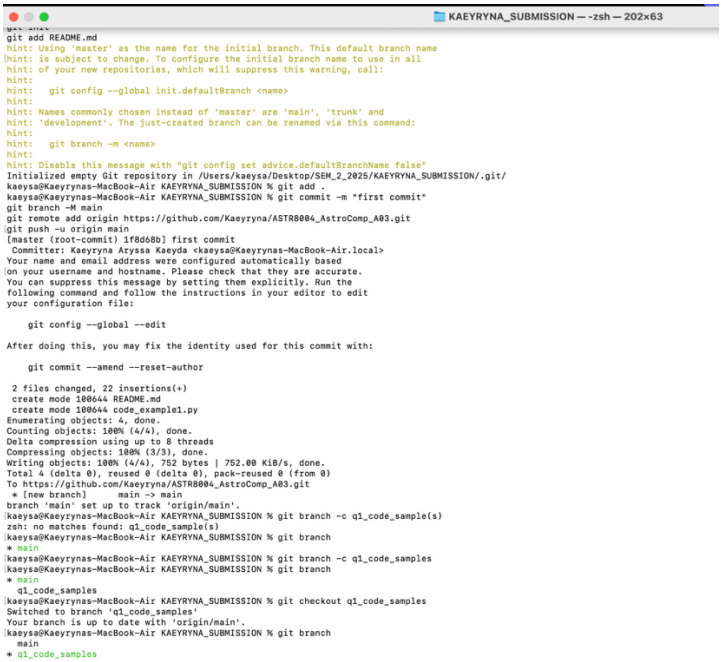
ASTRONOMICAL COMPUTING ASSIGNMENT 03 (U8041307)

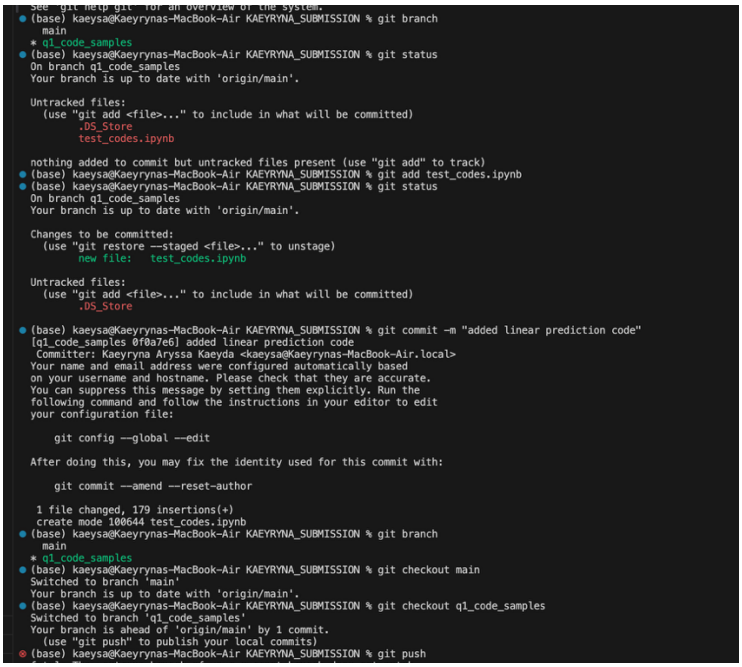
QUESTION 1

Below are the commands used to complete task 1:

In order to do this, I've ensured that the working direcotory is in the folder with the submission files that will be uploaded to github.

Link to github: https://github.com/Kaeyryna/ASTR8004_AstroComp_A03





```
(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION % git branch
  main
* q1_code_samples
(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION % git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION % git checkout q1_code_samples
Switched to branch 'q1_code_samples'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION % git push
fatal: The upstream branch of your current branch does not match
the name of your current branch.  To push to the upstream branch
on the remote, use

    git push origin HEAD:main

To push to the branch of the same name on the remote, use

    git push origin HEAD

To choose either option permanently, see push.default in 'git help config'.

To avoid automatically configuring an upstream branch when its name
won't match the local branch, see option 'simple' of branch.autoSetupMerge
in 'git help config'.

(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION % git push origin HEAD
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 20.88 KiB | 20.88 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'q1_code_samples' on GitHub by visiting:
remote:      https://github.com/Kaeyryna/ASTR8004_AstroComp_A03/pull/new/q1_code_samples
remote:
To https://github.com/Kaeyryna/ASTR8004_AstroComp_A03.git
 * [new branch]      HEAD -> q1_code_samples
(base) kaeysa@Kaeyrynas-MacBook-Air KAEYRYNA_SUBMISSION %
```

QUESTION 2

Before starting, Gaia is imported using the following line in python:

```
from astroquery.gaia import Gaia
```
✓ 2.0s

The following are the query used to complete task 2

2.1.1: Download all stars within 1 degree of the center of Ruprecht 147 that are brighter than G = 14 in Gaia DR3 (the table is called gaiadr.gaia source) and include a crossmatch these stars with the 2MASS catalog and report your ADQL query text (use the """query""" notation)

```
#Q2.1 USETHIS
query = f"""
SELECT source_id, ra, dec, parallax, phot_g_mean_mag

FROM gaiadr3.gaia_source AS gaia

WHERE DISTANCE(POINT(289.074, -16.323), POINT(ra, dec)) <= 1
    AND phot_g_mean_mag < 14
"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])
print("Number of stars queried: ", len(query_result))
```
✓ 10.2s

This query resulted to 3663 stars.

Then, to include crossmatching the query is as of:

```
#Q2.1_include_crossmatch USETHIS

query = f"""
SELECT gaia.solution_id, gaia.source_id, gaia.ra, gaia.dec, gaia.parallax, gaia.phot_g_mean_mag, xmatch.original_ext_source_id,
xmatch.clean_tmass_psc_xsc_oid,xjoin.clean_tmass_psc_xsc_oid

FROM gaiadr3.gaia_source AS gaia
JOIN gaiaedr3.tmass_psc_xsc_best_neighbour AS xmatch USING(source_id)
JOIN gaiaedr3.tmass_psc_xsc_join AS xjoin USING(clean_tmass_psc_xsc_oid)
JOIN gaiadr1.tmass_original_valid AS tmass
    ON xjoin.original_psc_source_id = tmass.designation

WHERE DISTANCE(POINT(289.074, -16.323), POINT(gaia.ra, gaia.dec)) < 1
    AND phot_g_mean_mag < 14
"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])
print("Number of stars queried: ", len(query_result))
✓ 28.9s
```

The numbers of stars queried is 3637 after crossmatching.

2.1.2: Determine how many stars are returned from the initial query

From the above query, we had obtained the total number of stars queried **before crossmatching to be 3663 stars and after crossmatching, the number of stars queried is 3637**. The way we queried the number of stars are as of the above, but if we're querying straight away in Gaia Database website, we can also do:

```
1 SELECT source_id, COUNT(source_id) AS star_count
2 FROM job_upload."job17594268444300"
3 GROUP BY source_id
```

2.1.3: Identify any stars with bad 2MASS photometry, where ph qual is not 'AAA'.

```
#Q2.3 PH_QUALTIY <> 'AAA'

query = f"""
SELECT gaia.solution_id, gaia.source_id, gaia.ra, gaia.dec, gaia.parallax, gaia.phot_g_mean_mag, xmatch.original_ext_source_id,
xmatch.clean_tmass_psc_xsc_oid,xjoin.clean_tmass_psc_xsc_oid, tmass.ph_qual

FROM gaiadr3.gaia_source AS gaia
JOIN gaiaedr3.tmass_psc_xsc_best_neighbour AS xmatch USING(source_id)
JOIN gaiaedr3.tmass_psc_xsc_join AS xjoin USING(clean_tmass_psc_xsc_oid)
JOIN gaiadr1.tmass_original_valid AS tmass
    ON xjoin.original_psc_source_id = tmass.designation

WHERE DISTANCE(POINT(289.074, -16.323), POINT(gaia.ra, gaia.dec)) < 1
    AND phot_g_mean_mag < 14
    AND tmass.ph_qual <> 'AAA'

"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])
print("Number of stars queried: ", len(query_result))
✓ 9.0s
```

This query resulted to 79 stars

2.1.4: Identify any stars with negative (or non-positive) parallaxes in the Gaia data.

```
#Q2.4 Gaia only parallax, non positive parallax

query = f"""
SELECT gaia.solution_id, gaia.source_id, gaia.ra, gaia.dec, gaia.parallax, gaia.phot_g_mean_mag, xmatch.original_ext_source_id,
xmatch.clean_tmass_psc_xsc_oid,xjoin.clean_tmass_psc_xsc_oid, tmass.ph_qual

FROM gaiadr3.gaia_source AS gaia
JOIN gaiaedr3.tmass_psc_xsc_best_neighbour AS xmatch USING(source_id)
JOIN gaiaedr3.tmass_psc_xsc_join AS xjoin USING(clean_tmass_psc_xsc_oid)
JOIN gaiadr1.tmass_original_valid AS tmass
    ON xjoin.original_psc_source_id = tmass.designation

WHERE DISTANCE(POINT(289.074, -16.323), POINT(gaia.ra, gaia.dec)) < 1
    AND phot_g_mean_mag < 14
    AND gaia.parallax < 0

"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])
print("Number of stars queried: ", len(query_result))
```
✓ 8.8s

This query resulted to 5 stars.

2.1.5: Apply these two quality cuts (removing stars with bad 2MASS photometry and non- positive parallaxes). After applying the cuts, determine how many stars remain.

```
#q2.5

query = f"""
SELECT gaia.solution_id, gaia.source_id, gaia.ra, gaia.dec, gaia.parallax, gaia.phot_g_mean_mag, xmatch.original_ext_source_id,
xmatch.clean_tmass_psc_xsc_oid,xjoin.clean_tmass_psc_xsc_oid, tmass.ph_qual

FROM gaiadr3.gaia_source AS gaia
JOIN gaiaedr3.tmass_psc_xsc_best_neighbour AS xmatch USING(source_id)
JOIN gaiaedr3.tmass_psc_xsc_join AS xjoin USING(clean_tmass_psc_xsc_oid)
JOIN gaiadr1.tmass_original_valid AS tmass
    ON xjoin.original_psc_source_id = tmass.designation

WHERE DISTANCE(POINT(289.074, -16.323), POINT(gaia.ra, gaia.dec)) < 1
    AND phot_g_mean_mag < 14
    AND tmass.ph_qual = 'AAA'
    AND gaia.parallax > 0

"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])
print("Number of stars queried: ", len(query_result))
```
✓ 12.4s

The query resulted to 3535 stars.

2.1.6: Using the remaining stars, generate a figure with two panels

To get the data of the query, the following line is used:

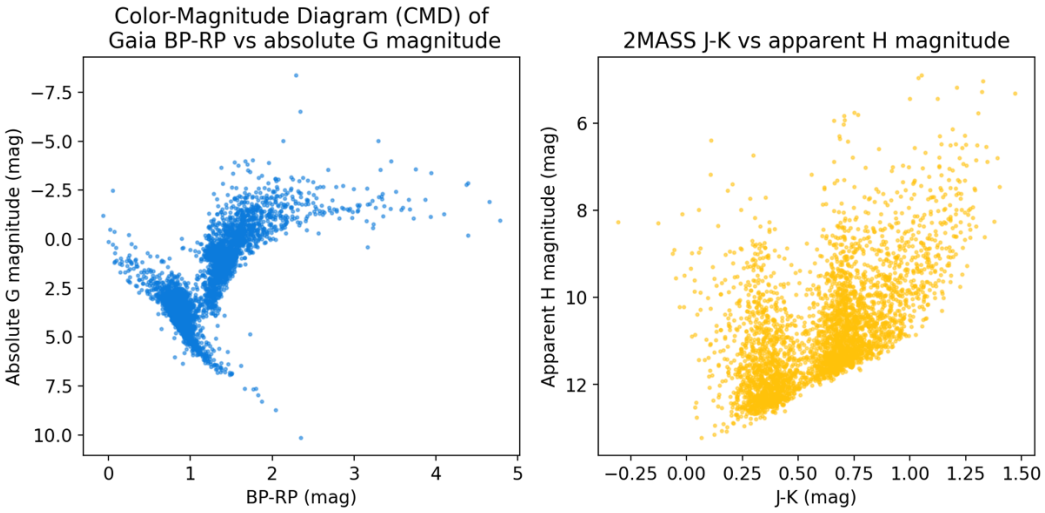
```
import pandas as pd

query = f"""
SELECT gaia.solution_id, gaia.source_id, gaia.ra, gaia.dec, gaia.parallax, gaia.bp_rp, gaia.phot_g_mean_mag, xmatch.original_ext_source_id,
xmatch.clean_tmass_psc_xsc_oid,xjoin.clean_tmass_psc_xsc_oid, tmass.ph_qual, tmass.j_m, tmass.ks_m, tmass.h_m

FROM gaiadr3.gaia_source AS gaia
JOIN gaiaedr3.tmass_psc_xsc_best_neighbour AS xmatch USING(source_id)
JOIN gaiaedr3.tmass_psc_xsc_join AS xjoin USING(clean_tmass_psc_xsc_oid)
JOIN gaiadr1.tmass_original_valid AS tmass
    ON xjoin.original_psc_source_id = tmass.designation

WHERE DISTANCE(POINT(289.074, -16.323), POINT(gaia.ra, gaia.dec)) < 1
    AND phot_g_mean_mag < 14
    AND tmass.ph_qual = 'AAA'
    AND gaia.parallax > 0
"""

job = Gaia.launch_job_async(query)
query_result = job.get_results()
print(query_result[:])

df = query_result.to_pandas()
df.to_csv('2.6_plot2_J-KvsH-result.csv', index=False)
✓ 11.4s
```

a) A color-magnitude diagram (CMD) of Gaia BP-RP vs. absolute G magnitude.
b) A 2MASS J-Ks vs. apparent H magnitude diagram.

To complete the above, the following lines were used, and the figure is saved with a resolution of 200dpi under the name ‘cmds_R147.png’ with the extension .png:

```
#KAEYRYNA ARYSSA BINTI KAEYDA ASTR8004 ASSIGNMENT 03

import pandas as pd
import matplotlib.pyplot as plt
import math
import numpy as np

plt.rcParams['font.size'] = 12

colmag = pd.read_csv('2.6_plot1_CMD-result.csv')
tmass = pd.read_csv('2.6_plot2_J-KvsH-result.csv')

#data to plot CMD
bprp = np.array(colmag['bp_rp'])
abs_MG = colmag['phot_g_mean_mag'] + 5 - 5*np.log10(1000/ colmag['parallax'])

#data to plot from tmass
j_k = tmass['j_m'] - tmass['ks_m']
arr_j_k = np.array(j_k)
hmag = np.array(tmass['h_m'])

#create the figure to plot 2 panels
fig, axs = plt.subplots(1,2, figsize=(10, 5))

#Panel 1
ax = axs[0]
ax.plot(bprp, abs_MG, 'o', markersize=2, alpha=0.5, color='#0C7BDC')
ax.set_title('Color-Magnitude Diagram (CMD) of\n Gaia BP-RP vs absolute G magnitude')
ax.set_ylabel('Absolute G magnitude (mag)')
ax.set_xlabel('BP-RP (mag)')
ax.invert_yaxis()

#Panel 2
ax = axs[1]
ax.plot(arr_j_k, hmag, 'o', markersize=2, alpha=0.5, color='#FFC20A')
ax.set_title('2MASS J-K vs apparent H magnitude')
ax.set_ylabel('Apparent H magnitude (mag)')
ax.set_xlabel('J-K (mag)')
ax.invert_yaxis()

# Show plot
plt.tight_layout()
plt.savefig('cmds_R147.png', dpi=200, bbox_inches='tight')
plt.show()

```

And resulted to the graph below:



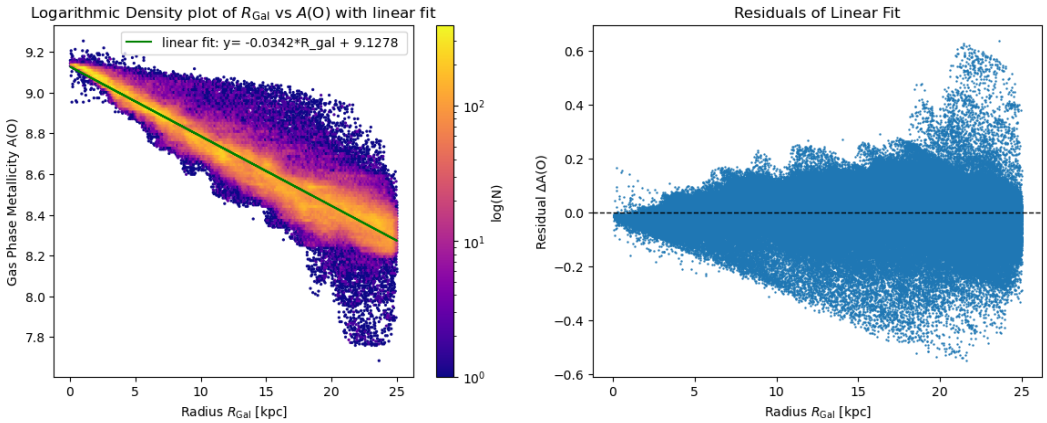
2.1.7: Give your colleague a recommendation for the potential proposal when only judging the fibre usage, that is, how many fibres of 2dF are available and would be used.

The 2dF fiber is designed to allow the acquisition of up to 392 simultaneous spectra of objects anywhere within a two degree field on the sky. After the quality cuts, we are left with 3535 stars, that would allow us to use ~9 of this 2dF fiber to its maximum potential.

QUESTION 3

3.1.1: Plot a 2-panel figure.

a) Logarithmic density plot of R_{Gal}. vs. A(O), with a linear fit and legend.
b) Residuals of the fit, R_{Gal}. vs. ΔA(O)



The code used:

```python
#KAEYRYNA ARYSSA BINTI KAEYDA

from astropy.table import Table
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
from scipy.stats import binned_statistic_2d
```

✓ 0.0s Python

```python
dat = Table.read('data/nihao_uhd_simulation_g8.26e11_xyz_positions_and_oxygen_ao.fits', format='fits')
df = dat.to_pandas()

print(df)

x_2 = df['x']**2
y_2 = df['y']**2
df['r_gal'] = np.sqrt(x_2+y_2)
```

✓ 0.1s Python

```python
# A) Logarithmic density plot R vs A(O)

def linear_function(x_data, y_data):
    linear_fit = np.polyfit(x_data, y_data, 1)
    m, b = linear_fit
    y_fit = (m * x_data) + b
    return m, b, y_fit

linear_fit = linear_function(df['r_gal'], df['A_O'])
m, b, y_fit = linear_fit

fig, axs = plt.subplots(1,2, figsize=(14, 5), sharex = True)

#logarithmic density plot
ax = axs[0]
hb = ax.hexbin(df['r_gal'], df['A_O'], gridsize=200, cmap='plasma', bins='log')
#Colourbar for density plot, N is data point
cb = plt.colorbar(hb)
cb.set_label('log(N)')

#plot linear fit
ax.plot(df['r_gal'], y_fit, 'g', label=f'linear fit: y= {m:.4f}*R_gal + {b:.4f} ')
ax.set_xlabel(r'Radius $R_{\mathrm{Gal}}$ [kpc]')
ax.set_ylabel('Gas Phase Metallicity A(O)')
ax.set_title(r'Logarithmic Density plot of $R_{\mathrm{Gal}}$ vs $A(\mathrm{O})$ with linear fit')
ax.legend()

# B) Residual of the fit

# Function for linear plotting
def y_predict(intercept, slope, x_data):
    y_model = intercept + slope * x_data
    return y_model

def delta_a (intercept, slope, x_data, y_data):
    y_model = y_predict(intercept, slope, x_data)
    diff_a = y_model - y_data
    return diff_a

#plot model line, for polyfit
df['diff_a'] = delta_a(b, m, df['r_gal'], df['A_O'])

ax = axs[1]
ax.scatter(df['r_gal'], df['diff_a'], label='delta_A', s=0.5)
ax.axhline(0, color='k', linestyle='--', lw=1)
ax.set_xlabel(r'Radius $R_{\mathrm{Gal}}$ [kpc]')
ax.set_ylabel('Residual ΔA(O)')
ax.set_title('Residuals of Linear Fit')
```

3.1.2: Use a python fitting tool to fit a linear function to the data, reporting the intercept and slope with uncertainties. Include any hyperparameters used.

Statsmodel was used to obtain the intercept and uncertainties of the linear fit. The result are as below:

3.2.2 Intercept with slope and uncertainties

```python
# Linear model
x_with_intercept = sm.add_constant(df['r_gal'])
linear_model = sm.OLS(df['A_O'], x_with_intercept).fit()

# Print linear model summary
print(linear_model.summary())

# Extract coefficients and uncertainties, and display
intercept, slope = linear_model.params
intercept_unc, slope_unc = linear_model.bse

print(f'Intercept = ({intercept:.4f} ± {intercept_unc:.4f})')
print(f'Slope = ({slope:.4f} ± {slope_unc:.4f})')
print(f'Linear model: y = ({intercept:.4f} ± {intercept_unc:.4f}) + ({slope:.4f} ± {slope_unc:.4f}) x')

df['y_pred'] = linear_model.predict(sm.add_constant(df['r_gal']))

#residual of y for statsmodel
def residual(y_data, y_pred):
    resid = y_pred - y_data
    return resid

df['resid_AO'] = residual(df['A_O'], df['y_pred'])
```

[5] ✓ 0.1s

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    A_O   R-squared:                       0.913
Model:                            OLS   Adj. R-squared:                  0.913
Method:                 Least Squares   F-statistic:                 5.337e+06
Date:                Sun, 05 Oct 2025   Prob (F-statistic):               0.00
Time:                        15:21:15   Log-Likelihood:             6.1547e+05
No. Observations:              511520   AIC:                        -1.231e+06
Df Residuals:                  511518   BIC:                        -1.231e+06
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1278      0.000   3.93e+04      0.000       9.127       9.128
r_gal         -0.0342   1.48e-05  -2310.203      0.000      -0.034      -0.034
==============================================================================
Omnibus:                    66623.269   Durbin-Watson:                   1.980
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           564064.307
Skew:                           0.349   Prob(JB):                         0.00
Kurtosis:                       8.097   Cond. No.                         36.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Intercept = (9.1278 ± 0.0002)
Slope = (-0.0342 ± 0.0000)
Linear model: y = (9.1278 ± 0.0002) + (-0.0342 ± 0.0000) x
```

3.1.3: Discuss where the linear model fits well and where it does not. Use statistical metrics, such as the root mean squares or other goodness-of-fit indicators, to quantify the performance of your linear fit in general and regions with larger residuals.

```
#root mean square error

#for region of large residual
def mean_square_error(y_data, y_pred):
    mse = np.sum(residual(y_data, y_pred)**2)/len(y_data)
    return mse

def root_mean_square_error(y_data, y_pred):
    rmse = np.sqrt(mean_square_error(y_data, y_pred))
    return rmse

print(f'Root mean square error:{root_mean_square_error( df['A_O'], df['y_pred']):.4f}')

#r-squared value from OLS, to quantify linear fitting performance
r_squared = linear_model.rsquared

print(f'R squared value: {r_squared:.4f} ')
```

✓ 0.0s

```
Root mean square error:0.0726
R squared value: 0.9125
```

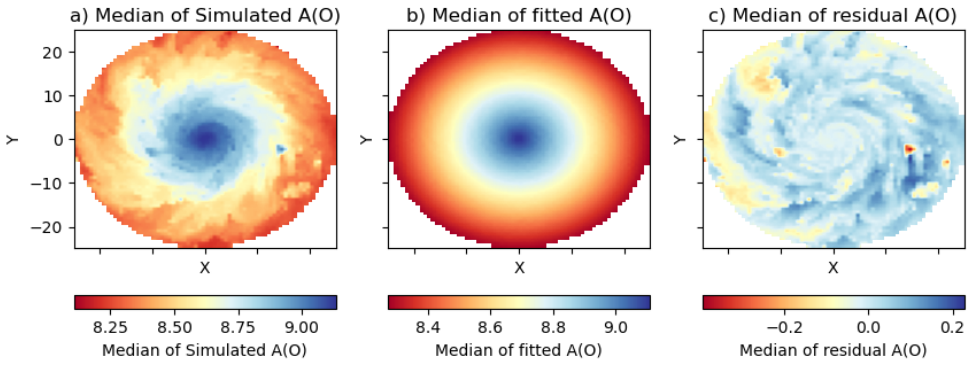
The linear model is shown to fit well with the data as the R^2 data has the value 0.9125, and is close to 1, for region with smaller R_{Gal}. This means that 91.25% variation of the A(O) data can be explained with the model. From the plotted logarithmic density plot in 3.1.1, we can see that the linear model fits the data at where the density is high, and hence, making the linear model fits well with the data in general. The region 20 kpc <= R_{Gal} <= 25kpc, it has larger residual, so this might contribute to the linear fit to have some variation that is not covered by the model.

Aside from the R^2, we also need to take into consideration of the root mean square error for the residual. For the residual, the root mean square error is quite small as it is ~0.07. The root mean square error describes that the predicted A(O) data has small difference from the actual data. We can see from the plot in 3.1.1(b), at a smaller R_{Gal} the residual is not large, but as we get to higher R_{Gal} , we can see the residual increases. Therefore, at a region with higher residual, the linear model is less suitable to describe the non-linear pattern in the plot, as the metallicity there might follow different mathematical description.

3.1.4: Plot a 3-panel figure for the x vs. y plane using the same bins and sensible colormaps

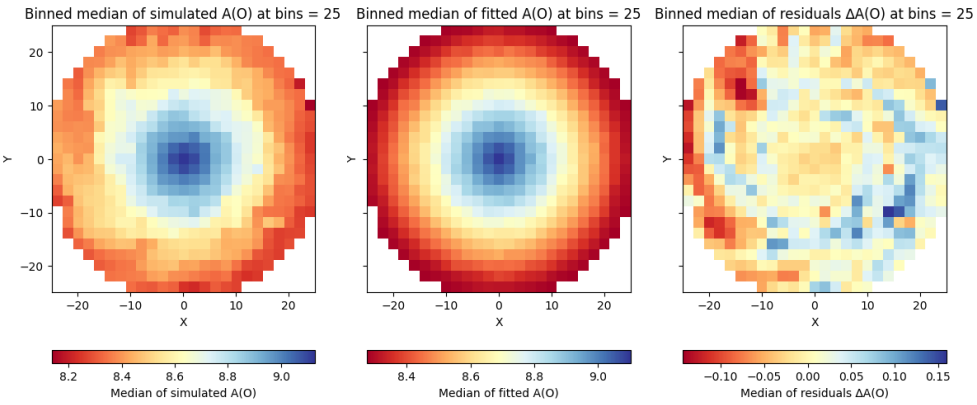
```python
# A) 2D Hist Median simulated


fig, axs = plt.subplots(3, 3, figsize=(10, 12), sharex=True, sharey=True)

bin_compare_1 = 30
bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['A_O'], statistic='median', bins=bin_compare_1)
im = axs[0,0].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[0,0].set_title("a) Median of Simulated A(O)")
axs[0,0].set_xlabel("X")
axs[0,0].set_ylabel("Y")
plt.colorbar(im, ax=axs[0,0], label="Median of Simulated A(O)", orientation='horizontal')

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['y_pred'], statistic='median', bins=bin_compare_1)
im = axs[0,1].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[0,1].set_title("b) Median of fitted A(O)")
axs[0,1].set_xlabel("X")
axs[0,1].set_ylabel("Y")
plt.colorbar(im, ax=axs[0,1], label="Median of fitted A(O)", orientation='horizontal')


bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['resid_AO'], statistic='median', bins=bin_compare_1)
im = axs[0,2].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[0,2].set_title("c) Median of residual A(O)")
axs[0,2].set_xlabel("X")
axs[0,2].set_ylabel("Y")
plt.colorbar(im, ax=axs[0,2], label="Median of residual A(O)", orientation='horizontal')



#define number of bins to be placed
bin_compare_2 = 70
bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['A_O'], statistic='median', bins=bin_compare_2)
im = axs[1,0].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[1,0].set_title("a) Median of Simulated A(O)")
axs[1,0].set_xlabel("X")
axs[1,0].set_ylabel("Y")
plt.colorbar(im, ax=axs[1,0], label="Median of Simulated A(O)", orientation='horizontal')

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['y_pred'], statistic='median', bins=bin_compare_2)
im = axs[1,1].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[1,1].set_title("b) Median of fitted A(O)")
axs[1,1].set_xlabel("X")
axs[1,1].set_ylabel("Y")
plt.colorbar(im, ax=axs[1,1], label="Median of fitted A(O)", orientation='horizontal')

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['resid_AO'], statistic='median', bins=bin_compare_2)
im = axs[1,2].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[1,2].set_title("c) Median of residual A(O)")
axs[1,2].set_xlabel("X")
axs[1,2].set_ylabel("Y")
plt.colorbar(im, ax=axs[1,2], label="Median of residual A(O)", orientation='horizontal')

#defince number of bins to be placed
bin_compare_3 = 100

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['A_O'], statistic='median', bins=bin_compare_3)
im = axs[2,0].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[2,0].set_title("a) Median of Simulated A(O)")
axs[2,0].set_xlabel("X")
axs[2,0].set_ylabel("Y")
plt.colorbar(im, ax=axs[2,0], label="Median of Simulated A(O)", orientation='horizontal')

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['y_pred'], statistic='median', bins=bin_compare_3)
im = axs[2,1].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[2,1].set_title("b) Median of fitted A(O)")
axs[2,1].set_xlabel("X")
axs[2,1].set_ylabel("Y")
plt.colorbar(im, ax=axs[2,1], label="Median of fitted A(O)", orientation='horizontal')

bin_stat_median, xedges, yedges, binnumber = binned_statistic_2d(df['x'], df['y'], df['resid_AO'], statistic='median', bins=bin_compare_3)
im = axs[2,2].imshow(bin_stat_median.T, origin='lower', aspect='auto',
                     extent=[xedges[0], xedges[-1], yedges[0], yedges[-1]],
                     cmap='RdYlBu')
axs[2,2].set_title("c) Median of residual A(O)")
axs[2,2].set_xlabel("X")
axs[2,2].set_ylabel("Y")
plt.colorbar(im, ax=axs[2,2], label="Median of residual A(O)", orientation='horizontal')
```

All figures were plotted with 70 bins. The above code showed for plotting of several bin size for testing purposes. Finally, plotting results with 70 bins were chosen.



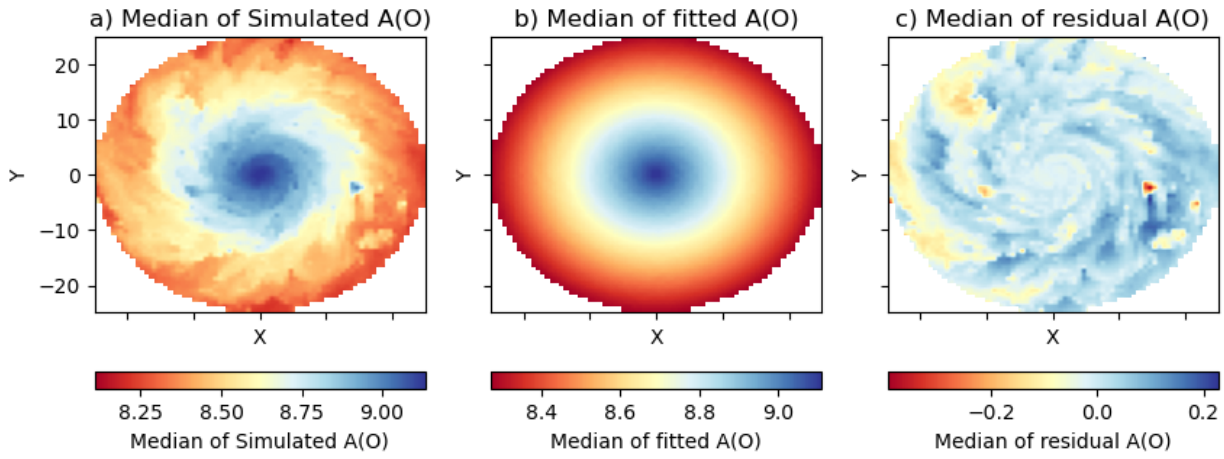
3.1.5: Describe your choice of 2D bins. Discuss what details would be missed with fewer bins or problems encountered with more bins.

Below is a plot with **smaller bin number** (25):

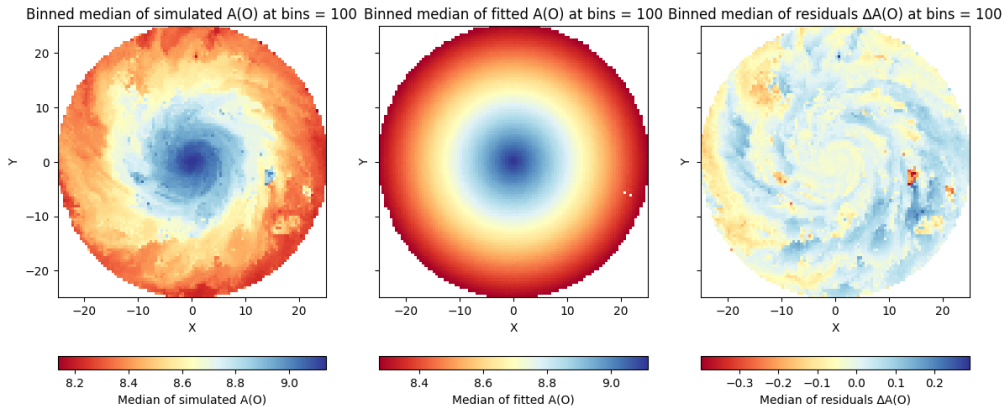


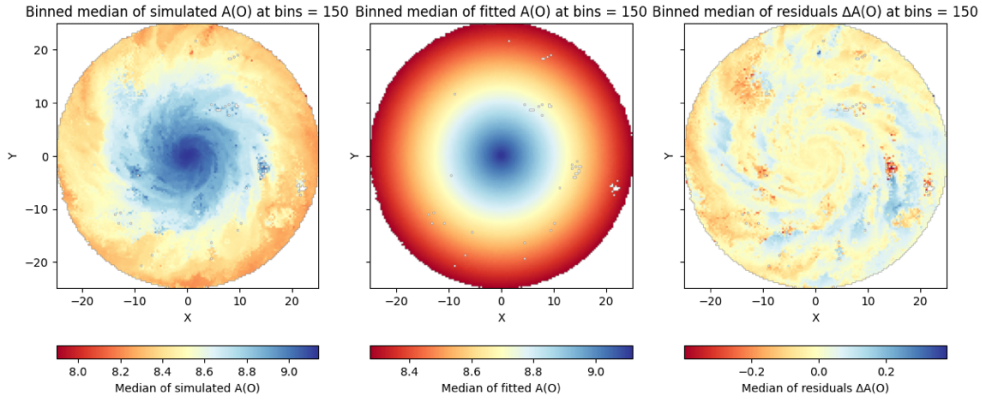
From the plot, we can see that the2D histogram does not give a good estimation on the data point cluster in a certain coordinate as the edge looks blurry, and making the structure of the data vague if we use small bins.

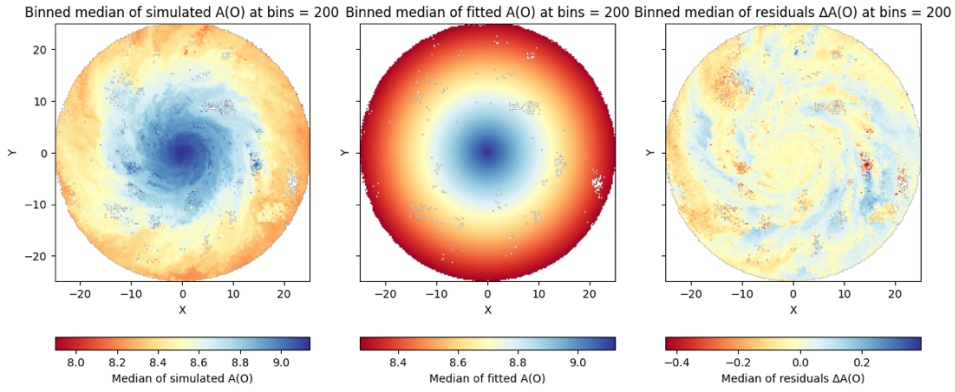
With increasing bin, the plot starts to look finer (have higher resolution). For the task, **my choice of bin is 70**, as we can see sharp edge and pattern for the plot, and it provides a good visualization on the data point clustering.



As we **increase the bin**, it was observed that the plot started to have empty spaces since we are slicing the bin very finely, so it is said that the plot does not visualize the data well for the amount of data that we have. A very small bin might fall between the data points that we have, resulting to the white pixels that we see as below:







3.1.6: Analyze the residuals in more detail and propose an explanation for any patterns you observe.

For the residual plot and the 2D histogram, it can be seen that at smaller R_{Gal} the data have small difference compared to the estimated A(O) that was calculated to make the linear fitting. This indicate that at small radius, the linear plot fits the data well.

As we go to higher R_{Gal} values, we can see that the data starts to have larger residuals, and this indicate that the linear model is unsuitable to be used to plot the data at these values. A reason for this might be that at the spiral arms of the milky way where stars are formed actively, the metallicity is higher might no longer follow the same slope that the linear model estimated. Therefore, we say at higher R_{Gal} the residual is large and the linear model does not fit the data well.