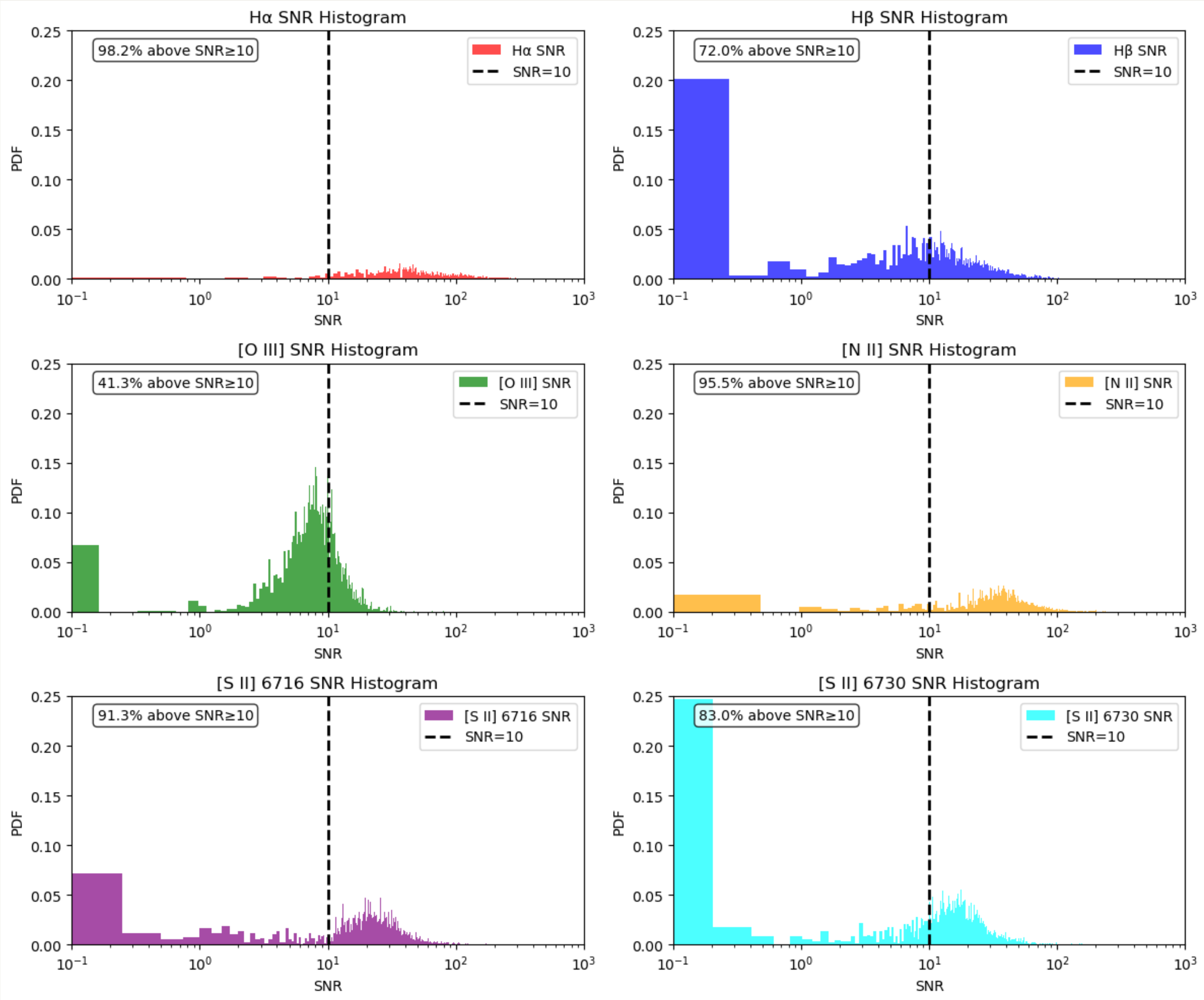
Then ,below is the case that I only apply the cut on balmer lines. Here, I am thinking of maybe cut on balmer lines (or even just the Hβ line) is enough.



My idea is that we should go back to look at the error propogations in BPT diagrams. BPT coordinates are $\log([\mathrm{NII}]/\mathrm{H}\alpha)$ and $\log([\mathrm{OIII}]/\mathrm{H}\beta)$, so the error of the ratios will be

$$\delta \log(\mathrm{ratio}) = \frac{1}{\ln 10}\sqrt{(\frac{\mathrm{ERR}_{\mathrm{numerator}}}{\mathrm{FLUX}_{\mathrm{numerator}}})^2 + (\frac{\mathrm{ERR}_{\mathrm{denominator}}}{\mathrm{FLUX}_{\mathrm{denominator}}})^2} \tag{1}$$

The $\frac{\mathrm{ERR}}{\mathrm{FLUX}}$ is exaclty the inverse of S/N. That means the unvertainties of the ratios is determined by those lower S/N lines.



The figure is the the PDF of each line. Clearly, we need to be careful with Hβ and [O III] lines' S/N.