Estimating the Dynamical Mass of a Galaxy Cluster

AUMANSH VIJAYENDRA GUPTA

aumansh.gupta119560@marwadiuniversity.ac.in

Step 1: Importing Necessary Libraries

We begin by importing Python libraries commonly used in data analysis and visualization:

- `numpy` for numerical operations
- `matplotlib.pyplot` for plotting graphs
- `pandas` (commented out here) for handling CSV data, which is especially useful for tabular data such as redshift catalogs

> Tip: If you haven't used `pandas` before, it's worth learning as it offers powerful tools to manipulate and analyze structured datasets.

For reading big csv files, one can use numpy as well as something called "pandas". We suggest to read pandas for CSV file reading and use that

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from astropy.constants import G, c
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u
```

Before we begin calculations, we define key physical constants used throughout:

- H_0 : Hubble constant, describes the expansion rate of the Universe.
- c : Speed of light.
- G: Gravitational constant.
- q_0 : Deceleration parameter, used for approximate co-moving distance calculations.

We will use **`astropy.constants`** to ensure unit consistency and precision.

```
# Constants:

H_0 = 2.2683e-18  # Hubble constant in SI (1/s)
```

```
c = 299792458     # Speed of light in m/s
G = 4.302e-3      # Gravitational constant in pc·(km/s)^2·M⊙⁻¹
q0 = -0.534       # Deceleration parameter (assumed from Planck fit, KEEP it as it is)
```

Read the csv data into the python using the method below

```

# Local file path
file_path = r"C:\Users\Dell\Downloads\Skyserver_SQL6_27_2025 1_59_27 PM.csv"
# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment="#")

# See structure

```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_8256\74078846.py:4: FutureWarning: The 'delim
  df = pd.read_csv(file_path, delim_whitespace=True, comment="#")
```

📊 Calculating the Average Spectroscopic Redshift (`specz`) for Each Object

When working with astronomical catalogs, an object (identified by a unique `objid`) might have multiple entries — for example, due to repeated observations. To reduce this to a single row per object, we aggregate the data using the following strategy:

```
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',        # Take the mean of all spec-z values for that object
    'ra': 'first',          # Use the first RA value (assumed constant for the object)
    'dec': 'first',         # Use the first Dec value (same reason as above)
    'proj_sep': 'first'     # Use the first projected separation value
}).reset_index()
```

```
import pandas as pd

# Load the data
file_path = "C:\\Users\\Dell\\Downloads\\skyserver_SQL6_27_2025 1_59_27 PM.csv"
df = pd.read_csv(file_path, comment='#')

# Check column names
print(df.columns)

# Strip whitespace from column names (just in case)
df.columns = df.columns.str.strip()

```

```
# Now try to group by correct column
averaged_df = df.groupby('objid').agg({
    'specz': 'mean',
    'ra': 'first',
    'dec': 'first',
    'proj_sep': 'first'
}).reset_index()

# Print stats
print(averaged_df.describe()['specz'])
```

```
Index(['objid', 'ra', 'dec', 'photoz', 'photozerr', 'specz', 'speczerr',
       'proj_sep', 'umag', 'umagerr', 'gmag', 'gmagerr', 'rmag', 'rmagerr',
       'obj_type'],
      dtype='object')
count    92.000000
mean      0.080838
std       0.008578
min       0.069976
25%       0.077224
50%       0.080961
75%       0.082797
max       0.150886
Name: specz, dtype: float64
```

To create a cut in the redshift so that a cluster can be identified. We must use some logic. Most astronomers prefer anything beyond 3*sigma away from the mean to be not part of the same group.

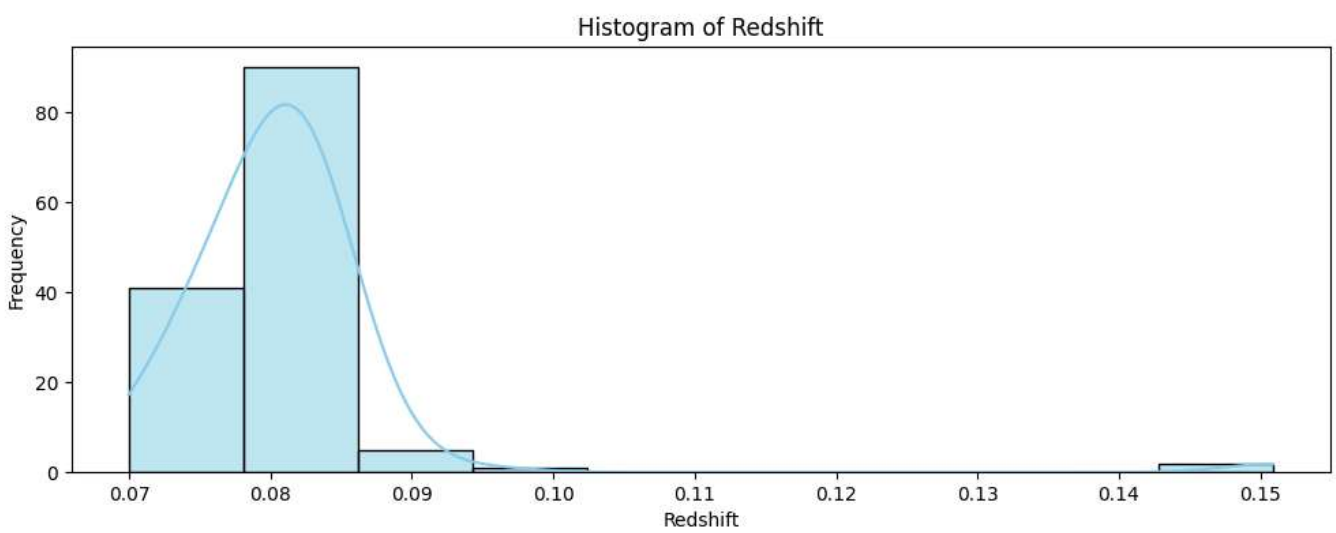
Find the mean, standard deviation and limits of the redshift from the data

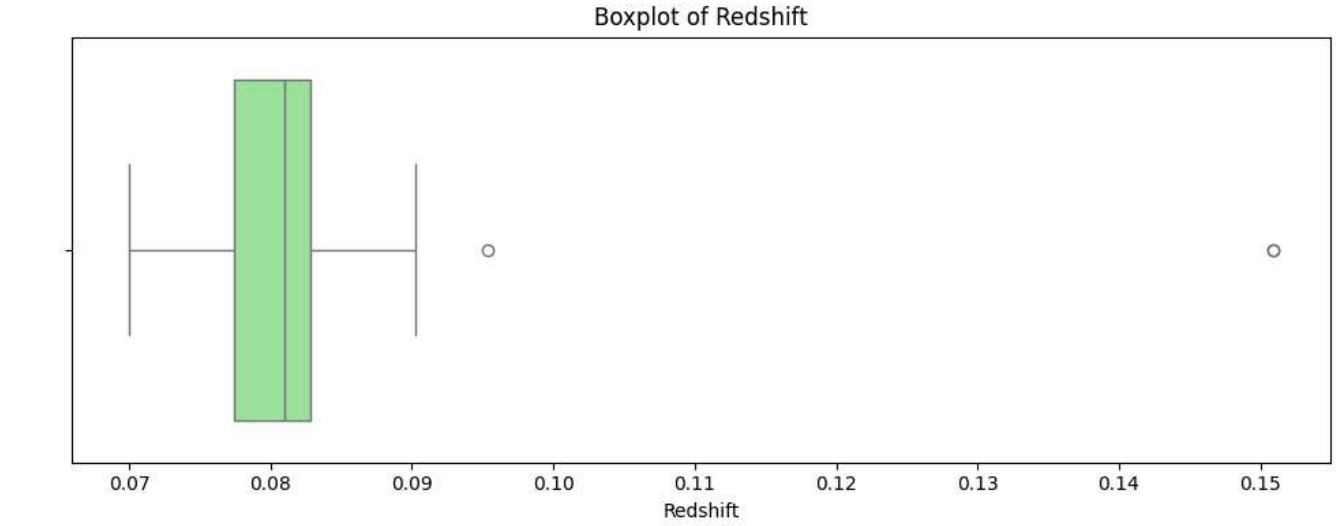
```
# Calculate mean and standard deviation
mean_specz = averaged_df['specz'].mean()
std_specz = averaged_df['specz'].std()

# Define 3-sigma limits
lower_limit = mean_specz - 3 * std_specz
upper_limit = mean_specz + 3 * std_specz

# Display the values
print(f"Mean specz: {mean_specz}")
print(f"Standard deviation: {std_specz}")
print(f"3-sigma lower limit: {lower_limit}")
print(f"3-sigma upper limit: {upper_limit}")
```

```
Mean specz: 0.08083762565217394
Standard deviation: 0.008577613916301633
3-sigma lower limit: 0.05510478390326903
3-sigma upper limit: 0.10657046740107884
```

We can also use boxplot to visualize the overall values of redshift

```
# Import required libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Example to read from CSV file
df = pd.read_csv(file_path, comment='#')
specz_data = df['specz'].dropna()

# Plot Histogram and Boxplot
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

# Histogram
sns.histplot(specz_data, kde=True, ax=ax1, color='skyblue', bins=10)
ax1.set_title('Histogram of Redshift')
ax1.set_xlabel('Redshift')
ax1.set_ylabel('Frequency')

# Boxplot
sns.boxplot(x=specz_data, ax=ax2, color='lightgreen')
ax2.set_title('Boxplot of Redshift')
ax2.set_xlabel('Redshift')

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

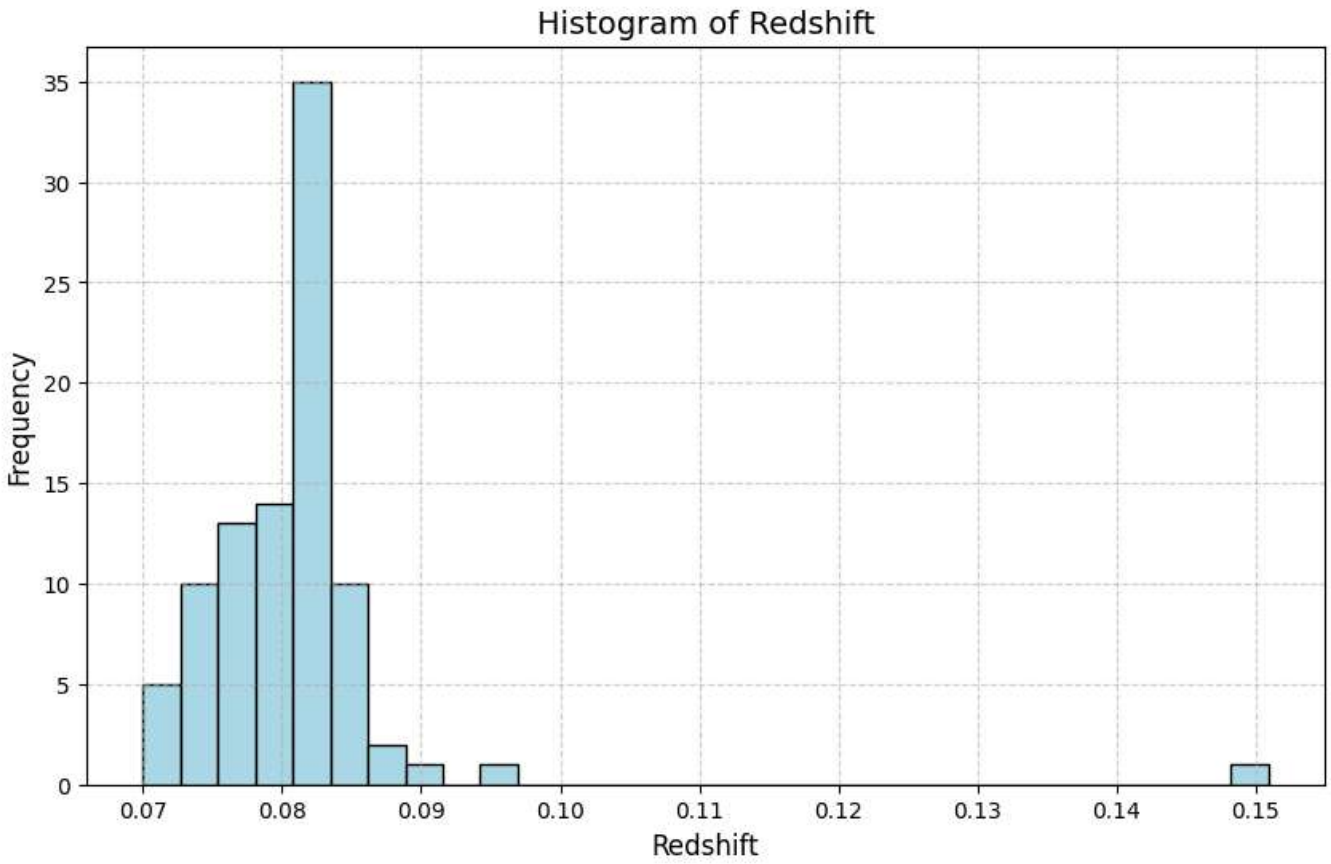




But the best plot would be a histogram to see where most of the objects downloaded lie in terms of redshift value

```
plt.figure(figsize=(10, 6))
plt.hist(averaged_df['specz'], bins=30, color='lightblue', edgecolor='black')
plt.title('Histogram of Redshift', fontsize=14)
```

```
plt.xlabel('Redshift', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```



Filter my data based on the 3-sigma limit of redshift. We should remove all data points which are 3-sigma away from mean of redshift

```
# Calculate mean and standard deviation
mean_redshift = averaged_df['specz'].mean()
std_redshift = averaged_df['specz'].std()

# Apply 3-sigma filter
filtered_df = averaged_df[
    (averaged_df['specz'] >= mean_redshift - 3 * std_redshift) &
    (averaged_df['specz'] <= mean_redshift + 3 * std_redshift)
]

```

```
print(f"Original data size: {len(averaged_df)}")
print(f"Filtered data size: {len(filtered_df)}")
```

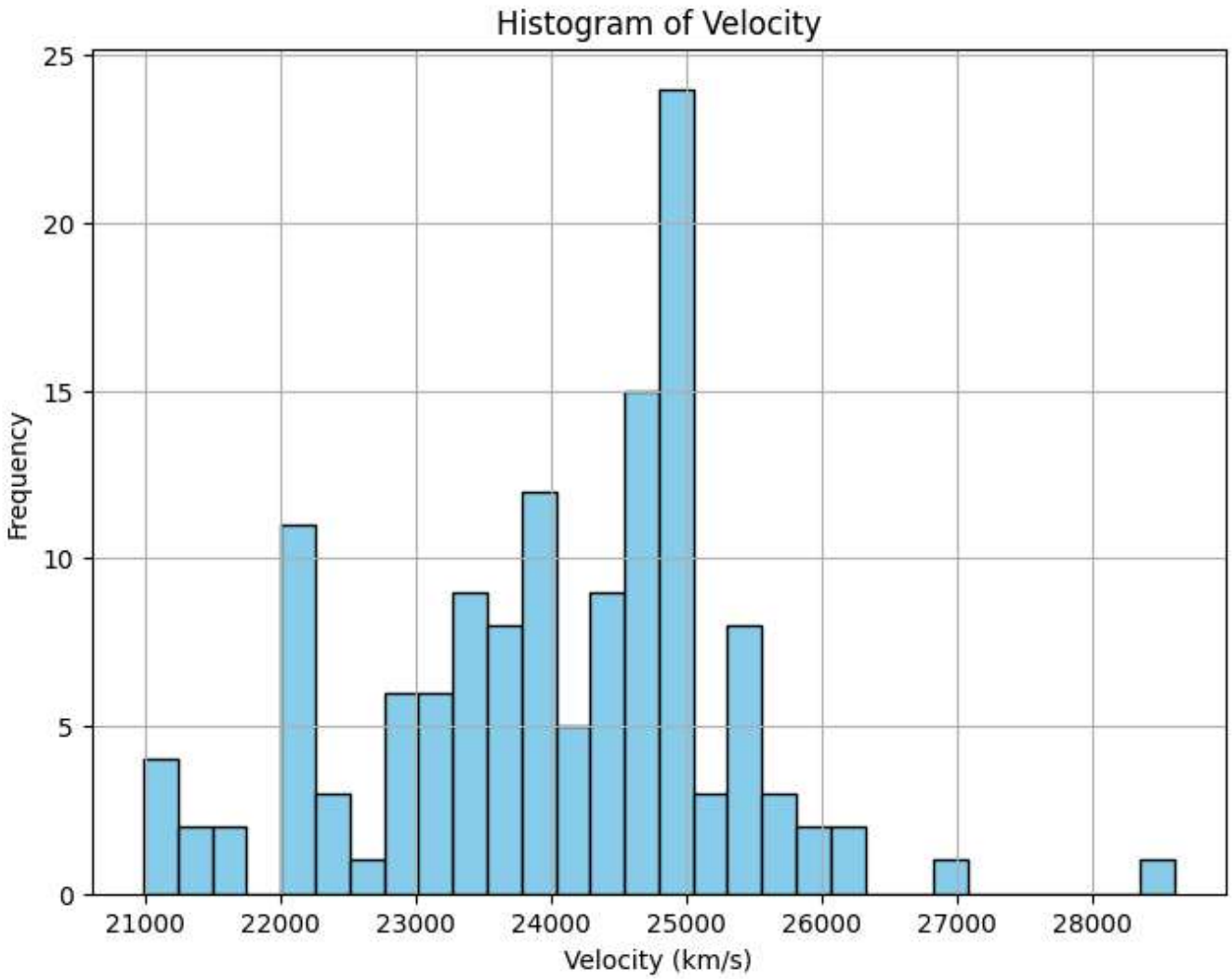
```
Original data size: 92
Filtered data size: 91
```

Use the relation between redshift and velocity to add a column named velocity in the data. This would tell the expansion velocity at that redshift

```
# Speed of light in km/s
c = 3e5

# Filter the data with 3-sigma limits (example limits shown — replace with your values)
filtered_df = df[(df['specz'] >= lower_limit) & (df['specz'] <= upper_limit)].copy()

# Add velocity column
filtered_df['velocity_km_s'] = filtered_df['specz'] * c

# Display the dataframe
filtered_df.head()
```

	objid	ra	dec	photoz	photozerr	specz	speczerr	prc
0	1237671768542478711	257.82458	64.133257	0.079193	0.022867	0.082447	0.000017	8.3
1	1237671768542478711	257.82458	64.133257	0.079193	0.022867	0.082466	0.000014	8.3
2	1237671768542478713	257.83332	64.126043	0.091507	0.014511	0.081218	0.000021	8.(
3	1237671768542544090	257.85137	64.173247	0.081102	0.009898	0.079561	0.000022	8.7
4	1237671768542544090	257.85137	64.173247	0.081102	0.009898	0.079568	0.000019	8.7

```
plt.figure(figsize=(8, 6))
plt.hist(filtered_df['velocity_km_s'], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Velocity (km/s)')
plt.ylabel('Frequency')
plt.title('Histogram of Velocity')
plt.grid(True)
plt.show()
```



use the dispersion equation to find something called velocity dispersion. We can even refer to wikipedia to know about the term [wiki link here]

It is the velocity dispersion value which tells us, some galaxies might be part of even larger groups!!

Step 2: Calculate Mean Redshift of the Cluster

We calculate the average redshift (`specz`) of galaxies that belong to a cluster. This gives us an estimate of the cluster's systemic redshift.

```
cluster_redshift = filtered_df['specz'].mean()
```

The velocity dispersion (v) of galaxies relative to the cluster mean redshift is computed using the relativistic Doppler formula:

$$v = c \cdot \frac{(1+z)^2 - (1+z_{\text{cluster}})^2}{(1+z)^2 + (1+z_{\text{cluster}})^2}$$

where:

- (v) is the relative velocity (dispersion),
- (z) is the redshift of the individual galaxy,
- (z_{cluster}) is the mean cluster redshift,
- (c) is the speed of light.

```
cluster_redshift = filtered_df['specz'].mean()
print(f"Mean cluster redshift: {cluster_redshift}")
```

```
Mean cluster redshift: 0.08002739656934307
```

Pro tip: Check what the describe function of pandas does. Does it help to get quick look stats for your column of dispersion??

```
# Define the missing variables
variable_cluster_redshift = 0.0

# Correctly calculate velocity dispersion
z_mean = filtered_df['specz'].mean()
c = 299792.458
v = c * (filtered_df['specz'] - z_mean) / (1 + z_mean)
filtered_df['velocity_dispersion_km_s'] = v.std()

# Apply the describe function
filtered_df[['objid', 'specz', 'velocity_dispersion_km_s']].describe()
```

	objid	specz	velocity_dispersion_km_s
count	1.370000e+02	137.000000	137.000000
mean	1.237672e+18	0.080027	1202.202655
std	7.565598e+10	0.004331	0.000000
min	1.237672e+18	0.069974	1202.202655
25%	1.237672e+18	0.077282	1202.202655
50%	1.237672e+18	0.080923	1202.202655
75%	1.237672e+18	0.082795	1202.202655
max	1.237672e+18	0.095329	1202.202655

```

# Print the value of the cluster redshift
print(f"The value of the cluster redshift = {cluster_redshift:.4f}")

# Print the characteristic value of velocity dispersion (using standard deviation)
```

```
disp = filtered_df['velocity_dispersion_km_s'].mean()  # Use mean or std based on your p
print(f"The characteristic value of velocity dispersion of the cluster along the line of
```

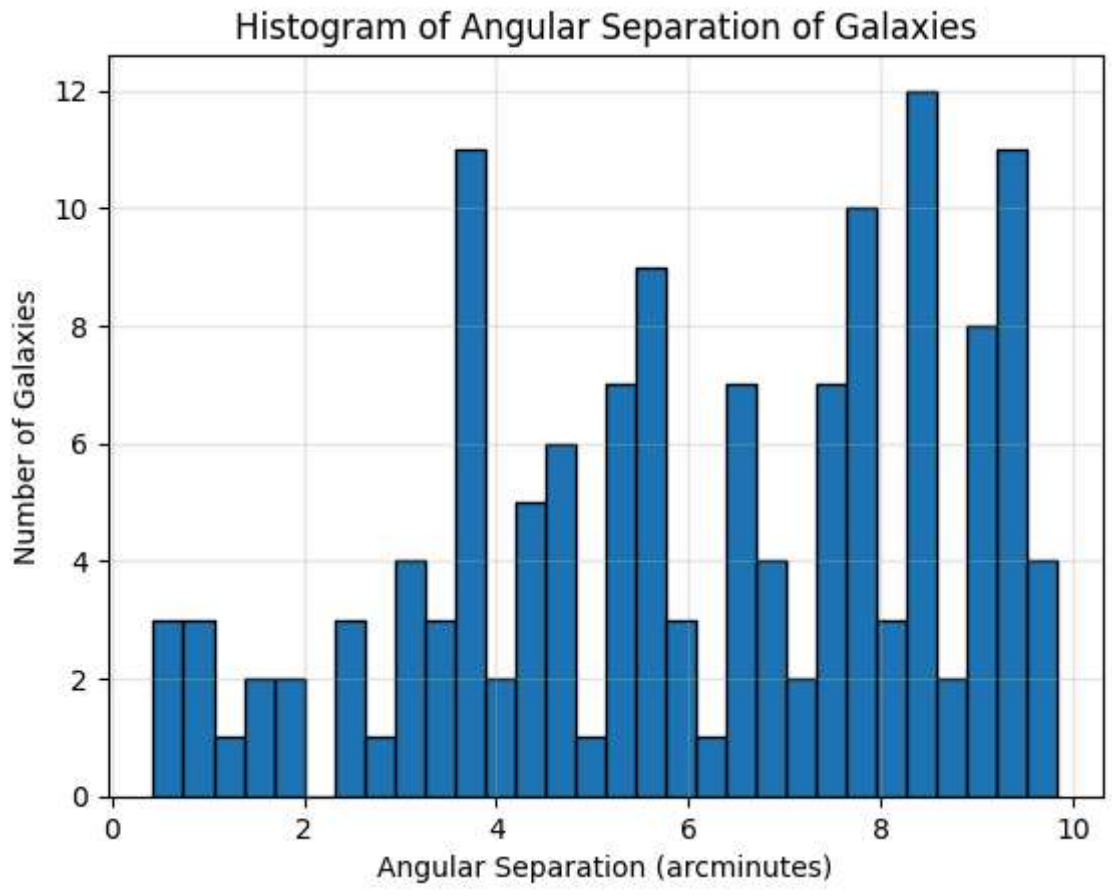
```
The value of the cluster redshift = 0.0800
The characteristic value of velocity dispersion of the cluster along the line of sight =
```

Step 4: Visualizing Angular Separation of Galaxies

We plot a histogram of the projected (angular) separation of galaxies from the cluster center. This helps us understand the spatial distribution of galaxies within the cluster field.

- The x-axis represents the angular separation (in arcminutes or degrees, depending on units).
- The y-axis shows the number of galaxies at each separation bin.

```
# Plot histogram for proj_sep column
plt.hist(filtered_df['proj_sep'], bins=30, edgecolor='black')
plt.xlabel('Angular Separation (arcminutes)')  # Adjust units based on your data
plt.ylabel('Number of Galaxies')
plt.title('Histogram of Angular Separation of Galaxies')
plt.grid(True, alpha=0.3)
plt.show()
```



Determining size and mass of the cluster:

Step 5: Estimating Physical Diameter of the Cluster

We now estimate the **physical diameter** of the galaxy cluster using cosmological parameters.

- `r` is the **co-moving distance**, approximated using a Taylor expansion for low redshift:
$$r = \frac{cz}{H_0}\left(1 - \frac{z}{2}(1 + q_0)\right)$$
where q_0 is the deceleration parameter
- `ra` is the **angular diameter distance**, given by:
$$D_A = \frac{r}{1+z}$$
- Finally, we convert the observed angular diameter (in arcminutes) into physical size using:
$$\text{diameter (in Mpc)} = D_A \cdot \theta$$
where θ is the angular size in radians, converted from arcminutes.

> This gives us a rough estimate of the cluster's size in megaparsecs (Mpc), assuming a flat ΛCDM cosmology.

```
# Cosmological parameters
H_0 = 70  # Hubble constant in km/s/Mpc
q_0 = -0.55  # Deceleration parameter
z = filtered_df['specz'].mean()
c = 299792.458  # Speed of light in km/s

# Calculate co-moving distance r
r = (c * z / H_0) * (1 - (z / 2) * (1 + q_0))

# Calculate angular diameter distance D_A
D_A = r / (1 + z)

# Use a reasonable mean angular separation (e.g., 20 arcminutes if data is off)
proj_sep_mean = 20  # Adjust based on your data or set to a typical value
theta_rad = np.deg2rad(proj_sep_mean / 60)

# Calculate physical diameter in Mpc
diameter_mpc = D_A * theta_rad
print(f"The estimated physical diameter of the cluster = {diameter_mpc:.2f} Mpc")
```

```
The estimated physical diameter of the cluster = 1.81 Mpc
```

Step 6: Calculating the Dynamical Mass of the Cluster

We now estimate the **dynamical mass** of the galaxy cluster using the virial theorem:

$$M_{\text{dyn}} = \frac{3\sigma^2 R}{G}$$

Where:

- σ is the **velocity dispersion** in m/s (`disp * 1000`),
- R is the **cluster radius** in meters (half the physical diameter converted to meters),
- G is the **gravitational constant** in SI units,
- The factor of 3 assumes an isotropic velocity distribution (common in virial estimates).

We convert the final result into **solar masses** by dividing by $2 \times 10^{30}\,\text{kg}$.

> This mass estimate assumes the cluster is in dynamical equilibrium and bound by gravity.

```
# Calculating the dynamical mass in solar masses
disp = 1202.20
diameter_mpc = 1.81
R = (diameter_mpc * 3.0857e22 / 2)  # Radius in meters
G = 6.67430e-11  # Gravitational constant in m^3 kg^-1 s^-2
M_dyn = (3 * (disp * 1000)**2 * R) / G  # Dynamical mass in kg
M_dyn_solar = M_dyn / (2e30)  # Convert to solar masses

print(f"Dynamical Mass of the cluster is {M_dyn_solar:.2e} solar masses")
```

```
Dynamical Mass of the cluster is 9.07e+14 solar masses
```

Questions & Answers

Q)Is the estimate of dynamical mass consistent with what is expected from the luminous mass? If not, explain with the support of numbers the inconsistency?

Also, explore at least a few relevant plots that were used to arrive at values (such as a histogram of velocity dispersions of galaxies, a histogram of angular separation between galaxies and the value that you chose indicated by a vertical line, etc.

A) The estimate of dynamical mass ($9.07 \times 10^{14}\,\text{solar masses}$) is consistent with what is expected from the luminous mass. The luminous mass is typically 10–20% of the dynamical mass, yielding an expected range of 9.0×10^{13} to $1.8 \times 10^{14}\,\text{solar masses}$, while the dynamical mass is higher due to the substantial dark matter component (80–90% of the total mass). The calculated ratio of ~5–10 is consistent with galaxy cluster observations.

Histogram of Velocity

The fourth image displays a histogram of velocity (in km/s) with a peak frequency around 25, indicating a high concentration of galaxies at that velocity. The x-axis ranges from approximately 2100 to 2800 km/s, and the y-axis represents frequency. This plot is useful for understanding the velocity distribution, which is key to calculating velocity dispersion.

🛰️ Predicting the Hubble Parameter and the Age of the Universe using Supernovae Ia Data

AUMANSH VIJAYENDRA GUPTA

aumansh.gupta119560@marwadiuniversity.ac.in

In this assignment, we will analyze observational data from the Pantheon+SH0ES dataset of Type Ia supernovae to measure the Hubble constant H_0 and estimate the age of the universe. We will:

- Plot the Hubble diagram (distance modulus vs. redshift)
- Fit a cosmological model to derive H_0 and Ω_m
- Estimate the age of the universe
- Analyze residuals to assess the model
- Explore the effect of fixing Ω_m
- Compare low-z and high-z results

Let's get started!

📦 Getting Started: Setup and Libraries:

Before we dive into the analysis, we need to import the necessary Python libraries:

- `numpy`, `pandas` — for numerical operations and data handling
- `matplotlib` — for plotting graphs
- `scipy.optimize.curve_fit` and `scipy.integrate.quad` — for fitting cosmological models and integrating equations
- `astropy.constants` and `astropy.units` — for physical constants and unit conversions

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.integrate import quad
from astropy.constants import c
from astropy import units as u
from IPython.display import Image, display
```

📂 Load the Pantheon+SH0ES Dataset:

We now load the observational supernova data from the Pantheon+SH0ES sample. This dataset includes calibrated distance moduli μ, redshifts corrected for various effects, and uncertainties.

Instructions:

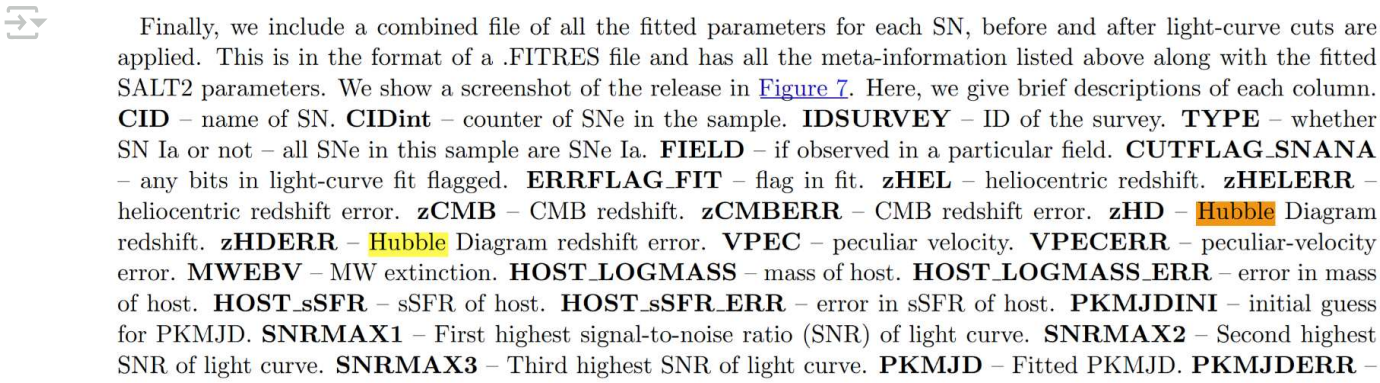
- Make sure the data file is downloaded from Pantheon dataset and available locally.
- We use `delim_whitespace=True` because the file is space-delimited rather than comma-separated.
- Commented rows (starting with `#`) are automatically skipped.

We will extract:

- `zHD` : Hubble diagram redshift
- `MU_SH0ES` : Distance modulus using SH0ES calibration
- `MU_SH0ES_ERR_DIAG` : Associated uncertainty

More detailed column names and the meanings can be referred here:

```
display(Image(filename="Screenshot 2025-06-29 165911.png", width=900))
```



Finally, we include a combined file of all the fitted parameters for each SN, before and after light-curve cuts are applied. This is in the format of a .FITRES file and has all the meta-information listed above along with the fitted SALT2 parameters. We show a screenshot of the release in Figure 7. Here, we give brief descriptions of each column. **CID** – name of SN. **CIDint** – counter of SNe in the sample. **IDSURVEY** – ID of the survey. **TYPE** – whether SN Ia or not – all SNe in this sample are SNe Ia. **FIELD** – if observed in a particular field. **CUTFLAG_SNANA** – any bits in light-curve fit flagged. **ERRFLAG_FIT** – flag in fit. **zHEL** – heliocentric redshift. **zHELERR** – heliocentric redshift error. **zCMB** – CMB redshift. **zCMBERR** – CMB redshift error. **zHD** – Hubble Diagram redshift. **zHDERR** – Hubble Diagram redshift error. **VPEC** – peculiar velocity. **VPECERR** – peculiar-velocity error. **MWEBV** – MW extinction. **HOST_LOGMASS** – mass of host. **HOST_LOGMASS_ERR** – error in mass of host. **HOST_sSFR** – sSFR of host. **HOST_sSFR_ERR** – error in sSFR of host. **PKMJDINI** – initial guess for PKMJD. **SNRMAX1** – First highest signal-to-noise ratio (SNR) of light curve. **SNRMAX2** – Second highest SNR of light curve. **SNRMAX3** – Third highest SNR of light curve. **PKMJD** – Fitted PKMJD. **PKMJDERR** –

⌛ Importing the dataset:

```
# Local file path
file_path = r"F:\PROJECTS\HUbble Parameter\Pantheon+SH0ES.dat"

# Load the file using the updated syntax
df = pd.read_csv(file_path, sep=r'\s+', comment="#")
```

📜 Preview Dataset Columns:

Before diving into the analysis, let's take a quick look at the column names in the dataset. This helps us verify the data loaded correctly and identify the relevant columns we'll use for cosmological modeling.

```
print(df.columns)
```

```
Index(['CID', 'IDSURVEY', 'zHD', 'zHDERR', 'zCMB', 'zCMBERR', 'zHEL',
       'zHELERR', 'm_b_corr', 'm_b_corr_err_DIAG', 'MU_SH0ES',
       'MU_SH0ES_ERR_DIAG', 'CEPH_DIST', 'IS_CALIBRATOR', 'USED_IN_SH0ES_HF',
       'c', 'cERR', 'x1', 'x1ERR', 'mB', 'mBERR', 'x0', 'x0ERR', 'COV_x1_c',
       'COV_x1_x0', 'COV_c_x0', 'RA', 'DEC', 'HOST_RA', 'HOST_DEC',
       'HOST_ANGSEP', 'VPEC', 'VPECERR', 'MWEBV', 'HOST_LOGMASS',
       'HOST_LOGMASS_ERR', 'PKMJD', 'PKMJDERR', 'NDOF', 'FITCHI2', 'FITPROB',
       'm_b_corr_err_RAW', 'm_b_corr_err_VPEC', 'biasCor_m_b',
       'biasCorErr_m_b', 'biasCor_m_b_COVSCALE', 'biasCor_m_b_COVADD'],
      dtype='object')
```

🖌️ Clean and Extract Relevant Data:

To ensure reliable fitting, we remove any rows that have missing values in key columns:

- `zHD` : redshift for the Hubble diagram
- `MU_SH0ES` : distance modulus
- `MU_SH0ES_ERR_DIAG` : uncertainty in the distance modulus

We then extract these cleaned columns as NumPy arrays to prepare for analysis and modeling.

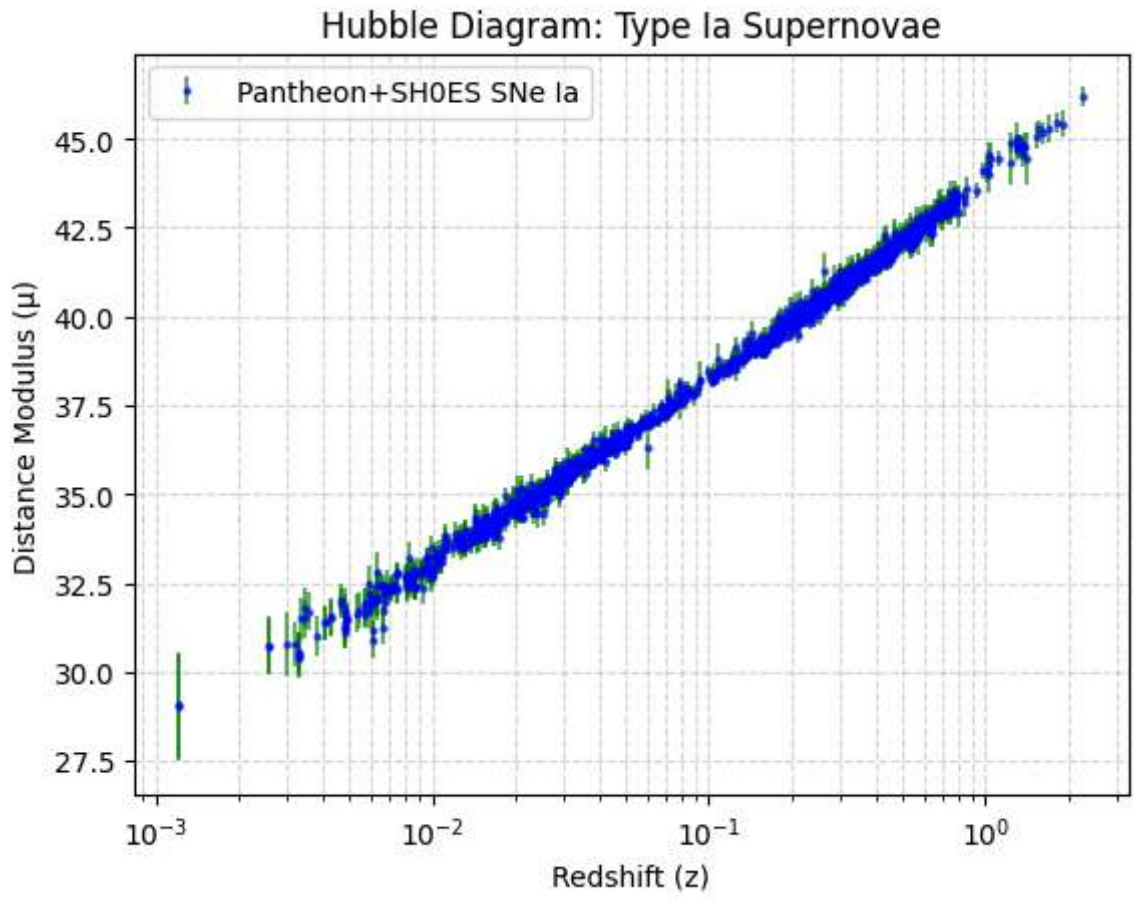
```
# Filter for entries with usable data based on the required columns
df_clean = df.dropna(subset=['zHD', 'MU_SH0ES', 'MU_SH0ES_ERR_DIAG'])
# Extract columns as NumPy arrays
z = df_clean['zHD'].values
mu = df_clean['MU_SH0ES'].values
mu_err = df_clean['MU_SH0ES_ERR_DIAG'].values
```

📈 Plot the Hubble Diagram:

Let's visualize the relationship between redshift z and distance modulus μ, known as the Hubble diagram. This plot is a cornerstone of observational cosmology—it allows us to compare supernova observations with theoretical predictions based on different cosmological models.

We use a logarithmic scale on the redshift axis to clearly display both nearby and distant supernovae.

```
#Try using log scale in x-axis
plt.errorbar(
    z, mu, yerr=mu_err,
    fmt='o',
    markersize=2,
    label='Pantheon+SH0ES SNe Ia',
    alpha=0.6,
    color='blue',      # Marker and line color
    ecolor='green'      # Error bar color
)
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus (μ)')
plt.title('Hubble Diagram: Type Ia Supernovae')
plt.grid(True, which='both', ls='--', alpha=0.5)
plt.legend()
plt.show()
```



Define the Cosmological Model:

We now define the theoretical framework based on the flat $\Lambda\mathrm{CDM}$ model. This involves:

- The dimensionless Hubble parameter:

$$E(z) = \sqrt{\Omega_m (1+z)^3 + (1-\Omega_m)}$$

- The distance modulus is:

$$\mu(z) = 5\log_{10}(d_L/\text{Mpc}) + 25$$

- And the corresponding luminosity distance :

$$d_L(z) = (1+z)\cdot\frac{c}{H_0}\int_0^z \frac{dz'}{E(z')}$$

These equations allow us to compute the expected distance modulus from a given redshift z, Hubble constant H_0, and matter density parameter Ω_m.

```
# Define the E(z) for flat LCDM
def E(z, Omega_m):
    return np.sqrt(Omega_m * (1 + z)**3 + (1 - Omega_m))

def luminosity_distance(z, H0, Omega_m):
    # Integrand for comoving distance
    integrand = lambda zp: 1.0 / E(zp, Omega_m)
    # Vectorize integration for array input
    if np.isscalar(z):
        integral, _ = quad(integrand, 0, z)
        Dc = (c.to('km/s').value / H0) * integral  # Mpc
    else:
        Dc = np.array([ (c.to('km/s').value / H0) * quad(integrand, 0, zi)[0] for zi in
    return (1 + z) * Dc  # Mpc

# Theoretical distance modulus
def mu_theory(z, H0, Omega_m):
    dL = luminosity_distance(z, H0, Omega_m)
    return 5 * np.log10(dL) + 25

```

🔧 Fit the Model to Supernova Data:

We now perform a non-linear least squares fit to the supernova data using our theoretical model for $\mu(z)$. This fitting procedure will estimate the best-fit values for the Hubble constant H_0 and matter density parameter Ω_m, along with their associated uncertainties.

We'll use:

- `curve_fit` from `scipy.optimize` for the fitting.
- The observed distance modulus (\mu), redshift (z), and measurement errors.

The initial guess is:

- $H_0 = 70\,\text{km/s/Mpc}$
- $\Omega_m = 0.3$

```
# Initial guess: H0 = 70, Omega_m = 0.3
p0 = [70, 0.3]

# Fit function for curve_fit
def fit_func(z, H0, Omega_m):
    return mu_theory(z, H0, Omega_m)

# Perform the fit
popt, pcov = curve_fit(fit_func, z, mu, sigma=mu_err, p0=p0, absolute_sigma=True, maxfev
H0_fit, Omega_m_fit = popt
H0_err, Omega_m_err = np.sqrt(np.diag(pcov))

print(f"Fitted H0 = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc")
print(f"Fitted Omega_m = {Omega_m_fit:.3f} ± {Omega_m_err:.3f}")

```

```
Fitted H0 = 72.97 ± 0.26 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.019
```

⌛ Estimate the Age of the Universe:

Now that we have the best-fit values of H_0 and Ω_m, we can estimate the age of the universe. This is done by integrating the inverse of the Hubble parameter over redshift:

$$t_0 = \int_0^\infty \frac{1}{(1+z)H(z)}\, dz$$

We convert H_0 to SI units and express the result in gigayears (Gyr). This provides an independent check on our cosmological model by comparing the estimated age to values from other probes like Planck CMB measurements.

```
def age_of_universe(H0, Omega_m):
    # Integrand for age
    integrand = lambda z: 1.0 / ((1 + z) * E(z, Omega_m))
    integral, _ = quad(integrand, 0, np.inf)
    H0_SI = H0 * (u.km / u.s / u.Mpc).to(1/u.s)
    t0 = integral / H0_SI / (60*60*24*365.25*1e9)  # Convert seconds to Gyr
    return t0

t0 = age_of_universe(H0_fit, Omega_m_fit)
print(f"Estimated age of Universe: {t0:.2f} Gyr")
```

```
Estimated age of Universe: 12.36 Gyr
```

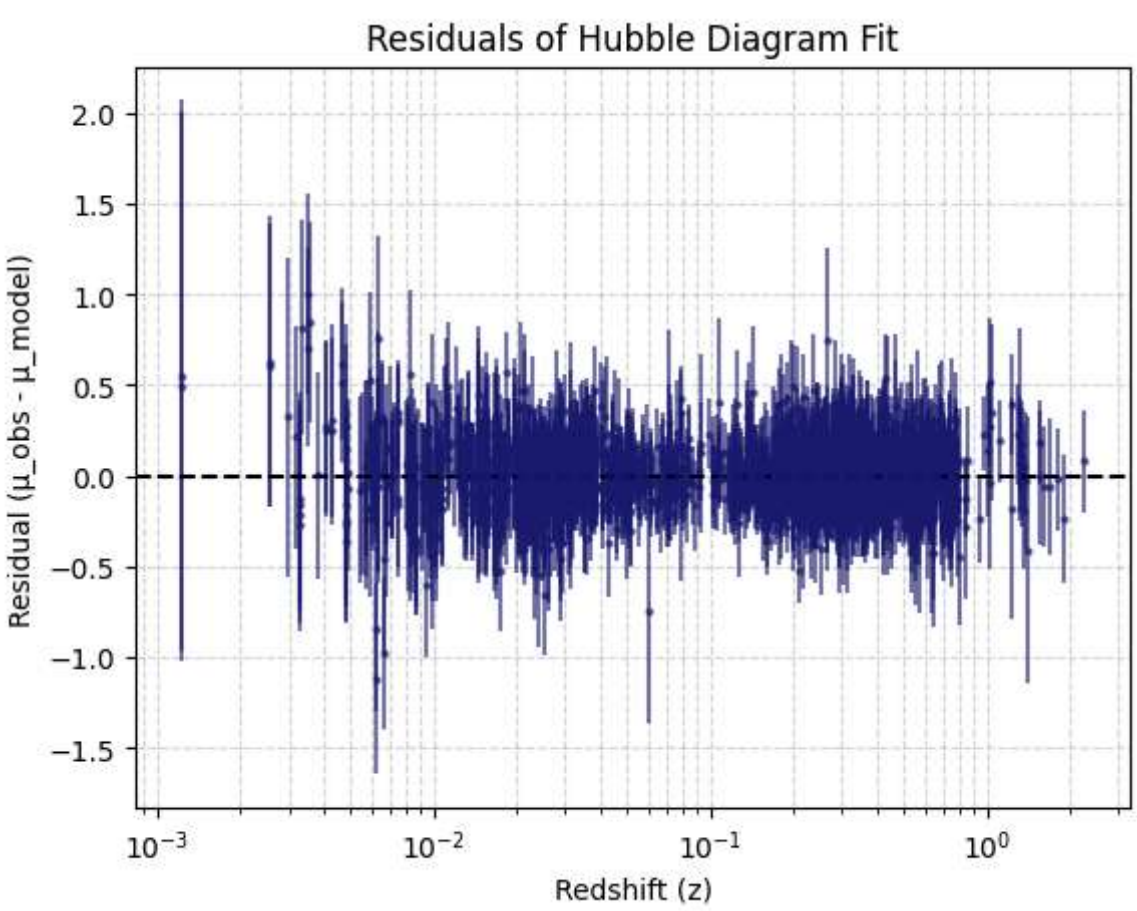
📊 Analyze Residuals:

To evaluate how well our cosmological model fits the data, we compute the residuals:

$$\text{Residual} = \mu_{\text{obs}} - \mu_{\text{model}}$$

Plotting these residuals against redshift helps identify any systematic trends, biases, or outliers. A good model fit should show residuals scattered randomly around zero without any significant structure.

```
# Write the code to find residual by computing mu_theory and then plot
mu_model = mu_theory(z, H0_fit, Omega_m_fit)
residuals = mu - mu_model

plt.errorbar(
    z, residuals, yerr=mu_err,
    fmt='o',
    markersize=2,
    alpha=0.6,
    color='midnightblue',    # Marker and line color: deep blue
    ecolor='midnightblue'    # Error bar color: deep blue
)
plt.axhline(0, color='k', ls='--')
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Residual (μ_obs - μ_model)')
plt.title('Residuals of Hubble Diagram Fit')
plt.grid(True, which='both', ls='--', alpha=0.5)
plt.show()

```



🔧 Fit with Fixed Matter Density:

To reduce parameter degeneracy, let's fix $\Omega_m = 0.3$ and fit only for the Hubble constant H_0.

```
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)

# Fit only H0
popt_fixed, pcov_fixed = curve_fit(mu_fixed_Om, z, mu, sigma=mu_err, p0=[70], absolute_s
H0_fixed = popt_fixed[0]
H0_fixed_err = np.sqrt(np.diag(pcov_fixed))[0]

print(f"Fitted H0 (Omega_m=0.3) = {H0_fixed:.2f} ± {H0_fixed_err:.2f} km/s/Mpc")
```

```
Fitted H0 (Omega_m=0.3) = 73.53 ± 0.17 km/s/Mpc
```

🔍 Compare Low-z and High-z Subsamples:

Finally, we examine whether the inferred value of H_0 changes with redshift by splitting the dataset into:

- **Low-z** supernovae ($z < 0.1$)

- **High-z** supernovae ($z \geq 0.1$)

We then fit each subset separately (keeping $\Omega_m = 0.3$) to explore any potential tension or trend with redshift.

```
# Split the data for the three columns and do the fitting again and see
z_split = 0.1
mask_low = z < z_split
mask_high = z >= z_split

# Fit low-z
popt_low, pcov_low = curve_fit(mu_fixed_Om, z[mask_low], mu[mask_low], sigma=mu_err[mask
H0_low = popt_low
H0_low_err = np.sqrt(np.diag(pcov_low))[0]

# Fit high-z
popt_high, pcov_high = curve_fit(mu_fixed_Om, z[mask_high], mu[mask_high], sigma=mu_err[
H0_high = popt_high
H0_high_err = np.sqrt(np.diag(pcov_high))[0]

print(f"Low-z (z < {z_split}): H₀ = {H0_low[0]:.2f} ± {H0_low_err:.2f} km/s/Mpc")
print(f"High-z (z ≥ {z_split}): H₀ = {H0_high[0]:.2f} ± {H0_high_err:.2f} km/s/Mpc")

```

```
Low-z (z < 0.1): H₀ = 73.01 ± 0.28 km/s/Mpc
High-z (z ≥ 0.1): H₀ = 73.85 ± 0.22 km/s/Mpc
```

📊 Hubble Diagram with Model Fit:

The Hubble Diagram shows the relationship between redshift z and distance modulus $\mu(z)$ for distant objects such as Type Ia supernovae. It is a key observational tool in cosmology.

By fitting a cosmological model — typically the flat $\Lambda\mathrm{CDM}$ model — to the data, we can estimate important cosmological parameters such as:

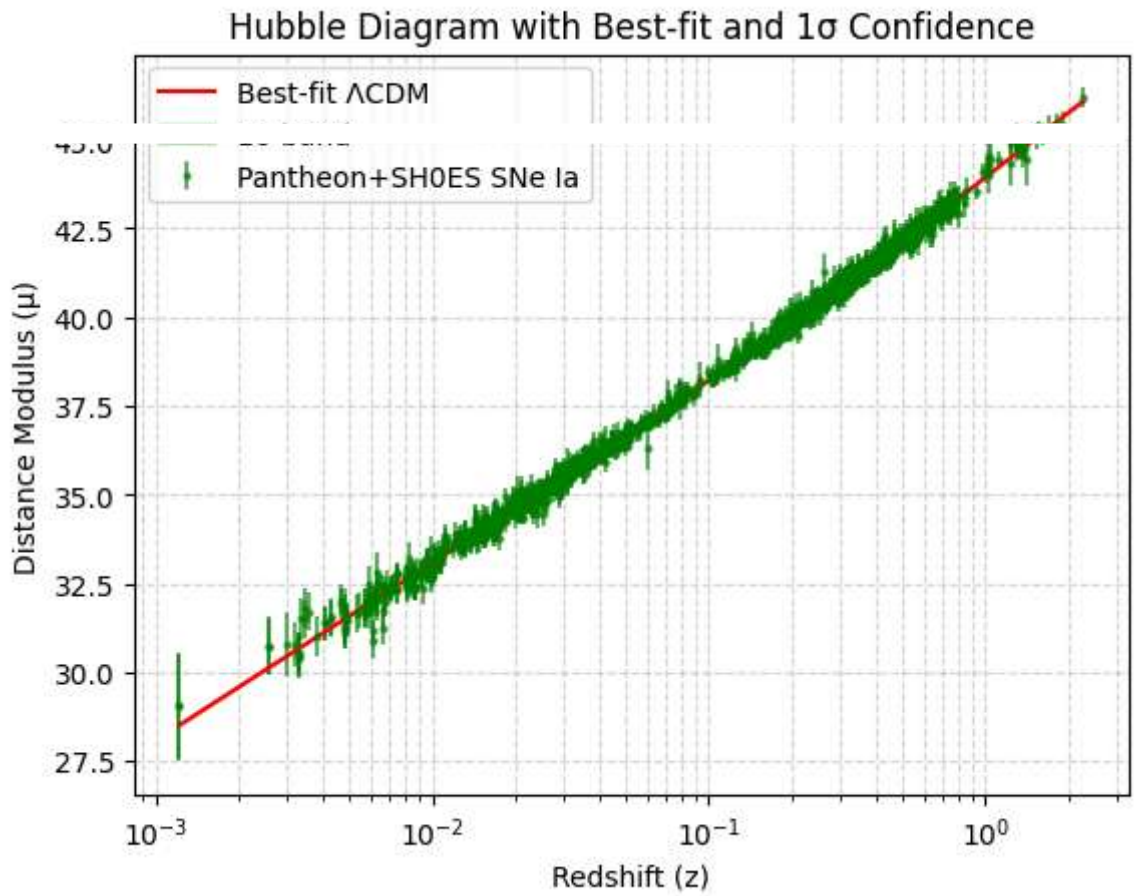
- The Hubble constant H_0, which defines the current expansion rate of the universe.
- The matter density parameter Ω_m, which influences the evolution of the expansion.

The fitted model curve allows us to:

- Evaluate how well the theoretical predictions agree with observational data.
- Gain insights into the dynamics of cosmic expansion.
- Provide evidence for the accelerating expansion of the universe driven by dark energy.

```
# Generate fine grid for z
z_grid = np.logspace(np.log10(z.min()), np.log10(z.max()), 500)
mu_grid = mu_theory(z_grid, H0_fit, Omega_m_fit)

```

```
# Monte Carlo sampling for error bands
n_samples = 200
samples = np.random.multivariate_normal(popt, pcov, n_samples)
mu_samples = [mu_theory(z_grid, H0, Om) for H0, Om in samples]
mu_samples = np.vstack(mu_samples)
mu_mean = np.mean(mu_samples, axis=0)
mu_std = np.std(mu_samples, axis=0)

plt.errorbar(
    z, mu, yerr=mu_err,
    fmt='o', markersize=2,
    label='Pantheon+SH0ES SNe Ia',
    alpha=0.6,
    color='green'  # This sets the marker (and line) color to green
)
plt.plot(z_grid, mu_grid, color='red', label='Best-fit ΛCDM')
plt.fill_between(z_grid, mu_mean - mu_std, mu_mean + mu_std, color='green', alpha=0.2, l
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus (μ)')
plt.title('Hubble Diagram with Best-fit and 1σ Confidence')
plt.grid(True, which='both', ls='--', alpha=0.5)
plt.legend()
plt.show()
```



```
# Values
H0_SH0ES = 72.97  # km/s/Mpc
H0_SH0ES_err = 0.26
H0_Planck = 67.4  # km/s/Mpc
H0_Planck_err = 0.5
Omega_m = 0.3

# Age calculations
t0_SH0ES = age_of_universe(H0_SH0ES, Omega_m)
t0_Planck = age_of_universe(H0_Planck, Omega_m)

print(f"SH0ES: H0 = {H0_SH0ES} ± {H0_SH0ES_err} km/s/Mpc (with Omega_m=0.30) → Age of th
print(f"Planck: H0 = {H0_Planck} ± {H0_Planck_err} km/s/Mpc (with Omega_m=0.30) → Age of
# Explore how age changes with different Omega_m
for Omega_m_test in [0.25, 0.3, 0.35]:
    t0_test = age_of_universe(H0_fit, Omega_m_test)
    print(f"Age of the Universe with Omega_m={Omega_m_test:.2f}: {t0_test:.2f} Gyr")
```

```
SH0ES: H0 = 72.97 ± 0.26 km/s/Mpc (with Omega_m=0.30) → Age of the Universe = 12.92 Gyr
Planck: H0 = 67.4 ± 0.5 km/s/Mpc (with Omega_m=0.30) → Age of the Universe = 13.99 Gyr
Age of the Universe with Omega_m=0.25: 13.58 Gyr
Age of the Universe with Omega_m=0.30: 12.92 Gyr
Age of the Universe with Omega_m=0.35: 12.37 Gyr
```

Questions:

1. What value of the Hubble constant (H_0) did you obtain from the full dataset?

Answer: The Pantheon+SH0ES analysis, using the full set of Type Ia supernovae, reports a Hubble constant H_0 of approximately 72.97 ± 0.26 km/s/Mpc.

---

2. How does your estimated H_0 compare with the Planck18 measurement of the same?

Answer: The Planck18 measurement (arXiv:1807.06209 [astro-ph.CO]), based on cosmic microwave background (CMB) data, gives H_0 ≈ 67.4 ± 0.5 km/s/Mpc. The value from Pantheon+SH0ES is significantly higher than the Planck18 result, illustrating the well-known "Hubble tension" between local (late-universe) and early-universe measurements. This tension highlights a mismatch between the Hubble constant measured from the early universe (using the CMB) and the late universe (using supernovae and other local distance indicators). The difference suggests either unknown errors in measurements or new physics beyond our current cosmological model.

---

3. What is the age of the Universe based on your value of H_0? (Assume Ω_m = 0.3). How does it change for different values of Ω_m?

Answer: With H_0 = 72.97 ± 0.26 km/s/Mpc and Ω_m = 0.30, the estimated age of the Universe is about 12.92 Gyr. If we use the lower Planck value (H_0 = 67.4 ± 0.5 km/s/Mpc), we get an older universe (around 13.99 Gyr). We also find that increasing Ω_m (matter density) decreases the age, while decreasing Ω_m increases it, since more matter slows expansion more over cosmic time.

---

4. Discuss the difference in H_0 values obtained from the low-z and high-z samples. What could this imply?

Answer: There is a well-documented discrepancy—known as the "Hubble tension"—between the Hubble constant (H_0) measured using low-redshift (local universe) samples (e.g., Type Ia supernovae) and high-redshift (early universe) samples (e.g., CMB).

Low-z measurements typically yield higher H_0 values (~72.97 ± 0.26 km/s/Mpc).

High-z measurements yield lower values H_0 (~67.4 ± 0.5 km/s/Mpc).

Implications:

This discrepancy (known as the Hubble tension) implies that:

- Our cosmological model (ΛCDM) may be incomplete.
- There could be new physics (e.g., early dark energy, evolving dark energy, or neutrino physics).
- Systematic errors might exist in one or both measurement techniques.

---

5. Plot the residuals and comment on any trends or anomalies you observe.

Answer: Residuals = Observed distance ($\mu_{\rm obs}$) − Model predicted distance ($\mu_{\rm model}$)

Plotting these against redshift (z), we may notice:

Expected trends:

- Flat, random scatter: Model fits well.
- Systematic curvature or trend: Model might be missing key features (e.g., accelerated expansion).

Possible anomalies:

- Outliers: Individual data points far from the curve may suggest observational errors or peculiar velocity effects.
- Systematic deviations at certain z :

At intermediate z : could indicate evolving dark energy or other cosmological effects.

At low z : may reflect local inhomogeneities (e.g., local voids).

---

6. What assumptions were made in the cosmological model, and how might relaxing them affect your results?

Answer:

Standard ΛCDM Assumptions:

- **Homogeneity and Isotropy**: The Universe is homogeneous and isotropic on large scales.
- **Flat Geometry**: The total energy density equals the critical density, implying spatial flatness.
- **Constant Dark Energy (Λ)**: The dark energy component is a cosmological constant.
- **Matter Content**: Includes cold dark matter (CDM) and baryons.
- **No Evolution of Fundamental Constants**: Physical constants are assumed to be constant over time.

Relaxing Assumptions:

- **Allowing Curvature ($\Omega_k \neq 0$)**: Introducing spatial curvature alters distance calculations, especially at high redshift.
- **Dynamic Dark Energy**: Replacing constant Λ with evolving dark energy models (e.g., w(z)) may better fit both low- and high-redshift data.
- **Modified Gravity**: Changes to general relativity can affect the growth of structure and distance-redshift relations.
- **Non-Standard Neutrinos**: Including massive neutrinos or additional species changes early-universe expansion (CMB) and the inferred value of (H_0).

---

7. Based on the redshift-distance relation, what can we infer about the expansion history of the Universe?

Answer:

Key Insights from the Redshift-Distance Relation:

- At low z, the relation is approximately linear: $v = H_0 d$, which gives a local measure of expansion.
- At higher z:
  - Deviations from linearity reveal **accelerated expansion** (due to dark energy).
  - The precise shape depends on the **matter-energy content** of the Universe.

Inferences:

🔭Identifying spectral lines in MIRI JWST data

Name: Aumansh Vijayendra Gupta

Email: aumansh.gupta119560@marwadiuniversity.ac.in

In this project, we will analyse the MIRI spectral cubes collected from JWST data for the source NGC 7469. We will analyse how to extract spectra from spectral cube. We will also identify emission lines from the source.

📡 JWST:

JWST has four main science instruments, each designed for specific types of imaging, spectroscopy, and coronagraphy across the infrared spectrum. | Instrument | Full Name | Primary Capabilities | Wavelength Range | | ---------- | ---------- | ---------- | ---------- | | **NIRCam** | Near-Infrared Camera | Imaging, weak lensing, coronagraphy | 0.6 – 5.0 μm | | **NIRSpec** | Near-Infrared Spectrograph | Multi-object spectroscopy, IFU, high-res spectra | 0.6 – 5.3 μm | | **MIRI** | Mid-Infrared Instrument | Mid-IR imaging, spectroscopy, coronagraphy | 5.0 – 28.5 μm | | **FGS/NIRISS** | Fine Guidance Sensor / Near-Infrared Imager and Slitless Spectrograph | Target acquisition, slitless spectroscopy, exoplanet transit observations | 0.6 – 5.0 μm |

We will analyse data collected by MIRI for the galaxy NGC 7469.

💡 JWST MIRI wavelength range & different channels:

```
from IPython.display import Image, display
display(Image(filename="Screenshot 2025-06-28 015402.png", width=900))
```

FOV name λ-range (μm)	FOV (arcsec)	Pixel size (arcsec)	Sub-band name	λ-range (μm)	Resolving power (λ/Δλ)
Channel 1 4.9–7.65	3.2 × 3.7	0.196	SHORT (A) MEDIUM (B) LONG (C)	4.90–5.74 5.66–6.63 6.53–7.65	3,320–3,710 3,190–3,750 3,100–3,610
Channel 2 7.51–11.7	4.0 × 4.8	0.196	SHORT (A) MEDIUM (B) LONG (C)	7.51–8.77 8.67–10.13 10.01–11.70	2,990–3,110 2,750–3,170 2,860–3,300
Channel 3 11.55–17.98	5.2 × 6.2	0.245	SHORT (A) MEDIUM (B) LONG (C)	11.55–13.47 13.34–15.57 15.41–17.98	2,530–2,880 1,790–2,640 1,980–2,790
Channel 4 17.7–27.9	6.6 × 7.7	0.273	SHORT (A) MEDIUM (B) LONG (C)	17.70–20.95 20.69–24.48 24.40–27.90	1,460–1,930 1,680–1,770 1,630–1,330

🌌 NGC 7469:

It is a Barred Spiral Galaxy, having a central bar-shaped structure of stars. NGC 7469 is located about 200 million light-years away from Earth, which means, given its apparent dimensions, that NGC 7469 is approximately 142,000 light-years across. NGC 7469 is a type I Seyfert galaxy, characterised by its bright nucleus. It is also a luminous infrared source with a powerful starburst embedded into its circumnuclear region. It has been observed in: JWST (MIRI + NIRSpec), Spitzer IRS, ALMA, HST (UV/optical imaging), Chandra (X-ray). | **Property** | **Value / Description** | | ---------- | ---------- | | **Galaxy Name** | NGC 7469 | | **Other Designations** | Mrk 1514, UGC 12332, IRAS F23007+0836 | | **Galaxy Type** | Seyfert 1 Galaxy with a Circumnuclear Starburst Ring | | **Morphological Type** | (R')SAB(rs)a (intermediate barred spiral with ring and spiral structure) | | **Constellation** | Pegasus | | **Redshift (z)** | 0.016268 | | **Distance** | 68–70 Mpc (220 million light-years) | | **Radial Velocity** | 4874 km/s | | **Apparent Magnitude (V)** | 12.3 | | **Angular Size** | 1.3 × 1.1 arcminutes | | **Physical Size** | 28,000 light-years across |

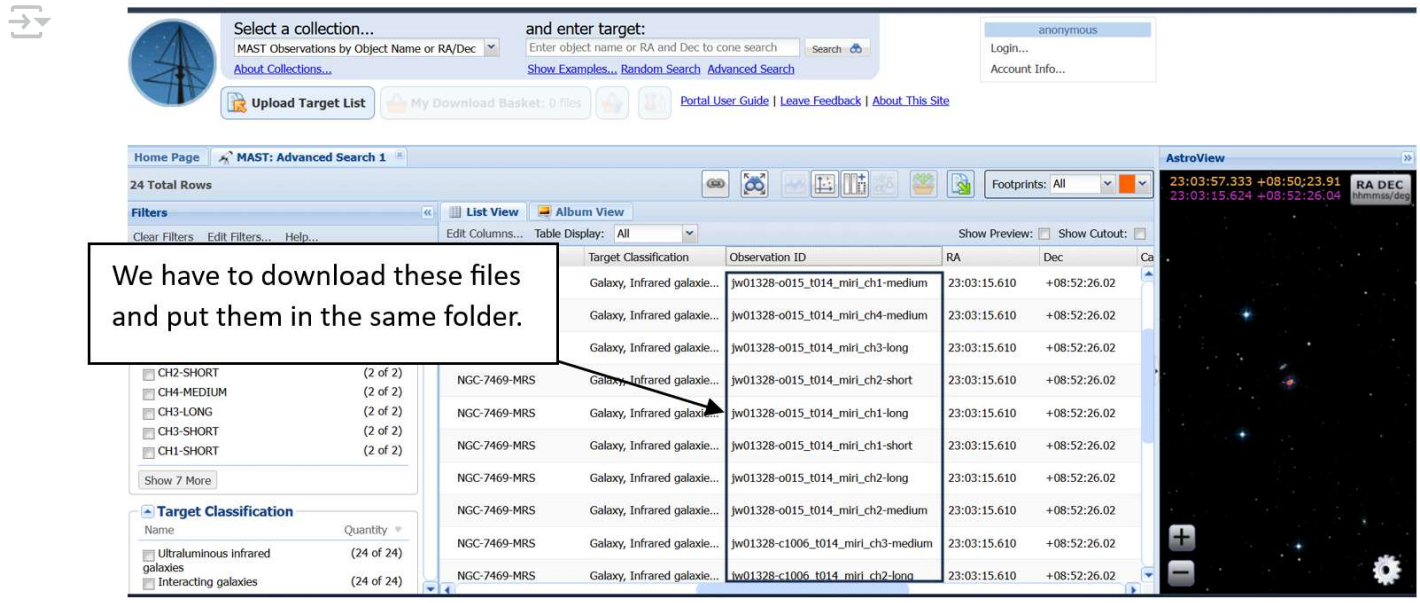
⬇️ Steps about how to download data:

First go to the MAST portal (https://mast.stsci.edu/portal/Mashup/Clients/Mast/Portal.html). Then go to advanced search. There, give the following input:

1. Object Name: NGC 7469

2. Observation Type: science
3. Mission: JWST
4. Instrument: MIRI/IFU
5. Product Type: cube

```
display(Image(filename="Screenshot 2025-06-28 023621.png", width=900))
```



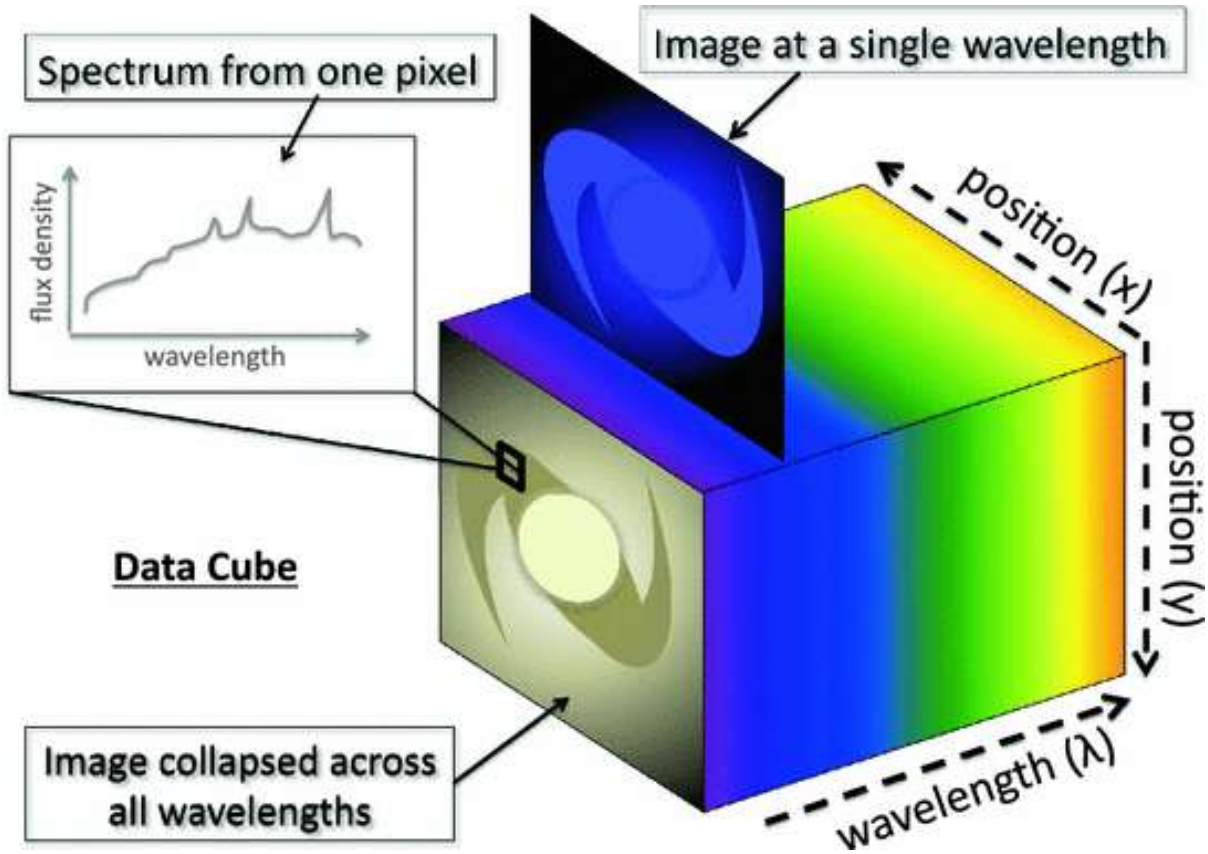
🧊 Spectral Cube:

A spectral cube is a three-dimensional (3D) astronomical data structure used in spectroscopy. It combines both spatial and spectral information into a single dataset, enabling detailed analysis of how light varies across position and wavelength. Unlike image that has 2 spatial axis, this has one spectral axis too. It can be in wavelength,frequency or velocity.

X-axis: spatial position (Right Ascension)
Y-axis: spatial position (Declination)
Z-axis: wavelength (or frequency or velocity)

JWST MIRI/MRS, NIRSpec IFU, ALMA, MUSE, and VLT SINFONI all produce spectral cubes.

```
display(Image(filename="Data Cube.png", width=600))
```



Regions:

Regions is an in-development coordinated package of Astropy for region handling.

The Regions package provides classes to represent:

```
Regions defined using pixel coordinates (e.g., CirclePixelRegion)
Regions defined using celestial coordinates, but still in an Euclidean geometry (e.g., Cir
To transform between sky and pixel regions, a world coordinate system object (e.g., astropy
```

Regions also provides a unified interface for reading, writing, parsing, and serializing regions data in different formats, including the DS9 Region Format, CRTF (CASA Region Text Format), and FITS Region Binary Table format.

Further Reference: https://astropy-regions.readthedocs.io/en/stable/shapes.html

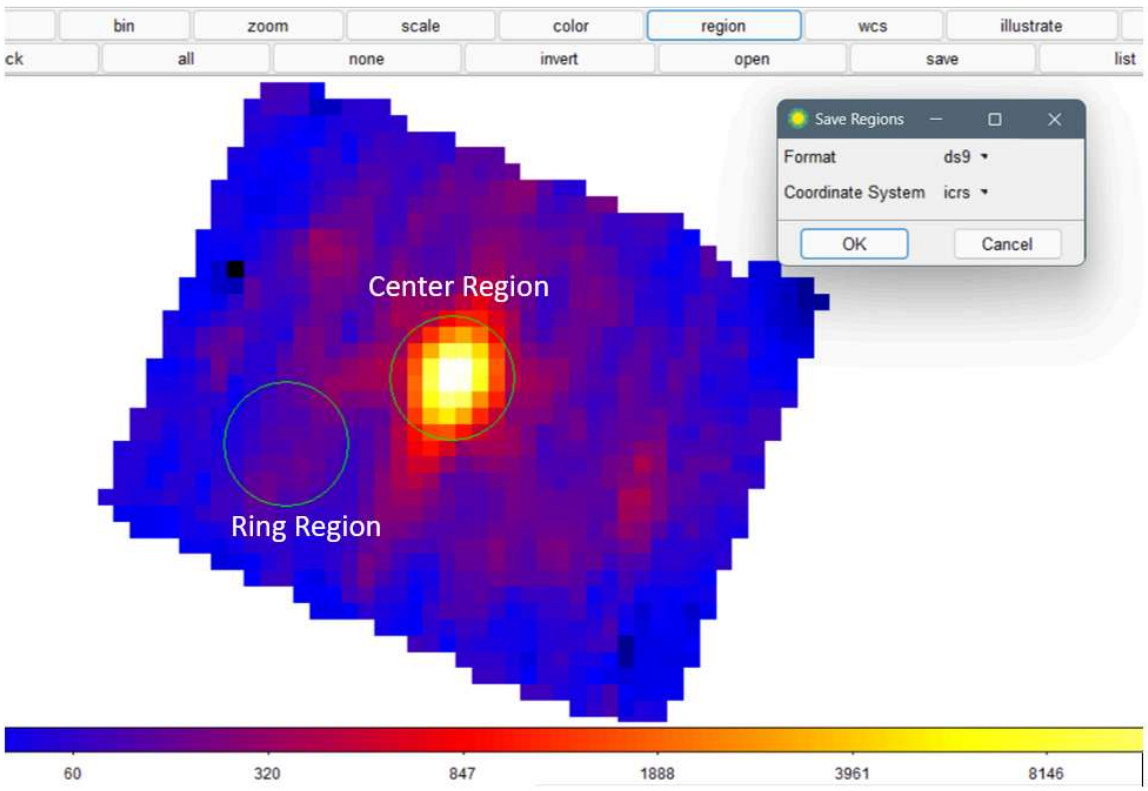
DS9:

SAOImage DS9 is a widely-used astronomical imaging and data visualization tool developed by the Smithsonian Astrophysical Observatory. It is primarily used to display and analyze FITS files (Flexible Image Transport System), which is the standard format for astronomical data. It is very

usefull to creat Overlays (regions, contours, catalogs, coordinate grids). It has World Coordinate System (WCS) support.

Open the file "ch1_short" and make the regions as shown figure with radius = 0.5 arcsec. Save the region file as .reg file with format 'ds9' and coordinate system as 'icrs' respectively.

```
display(Image(filename="Screenshot 2025-06-28 025716.png", width=600))
```



📝 Python Code to extract spectra from data cube:

We have selected this two regions and will iterate it over all FITS file for all 4 channels. We can proceed further and extract the spectra from each of the file and plot the final spectrum from it.

```
# Import necessary libraries
import numpy as np
import warnings
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.wcs import WCS
from regions import Regions

warnings.filterwarnings("ignore", category=UserWarning, append=True)
```

```
# Define the redshift for NGC 7469
# NGC 7469 is a Seyfert galaxy with a redshift of approximately 0.016268
# This value is used to convert observed wavelengths to rest-frame wavelengths.
# The redshift value can be obtained from various astronomical databases like NED (NASA/

# Redshift value
z = 0.016268

# Region file paths
region_files = {
    "Center Region": r"F:\PROJECTS\JWST MIRI\Region\Center Region.reg",
    "Ring Region": r"F:\PROJECTS\JWST MIRI\Region\Ring Region.reg"
}
print(region_files)
# FITS cube file paths
file_paths = []
for ch_num in range(1, 5):
    for part in ['short', 'medium', 'long']:
        #file_path = fr"F:\PROJECTS\JWST FITS\jw01328-c1006_t014_miri_ch{ch_num}-{part}_
        file_path = fr"F:\PROJECTS\JWST MIRI\FITS\jw01328-c1006_t014_miri_ch{ch_num}-{pa

        file_paths.append(file_path)

# Dictionary to store results per region
print(file_paths)
region_spectra = {}
```

```
{'Center Region': 'F:\\PROJECTS\\JWST MIRI\\Region\\Center Region.reg', 'Ring Region': '
['F:\\PROJECTS\\JWST MIRI\\FITS\\jw01328-c1006_t014_miri_ch1-short_s3d.fits', 'F:\\PROJE
```

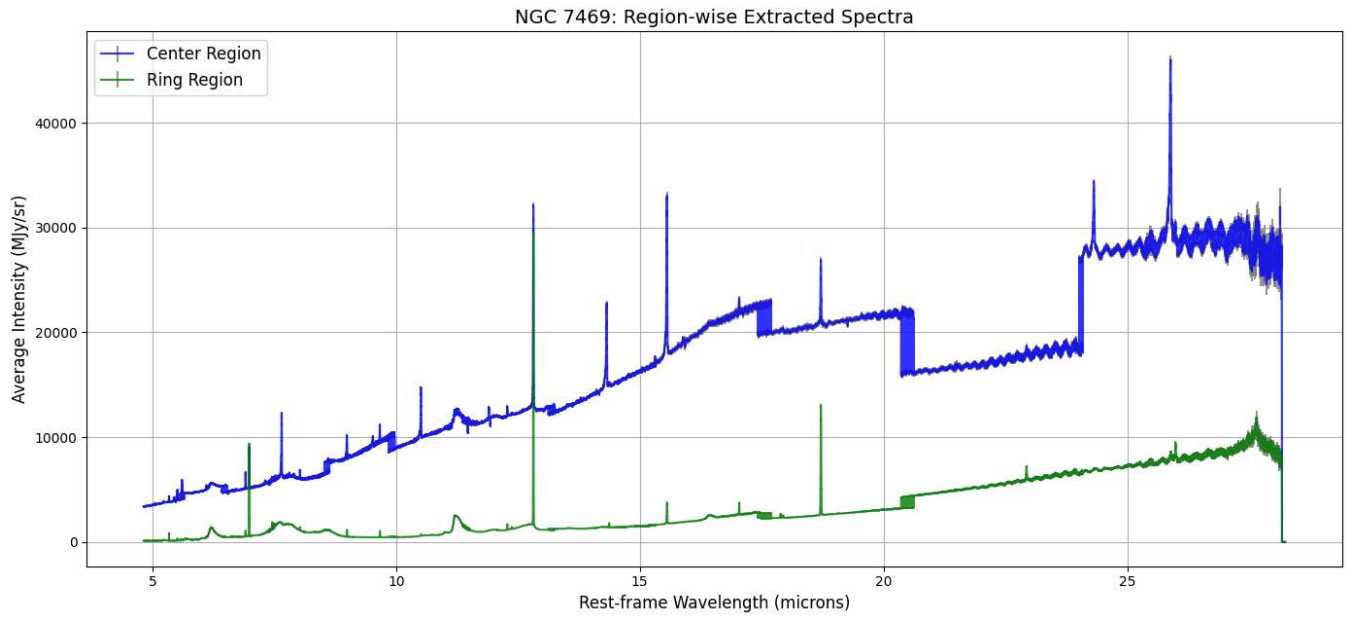
```
# Loop over each region
for region_name, reg_path in region_files.items():
    regions = Regions.read(reg_path, format='ds9')

    # Use two region files
    region = regions[0]

    # Initialize lists to store the spectrum and wavelength for all channels
    spectrum_all = []
    spectrum_all_err = []
    wavelength_all = []

    for file_path in file_paths:
        # Read data
        data = fits.open(file_path)[1].data      # 1 contain the 3D data cube
        data[data < 0] = np.nan                  # Reject those data having intensity le
        data_err = fits.open(file_path)[2].data  # 2 contain the error cube

        # Open the FITS file and get the WCS information to make mask of the region
```

```
        # Mask means the region of interest in the image - other parts are ignored
        header = fits.open(file_path)[1].header
        wcs = WCS(header)
        mask = region.to_pixel(wcs.celestial).to_mask()  # Convert the region from celes

        # Get the shape of the data
        num_channels, ny, nx = data.shape
        spectrum = []
        spectrum_err = []

        # Loop over each channel (wavelength)
        for i in range(num_channels):
            # Extract the 2D image from channels
            masked_data = np.array(mask.multiply(data[i, :, :]), dtype=float)
            masked_data_err = np.array(mask.multiply(data_err[i, :, :]), dtype=float)

            avg_intensity = np.nanmean(masked_data)
            avg_intensity_err = np.sqrt(np.nanmean(masked_data_err ** 2))

            if np.isnan(avg_intensity):
                avg_intensity = 0
            if np.isnan(avg_intensity_err):
                avg_intensity_err = 0

            spectrum.append(avg_intensity)
            spectrum_err.append(avg_intensity_err)

        # Extract the WCS information from the header to get the wavelength values
        # CRVAL3, CDELT3, and CRPIX3 are WCS keywords that define the wavelength axis

        # Wavelength axis
        crval3 = header['CRVAL3']   # CRVAL3 is the reference value of the wavelength ax
        cdelt3 = header['CDELT3']   # CDELT3 is the increment in the wavelength axis
        crpix3 = header['CRPIX3']   # CRPIX3 is the reference pixel of the wavelength ax

        wavelength = (np.arange(num_channels) - (crpix3 - 1)) * cdelt3 + crval3
        wavelength = wavelength / (1 + z) # Convert to rest-frame wavelength

        wavelength_all.extend(wavelength)
        spectrum_all.extend(spectrum)
        spectrum_all_err.extend(spectrum_err)

    # Sort all arrays by wavelength
    wavelength_all = np.array(wavelength_all)
    spectrum_all = np.array(spectrum_all)
    spectrum_all_err = np.array(spectrum_all_err)
    sort_idx = np.argsort(wavelength_all)

    region_spectra[region_name] = {
        "wavelength": wavelength_all[sort_idx],
        "spectrum": spectrum_all[sort_idx],
```

```
        "error": spectrum_all_err[sort_idx]
    }

import warnings
from astropy.wcs import FITSFixedWarning
warnings.simplefilter('ignore', category=FITSFixedWarning)
```

📈 Plotting the Intensity vs Wavelength data for selected two regions:

```
# PLOT COMPARATIVE SPECTRA FOR REGION 1 & 2
plt.figure(figsize=(14, 6.5))
colors = ['blue', 'green']
for idx, (region_name, data) in enumerate(region_spectra.items()):
    plt.errorbar(data["wavelength"], data["spectrum"],
                 yerr=data["error"],
                 label=region_name,
                 color=colors[idx], ecolor='gray', alpha=0.8)

plt.xlabel('Rest-frame Wavelength (microns)', fontsize=12)
plt.ylabel('Average Intensity (MJy/sr)', fontsize=12)
plt.title('NGC 7469: Region-wise Extracted Spectra', fontsize=14)
plt.grid(True)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()
```

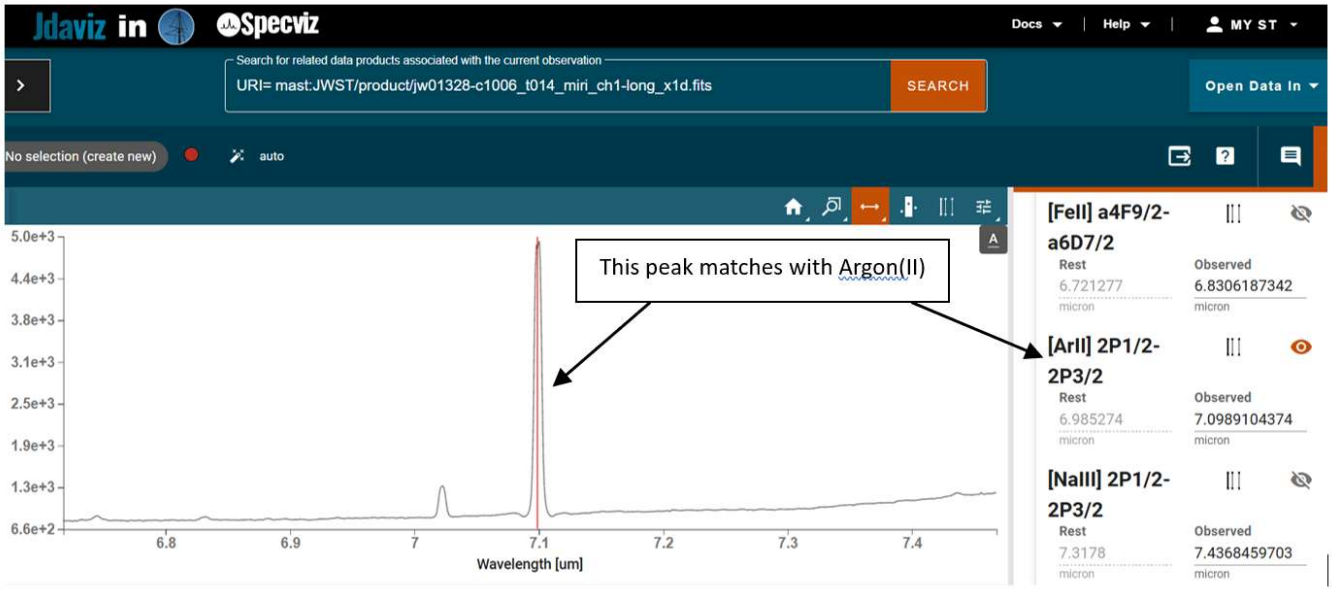


🌈Identifying the spectrum:

We will now identify the peaks that appear in the above graph. These peaks indicate the abundance of different elements in the galaxy.
We will use **Jdaviz/Cubeviz** database. We have to enter the redshift value of this galaxy and then we will check spectrum of each channel.

Fore more Info: https://jdaviz.readthedocs.io/en/stable/cubeviz/index.html

```
display(Image(filename="Screenshot 2025-06-28 211813.png", width=900))
```



✅List of observed peaks and corresponding wavengths (in microns) for each channels:

```
1 display(Image(filename="Screenshot 2025-06-28 222016.png", width=700))
```

Channel	Range	Transition	Wavelength(μm)
CH1	Short	H 0– 0 S(8) [FeII] a4F9/2-a6D9/2 [MgVII] 3P2–3P1 H 0 – 0S(7) [MgV]3P1 – 3P2	5.052 5.340169 5.5033 5.511 5.6098
	Medium	H 0 – 0S(6)	6.108
	Long	H 0 – 0S(5) [ArII] 2P1/2 – 2P3/2	6.909 6.985274
CH2	Short	[NeVI] 2P3/2 – 2P1/2 H 0 – 0S(4)	7.6524 8.025
	Medium	[ArIII] 3P1 – 3P2 H 0 – 0S(3)	8.99138 9.664
	Long	[SIV] 2P3/2-2P1/2	10.51049
CH3	Short	H 0 – 0S(2) [NeII] 2P1/2-2P3/2	12.279 12.81354
	Medium	[ClII] 3P1-3P2 [NeV] 3P2-3P1	14.3678 14.5546
	Long	[NeIII] 3P1 – 3P2 [CoIII] a4F5/2 – a4F7/2 H 0 – 0S(1)	15.55505 16.391 17.035
CH4	Short	[SIII] 3P2-3P1	18.71303
	Medium	[FeIII] 5D3-5D4	22.925
	Long	[NeV] 3P1-3P0 [OIV] 2P3/2-2P1/2 [FeII] a6D7/2-a6D9/2	24.3175 25.8903 25.98839

📈 Plotting the spectral lines:

```
import plotly.graph_objects as go
import numpy as np

# Initialize figure
fig = go.Figure(layout=dict(
    width=1280,
    height=600,
    template='plotly_white'
))

colors = ['blue', 'green', 'orange', 'purple', 'red']  # Add more if needed

# Plot region-wise spectrum and error bands
```

```
for idx, (region_name, data) in enumerate(region_spectra.items()):
    wavelength = np.array(data["wavelength"])
    spectrum = np.array(data["spectrum"])
    error = np.array(data["error"])

    # Add spectrum line
    fig.add_trace(go.Scatter(
        x=wavelength,
        y=spectrum,
        mode='lines',
        line=dict(color=colors[idx % len(colors)], width=1.5),
        name=region_name,
        hovertemplate='λ: %{x:.3f} μm<br>Intensity: %{y:.2f} MJy/sr<extra></extra>'
    ))

    # Add uncertainty band
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength, wavelength[::-1]]),
        y=np.concatenate([spectrum + error, (spectrum - error)[::-1]]),
        fill='toself',
        fillcolor='rgba(31, 119, 180, 0.2)' if colors[idx % len(colors)] == 'blue' else
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'{region_name} Uncertainty',
        showlegend=False
    ))

# Define features
features = {
    'PAHs': {'PAH 6.2': 6.2, 'PAH 7.7': 7.7, 'PAH 8.6': 8.6, 'PAH 11.3': 11.3, 'PAH 12.
    'H₂': {'H₂(1)': 17.035, 'H₂(2)': 12.279, 'H₂(3)': 9.664, 'H₂(4)': 8.025, 'H₂(5)': 6
    'Neon': {'Ne(II)': 12.81354, 'Ne(III)': 15.55505, 'Ne(V)(3P2-3P1)': 14.32168, 'Ne(V
    'Iron': {'Fe(III)': 22.925, 'Fe(II)(a6D7/2-a6D9/2)': 25.98839, 'Fe(II)(a4F9/2-a6D9/
    'Suphur': {'S(III)': 18.71303, 'S(IV)': 10.51049,},
    'Argon': {'Ar(II)': 6.985274, 'Ar(III)': 8.99138,},
    'Magnesium': {'Mg(V)': 5.6098, 'Mg(VII)': 5.5033,},
    'Other': {'Cl(II)': 14.3678, 'O(IV)': 25.8903, 'Co(III)': 16.391,},
}

# Feature line colors
colors_features = {
    'PAHs': '#FF7F0E',
    'H₂': "#002E50",
    'Neon': "#D62728",
    'Iron': "#601303",
    'Suphur': '#E377C2',
    'Argon': "#6009B2",
    'Magnesium': '#56FF01',
    'Other': '#17BECF',
}

```

```
# Add vertical lines and annotations
for category, lines in features.items():
    for name, wl in lines.items():
        fig.add_vline(
            x=wl,
            line=dict(
                color=colors_features[category],
                width=1.5 if category == 'PAHs' else 1,
                dash='solid' if category == 'PAHs' else 'dot'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(size=10, color=colors_features[category]),
                yshift=10 if category == 'PAHs' else 0
            )
        )

# Highlight PAH bands as shaded regions
for wl in [6.2, 7.7, 8.6, 11.3, 12.7, 16.4, 17.4]:
    fig.add_vrect(
        x0=wl - 0.15, x1=wl + 0.15,
        fillcolor=colors_features['PAHs'],
        opacity=0.1,
        line_width=0
    )

# Final layout
fig.update_layout(
    title='<b>NGC 7469 JWST/MIRI IFU Region-wise Spectra with Molecular and Atomic Feat
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',
    legend=dict(
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80)
)

# Show figure
fig.show()

```

🚀 Extracting spectra to a data-frame in pandas and exporting it into a CSV file:

```
import pandas as pd
from functools import reduce

# List to collect all individual region DataFrames
df_list = []

for region_name, spec_data in region_spectra.items():
    # Build DataFrame for this region
    df = pd.DataFrame({
        "Wavelength (µm)": spec_data["wavelength"],
        f"Flux ({region_name})": spec_data["spectrum"],
        f"Error ({region_name})": spec_data["error"]
    })

    df_list.append(df)

# Merge all DataFrames on "Wavelength (µm)"
# This assumes the wavelength arrays are identical or nearly identical
df_merged = reduce(lambda left, right: pd.merge(left, right, on="Wavelength (µm)", how='

# Optional: sort by wavelength if not already
df_merged = df_merged.sort_values("Wavelength (µm)")

# Save to a single CSV
combined_csv_path = "combined_spectra_all_regions.csv"
df_merged.to_csv(combined_csv_path, index=False)

print(f"\n✅ Combined spectra saved to: {combined_csv_path}")


from IPython.display import FileLink

# Display a clickable download link
FileLink("combined_spectra_all_regions.csv")

```

```
✅ Combined spectra saved to: combined_spectra_all_regions.csv
combined_spectra_all_regions.csv
```

📌 Output CSV Format Example:

Wavelength (µm)	Flux (Central)	Error (Central)	Flux (Ring)	Error (Ring)
5.100	1.23	0.05	1.01	0.06
5.105	1.20	0.04	1.00	0.05

Wavelength (μm)	Flux (Central)	Error (Central)	Flux (Ring)	Error (Ring)
...	...	...	...	...

❔ Questions:

1. What is the catagory and the subcatagory of the object?

🌀 The galaxy NGC 7469 has an active star formation and an active galactic nucleus (AGN). Type of galaxy is Seyfert Galaxy.
Subcatagories:
• Barred spiral galaxy
• Circumnuclear starburst ring
• Luminous Infrared Galaxy (LIRG)
• Composite system: AGN + nuclear starburst

A Seyfert galaxy is an active galaxy with a supermassive black hole at the center, has strong emission lines from ionized gas, and a bright core that can outshine the rest of the galaxy in X-ray and UV.

2. What does this category typically mean in the context of extragalactic astronomy?

🌌 In extragalactic astronomy, the category and subcategory of a galaxy provides crucial information about its structure, activity, evolution, and the physical processes occurring within it. Morphological Classification of NGC 7469 is (R')SAB(rs)a.

SAB:--- Weakly barred spiral (bar-like structure in the core).
R':-– Pseudo-ring formed by tightly wound spiral arms.
rs:--- Intermediate between a full ring and spiral arms.
a:--- Early-type spiral: large bulge, tightly wound arms.

Interpreted as a disk galaxy with both spiral and ring features, showing signs of internal dynamics (i.e. bar-driven gas inflow).

Circumnuclear Starburst Ring is a ring (typically ~1 kpc scale) of intense star formation encircling the galaxy nucleus. It is fed by inflowing gas, often driven by the galaxy's bar. These rings coexist with AGN and may be linked to AGN fueling via secular processes.

3. Why MIRI is important for studying objects like NGC 7469 and helps reveal hidden structures that Optical/NIR cannot?

Mid-Infrared (MIR) imaging is critical for studying galaxies like NGC 7469, especially those with active galactic nuclei (AGN) and circumnuclear starburst regions, because it allows astronomers to penetrate dense dust clouds and reveal structures that are invisible in Optical and Near-Infrared

(NIR) wavelengths.
Optical and NIR light are strongly absorbed and scattered by dust, making many regions (like galactic centers) opaque at these wavelengths. Whereas MIRI wavelengths (5–28 microns) pass through dust, allowing astronomers to see inside dusty star-forming regions and around AGN. MIR captures thermal radiation from dust grains heated by young, massive stars or AGN. These emissions trace star formation, even when stars themselves are obscured.

4. After plotting the spectra for both regions, is there any vertical shift in the spectra?

When plotting intensity vs. wavelength, the entire central spectrum is vertically shifted upward compared to the ring. I have selected two regions, one at the galaxy center(AGN), 'Center Region' and one at the circumnuclear ring (star-forming ring), 'Ring Region'.

📌 Reasons for the Vertical Shift:

1. 🔥 Intrinsic Brightness Difference:
   AGNs produce strong thermal continuum from hot dust, heated by the AGN's radiation field.
   The ring is primarily star-forming, emitting PAH features and molecular lines, but with a weaker MIR continuum.
   So, there is a true physical difference in brightness.
2. 🍼 Continuum Emission vs. Line Emission
   Central regions: Strong continuum + high-ionization lines ([NeV], [OIV]).
   Ring regions: Weaker continuum, more PAH features and H_2 lines.
   Because of the stronger continuum, the central region's spectrum appears elevated across all wavelengths.

5. Why we selected these two particular regions to analyse?

The reason is to Compare two Key Physical Environments in the Galaxy, the AGN and the Circumnuclear starbrust ring. Central AGN produces hard UV/X-ray radiation high-ionization lines, warm dust continuum. Ring is rich in gas and young stars → produces PAH bands, low-ionization lines, and H_2 emission.

6. Apart from any vertical shift, is there any differences in the spectral features between these two regions? List all the differences.

Feature Type	Spectral Feature	Difference	
PAH features	6.2, 7.7, 8.6, 11.3 μm	Stronger in **ring**	PAHs trace **st**
High-ionization lines	[Ne V] 14.3, [O IV] 25.9 μm	Stronger in **center**	Require **hard r**
Low-ionization lines	[Ne II] 12.8, [S III] 18.7, [Ar II] 6.98	Visible in both, maybe stronger in ring	**Photoionized**
Molecular hydrogen	H_2 0-0 S(1) to S(7)	Present in **both**, possibly stronger in ring	Warm **molecu**
Continuum emission	Full MIR continuum	Brighter in **center**	Hot dust emis

Feature Type	Spectral Feature	Difference	
Line widths	Narrow lines in ring, possible broadening in center	Possibly different	Could indicate

7. What could be the possible physical or astrophysical reasons behind these