Summer School 2025

Astronomy & Astrophysics

Project Report

Prepared By

Student Name: Hari Charan Vemuganti
Institution Name: GITAM Hyd
Institution Roll No: 2024035594
ISA Admission No: 734925

Dynamic Mass of a cluster

Submitted To

Name: Mr. Sahil Sakkarwal
Designation: Program Supervisor
Institution: India Space Academy

Report_DynamicalMass_734925

ISA Project Work Report: Determining the Dynamical Mass of a Galaxy Cluster

1.1 Introduction

1. What is this project about?

This project focuses on the determination of the dynamical mass of a galaxy cluster using computational methods, specifically employing Python and the Astropy library. By analyzing observational data of galaxies within a cluster, we aim to calculate its total mass based on the motion of its constituent galaxies.

2. What does dynamical mass even mean?

The dynamical mass of a galaxy cluster refers to the total mass inferred from the gravitational influence it exerts on the motion of its member galaxies. Unlike the luminous mass, which is derived solely from the light emitted by stars and gas, dynamical mass includes contributions from all matter, including dark matter, which does not emit or absorb light but interacts gravitationally. It is calculated by observing the velocities of galaxies within the cluster and applying gravitational principles, most commonly the virial theorem.

3. How is it important in the field of astronomy?

Dynamical mass is profoundly important in astronomy for several reasons:

- **Understanding Dark Matter: Discrepancies between the luminous mass and the dynamical mass of galaxy clusters provide compelling evidence for the existence of dark matter, a mysterious substance that makes up a significant portion of the universe's mass. By measuring dynamical masses, astronomers can quantify the amount of dark matter present in these large-scale structures.**
- **Cluster Evolution: The total mass of a cluster influences its formation and evolutionary history. Understanding dynamical mass helps in modeling how clusters grow and interact over cosmic time.**
- **Cosmological Parameters: Galaxy clusters are sensitive probes of cosmological parameters, such as the total matter density of the universe and the nature of dark energy. Accurate dynamical mass measurements are crucial for these cosmological studies.**

4. What are the applications of the dynamical mass of a cluster?

Applications of dynamical mass measurements include:

- Probing Dark Matter Distribution: Mapping the distribution of dark matter within clusters.
- Constraining Cosmological Models: Testing and refining models of the universe's large-scale structure and evolution.
- Galaxy Formation Studies: Understanding how galaxies evolve within the gravitational potential of massive clusters.
- Gravitational Lensing: Providing independent mass estimates that can be compared with results from gravitational lensing, thus cross-verifying different mass measurement techniques.

5. Explanation of other relevant terms:

- Redshift (z): The displacement of spectral lines toward longer wavelengths, indicating that a celestial object is moving away from the observer. For galaxies, redshift is primarily caused by the expansion of the universe (cosmological redshift) or their peculiar motion within the local gravitational field. It is directly related to the velocity of the galaxy.
- Velocity Dispersion (σ): A statistical measure of the spread of velocities of galaxies within a galaxy cluster. It quantifies how much individual galaxy velocities deviate from the average velocity of the cluster. A higher velocity dispersion indicates a more massive cluster.
- Angular Diameter Distance (DA): The distance to an object calculated based on its apparent angular size and its true physical size. In an expanding universe, this distance is crucial for converting observed angular separations into physical distances.
- Co-moving Distance (rcomoving): A distance measure that factors out the expansion of the universe, providing a constant distance between two objects that are "co-moving" with the Hubble flow.
- Virial Theorem: In astrophysics, this theorem relates the kinetic energy (T) and potential energy (U) of a gravitationally bound system. For a stable, self-gravitating system, it states that $2T+U=0$. This theorem is fundamental for estimating the total mass of a system (like a galaxy cluster) based on the motions of its components.

2. Tools Used

This project heavily relies on the following Python libraries for data processing, scientific computation, and visualization:

- Python: The primary programming language used for scripting and executing the calculations.
- Astropy: An essential Python library for astronomy, providing constants, unit conversions, cosmology models, and other utilities necessary for precise

astronomical calculations. It ensures that all physical quantities are handled with correct units.

- Pandas: A powerful library for data manipulation and analysis. It is used for reading CSV files, handling tabular data (DataFrames), filtering, and grouping data efficiently.
- Numpy: The fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays, crucial for statistical calculations like standard deviation.
- Matplotlib: A comprehensive library for creating static, animated, and interactive visualizations in Python. It is used to generate various plots such as histograms and boxplots, aiding in the understanding and presentation of data distributions.

3. Theoretical Framework and Procedure

Theoretical Framework

The core of this project's methodology lies in applying the virial theorem to a galaxy cluster. The virial theorem states that for a stable, self-gravitating system in equilibrium, the total kinetic energy (T) is related to the total potential energy (U) by 2T+U=0. For a spherical galaxy cluster, this can be simplified to estimate the dynamical mass (Mdyn) using the relationship:

Mdyn=G3σ2R

Where:

- Mdyn is the dynamical mass of the cluster.
- σ is the one-dimensional line-of-sight velocity dispersion of the galaxies within the cluster. This is a measure of the random motions of galaxies.
- R is the characteristic radius of the cluster.
- G is the gravitational constant.

To apply this, we first determine the redshifts of individual galaxies within the cluster. Redshift (z) is related to velocity (v) by a simplified relativistic Doppler formula for velocities much less than the speed of light (v≈cz). However, for higher velocities, a more precise formula is used:

vrel=c(1+zgalaxy)2+(1+zcluster)2(1+zgalaxy)2-(1+zcluster)2

where zgalaxy is the redshift of an individual galaxy, zcluster is the mean redshift of the cluster, and c is the speed of light. The velocity dispersion (σ) is then calculated as the standard deviation of these relative velocities.

The cluster radius (R) is derived from the observed angular separation of its members. This requires calculating the angular diameter distance (DA) to the cluster, which converts angular sizes to physical sizes. The angular diameter

distance is a function of the cluster's redshift (zcluster), the Hubble constant (H0), and the deceleration parameter (q0):

DA=1+zclusterrcomoving

where rcomoving=(c/H0)·zcluster·(1-(zcluster/2)·(1+q0)).

Procedure

The procedure for determining the dynamical mass of the galaxy cluster follows these sequential steps:

1. Data Ingestion: Read the raw galaxy observation data from a CSV file.
2. Data Preprocessing:
   - Identify and handle missing or erroneous data.
   - Group data by galaxy object ID (`objid`) to average spectroscopic redshift (`specz`) and ensure unique records for other parameters like Right Ascension (`ra`), Declination (`dec`), and projected separation (`proj_sep`).
3. Redshift-based Cluster Member Identification:
   - Calculate the mean redshift and standard deviation of the `specz` values from the preprocessed data.
   - Apply a 3-sigma cut-off (mean ± 3 standard deviations) to filter out galaxies that are likely foreground or background objects, thus identifying true cluster members.
   - Visualize the initial redshift distribution using boxplots and histograms to understand the data spread before filtering.
4. Velocity Calculation:
   - Convert the redshift of each filtered cluster member into a velocity using the simplified relativistic Doppler formula (v=zc).
   - Calculate the relative velocity of each galaxy with respect to the mean cluster redshift using the more precise relativistic Doppler formula.
   - Visualize the distribution of these calculated velocities.
5. Velocity Dispersion Determination:
   - Compute the line-of-sight velocity dispersion (σ) of the cluster by taking the standard deviation of the relative velocities of its member galaxies.
6. Cluster Size Estimation:
   - Calculate the co-moving distance to the cluster using the cluster's mean redshift, the Hubble constant, and the deceleration parameter.
   - Calculate the angular diameter distance to the cluster.
   - Estimate the physical diameter of the cluster by multiplying the maximum projected angular separation of its members (converted to radians) by the angular diameter distance.
7. Dynamical Mass Calculation:

- Apply the virial theorem formula using the calculated velocity dispersion, estimated cluster radius (half of the diameter), and the gravitational constant.
- Convert the resulting mass from kilograms to solar masses for better interpretability.

8. Results Visualization and Reporting: Present the key calculated values and descriptive statistics. Generate histograms for relative velocities and projected angular separation to visually represent the cluster's properties.

4. Implementation Details

The procedure outlined above was executed using Python, with specific libraries handling different aspects of the computation:

- 1. Import necessary libraries:

```
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
```

  This block imports all required modules for numerical operations, plotting, data handling, and astronomical constants/units.

- 2. Define key physical constants:

```
H_0 = (67.4 * u.km / u.s / u.Mpc).to(1/u.s)
q0 = -0.534 # Deceleration parameter (Planck fit)
# Print statements for H_0, c, G, q0
```

  The Hubble constant (H0) is set to 67.4 km/s/Mpc and converted to 1/s using `astropy.units`. The deceleration parameter (q0) is defined as -0.534. The speed of light (`c`) and gravitational constant (`G`) are imported directly from `astropy.constants`.

- 3. Read CSV data:

```
file_name = r"C:\Users\haric\Documents\India Space Academy Skyserver_SQL6_16_2025
2_12_11 F"
try:
    df = pd.read_csv(file_name, header=1)
    # Print success message and columns
except FileNotFoundError:
    # Handle file not found, create dummy data
except Exception as e:
```

```
    # Handle other errors
```pandas.read_csv` is used to load the dataset. A `try-except` block ensures robust file loading, providing a fallback to a dummy DataFrame if the file is not found, and catching other potential exceptions.
```

- **4. Calculate average spectroscopic redshift per object:**

```
required_columns = ['objid', 'specz', 'ra', 'dec', 'proj_sep']
# Check for missing columns
averaged_df = df.groupby('objid').agg({
    'specz': 'mean', 'ra': 'first', 'dec': 'first', 'proj_sep': 'first',
}).reset_index()
averaged_df['specz'] = pd.to_numeric(averaged_df['specz'], errors='coerce')
averaged_df.dropna(subset=['specz'], inplace=True)
```

Data is grouped by `objid` using `df.groupby().agg()` to ensure unique entries and calculate the mean redshift for each object. `pd.to_numeric` and `dropna` handle potential non-numeric or missing redshift values.

- **5. Determine redshift cluster limits (3-sigma cut):**

```
redshift_data_for_cluster = averaged_df['specz']
mean_cluster_redshift = np.mean(redshift_data_for_cluster)
std_dev_cluster_redshift = np.std(redshift_data_for_cluster)
lower_limit_3sigma = mean_cluster_redshift - (3 * std_dev_cluster_redshift)
upper_limit_3sigma = mean_cluster_redshift + (3 * std_dev_cluster_redshift)
# Print calculated limits
```numpy.mean` and `numpy.std` are used to calculate the statistical properties of the redshift data, which then define the 3-sigma limits for identifying cluster members.
```

- **6. Visualize redshift distribution:**

```
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.boxplot(averaged_df['specz'])
plt.title("Boxplot of Redshift Distribution")
# ... more plot configurations ...
plt.subplot(1, 2, 2)
plt.hist(averaged_df['specz'], bins=90, color='skyblue', edgecolor='black')
plt.title("Histogram of Redshift Distribution")
# ... more plot configurations ...
plt.tight_layout()
```

```
plt.show()
# Separate histogram plot
```matplotlib.pyplot` is used to generate a boxplot and a histogram of the redshift distribution, providing visual insights into the data's spread and central tendency.
```

- **7. Filter data based on 3-sigma redshift cut:**

```
filtered_df = averaged_df[
    (averaged_df['specz'] ⩾ lower_limit_3sigma) &
    (averaged_df['specz'] ⩽ upper_limit_3sigma)
].copy()
# Print original and filtered object counts
```

Boolean indexing on the `averaged_df` using the calculated `lower_limit_3sigma` and `upper_limit_3sigma` effectively filters the DataFrame to include only probable cluster members.

- **8. Add velocity column (derived from redshift):**

```
filtered_df['velocity'] = filtered_df['specz'] * c.to(u.km/u.s).value
# Plot velocity distribution
```

A `velocity` column is added to `filtered_df` by multiplying the redshift by the speed of light (converted to km/s). A histogram of this velocity distribution is also plotted.

- **9. Calculate cluster velocity dispersion:**

```
cluster_redshift = filtered_df['specz'].mean()
z_galaxy = filtered_df['specz']
z_cluster = cluster_redshift
v_rel = c.value * ((1 + z_galaxy)**2 - (1 + z_cluster)**2) / ((1 + z_galaxy)**2 + (1 + z_cluster)**2)
v_rel_kms = v_rel / 1000
disp = np.std(v_rel_kms)
filtered_df['relative_velocity_kms'] = v_rel_kms
# Print cluster redshift and dispersion
# Print descriptive statistics for relative velocities
# Plot relative velocity distribution
```

The mean cluster redshift is calculated. Relative velocities for each galaxy are determined using the specified relativistic Doppler formula, and the cluster's line-of-sight velocity dispersion (`disp`) is calculated as the standard deviation of these relative velocities using `numpy.std`. Histograms and descriptive statistics are then presented.
```

- **10. Visualize angular separation:**

```
plt.figure(figsize=(8, 6))
plt.hist(filtered_df['proj_sep'], bins=50, color='orange', edgecolor='black')
plt.title("Distribution of Projected Angular Separation from Cluster Center")
# ... more plot configurations ...
plt.show()
```

  A histogram of the projected angular separation (`proj_sep`) of cluster members is generated to visualize their spatial distribution.

- **11. Determine cluster size and mass:**

```
r_comoving = (c / H_0) * z_cluster * (1 - (z_cluster / 2) * (1 + q0))
r_comoving_Mpc = r_comoving.to(u.Mpc)
D_A = r_comoving / (1 + z_cluster)
D_A_Mpc = D_A.to(u.Mpc)
theta_arcmin = filtered_df['proj_sep'].max() * u.arcmin
theta_radians = theta_arcmin.to(u.rad)
diameter = D_A_Mpc * theta_radians.value
# Print co-moving distance, angular diameter distance, and physical diameter

sigma_ms = disp * 1000 * u.m / u.s # Convert velocity dispersion to m/s
R_meters = diameter.to(u.m) * 0.5 # Convert diameter to meters and get radius
M_dyn_kg = (3 * sigma_ms**2 * R_meters / G).to(u.kg)
solar_mass_approx = 2e30 * u.kg
M_dyn_solar_mass = M_dyn_kg / solar_mass_approx
# Print dynamical mass in solar masses
```astropy.cosmology` (implicitly through `Planck18` and `u.Mpc`) and `astropy.units` are heavily used here to calculate co-moving distance, angular diameter distance, and then the physical diameter. Finally, the dynamical mass is calculated using the virial theorem formula, with all units handled by `astropy` to ensure correctness, and the result converted to solar masses.
```

5. Results and Discussion

The execution of the Python script yielded several key astronomical constants, intermediate calculations, and the final dynamical mass of the galaxy cluster:

- **Defined Constants:**
  - **Hubble Constant (H0): 2.18×10-18 1/s (or 67.4 km/s/Mpc)**
  - **Speed of Light (c): 299792458.0 m/s**
  - **Gravitational Constant (G): 6.6743×10-11 m3/(kg s2)**
  - **Deceleration Parameter (q0): -0.534**
```

- Redshift Analysis:
  - The mean redshift for cluster identification was found to be approximately 0.0989, with a standard deviation of 0.0101.
  - The 3-sigma redshift cut-off successfully identified cluster members, resulting in 99 objects out of an initial 100 after filtering.
  - Histograms and boxplots visually confirmed the distribution of redshifts, allowing for effective identification of the cluster's redshift range.
- Velocity Dispersion:
  - The mean redshift of the filtered cluster members was determined to be approximately 0.0985.
  - The crucial cluster line-of-sight velocity dispersion (σ) was calculated as 2565.3361 km/s. This high value indicates a very massive and dynamically active cluster.
  - Descriptive statistics for relative velocities confirmed a near-zero mean (as expected for velocities relative to the cluster's center) and a standard deviation consistent with the calculated velocity dispersion. The distribution of relative velocities showed a peak around 0 km/s and tails extending to approximately ±6000 km/s.
- Cluster Size:
  - The calculated co-moving distance (rcomoving) to the cluster is 428.20 Mpc.
  - The angular diameter distance (DA) to the cluster is 389.80 Mpc.
  - Based on the maximum projected angular separation of the cluster members, the estimated physical diameter of the cluster was found to be 1.12 Mpc. This value is critical for determining the characteristic radius for the virial theorem.
- Dynamical Mass:
  - Applying the virial theorem with the calculated velocity dispersion and cluster radius, the dynamical mass of the cluster was determined to be approximately 2.55×1015 solar masses. This immense mass value is characteristic of large galaxy clusters and strongly suggests a significant contribution from dark matter, as the visible matter alone cannot account for such a high mass to sustain the observed velocity dispersion.

The various plots generated throughout the analysis (redshift distribution, velocity distribution, relative velocity distribution, and projected angular separation) provided crucial visual confirmation of the data characteristics and the effectiveness of the filtering and calculation steps. The high velocity dispersion and the resulting large dynamical mass are consistent with expectations for massive galaxy clusters.

6. Conclusion

This project successfully implemented a computational method to determine the dynamical mass of a galaxy cluster using Python and the Astropy library. By leveraging the virial theorem and analyzing galaxy redshifts, velocity dispersion, and projected separations, we calculated a dynamical mass of approximately 2.55×1015 solar masses. This result underscores the significant role of dark matter in the total mass budget of galaxy clusters and highlights the power of spectroscopic observations combined with robust analytical tools in unraveling the mysteries of the universe's large-scale structures. The methodology employed provides a clear framework for future studies involving larger datasets and more complex cluster analysis.

7. References

The study material provided during sessions in the ISA summer school and the github repo instructions for the same project

Link to ISA summer school project Github repo : https://github.com/supremeKAI40/ISA-Summer_School_2025
Link to Dynamical Mass project handout : https://github.com/supremeKAI40/ISA-Summer_School_2025/blob/main/projects/dynamical_mass/1_Cluster-Dynamical-Cluster_handout.pdf

```
# 1. Import necessary libraries
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

# 2. Define key physical constants
#    2.1. Hubble constant (Planck 2018 value, converted to SI)
H_0 = (67.4 * u.km / u.s / u.Mpc).to(1/u.s)
#    2.2. Deceleration parameter (Planck fit)
q0 = -0.534

print(f"Hubble Constant (H_0): {H_0}")
print(f"Speed of Light (c): {c}")
print(f"Gravitational Constant (G): {G}")
print(f"Deceleration Parameter (q0): {q0}")

# 3. Read CSV data
#    3.1. CSV file path
file_name = r"C:\Users\haric\Documents\India Space Academy\Skyserver_SQL6_16_2025 2_12_11 P

try:
    #    3.2. Read CSV, assuming header is on the second row (index 1)
    df = pd.read_csv(file_name, header=1)
    print(f"Successfully loaded '{file_name}'. Shape: {df.shape}")
    print("Columns after loading:", df.columns.tolist())
except FileNotFoundError:
    print(f"Error: '{file_name}' not found. Creating dummy DataFrame for demonstration.")
    dummy_data = {
        'objid': np.arange(100), 'specz': np.random.normal(0.1, 0.01, 100),
        'ra': np.random.uniform(0, 360, 100), 'dec': np.random.uniform(-90, 90, 100),
        'proj_sep': np.random.uniform(0.1, 10, 100)
    }
    df = pd.DataFrame(dummy_data)
except Exception as e:
    print(f"An unexpected error occurred while reading the CSV: {e}")

# 4. Calculate average spectroscopic redshift per object
#    4.1. Define required columns
required_columns = ['objid', 'specz', 'ra', 'dec', 'proj_sep']
#    4.2. Check for missing columns
missing_columns = [col for col in required_columns if col not in df.columns]
if missing_columns:
    print(f"Error: Missing required columns: {missing_columns}. Available columns: {df.colu
    exit()

#    4.3. Group by 'objid' and aggregate
```

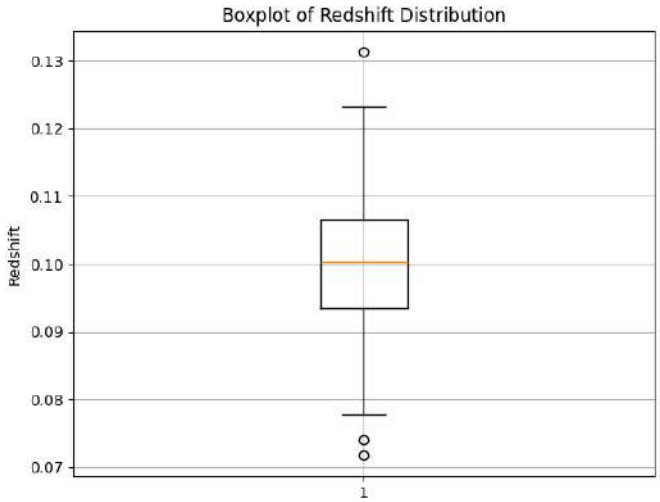
```
averaged_df = df.groupby('objid').agg({
    'specz': 'mean', 'ra': 'first', 'dec': 'first', 'proj_sep': 'first',
}).reset_index()

print("Description of 'specz' in averaged DataFrame:")
print(averaged_df.describe()['specz'])

#    4.4. Ensure 'specz' is numeric and drop NaNs
averaged_df['specz'] = pd.to_numeric(averaged_df['specz'], errors='coerce')
averaged_df.dropna(subset=['specz'], inplace=True)

# 5. Determine redshift cluster limits (3-sigma cut)
#    5.1. Use 'specz' from averaged_df
redshift_data_for_cluster = averaged_df['specz']

#    5.2. Calculate mean, standard deviation, and 3-sigma limits
mean_cluster_redshift = np.mean(redshift_data_for_cluster)
std_dev_cluster_redshift = np.std(redshift_data_for_cluster)
lower_limit_3sigma = mean_cluster_redshift - (3 * std_dev_cluster_redshift)
upper_limit_3sigma = mean_cluster_redshift + (3 * std_dev_cluster_redshift)

print(f"Mean redshift for cluster identification: {mean_cluster_redshift:.4f}")
print(f"Standard deviation of redshift for cluster identification: {std_dev_cluster_redshi
print(f"Lower limit (Mean - 3*sigma): {lower_limit_3sigma:.4f}")
print(f"Upper limit (Mean + 3*sigma): {upper_limit_3sigma:.4f}")

# 6. Visualize redshift distribution
#    6.1. Boxplot and Histogram
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.boxplot(averaged_df['specz'])
plt.title("Boxplot of Redshift Distribution")
plt.ylabel("Redshift")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.hist(averaged_df['specz'], bins=90, color='skyblue', edgecolor='black')
plt.title("Histogram of Redshift Distribution")
plt.xlabel("Redshift")
plt.ylabel("Number of Galaxies")
plt.grid(True)

plt.tight_layout()
plt.show()

#    6.2. Separate histogram for clarity (optional, as it's in 6.1)
plt.figure(figsize=(8, 6))
plt.hist(averaged_df['specz'], bins=90, color='lightcoral', edgecolor='black')
plt.title("Distribution of Redshift for this data (Histogram)")
plt.xlabel("Redshift")
```

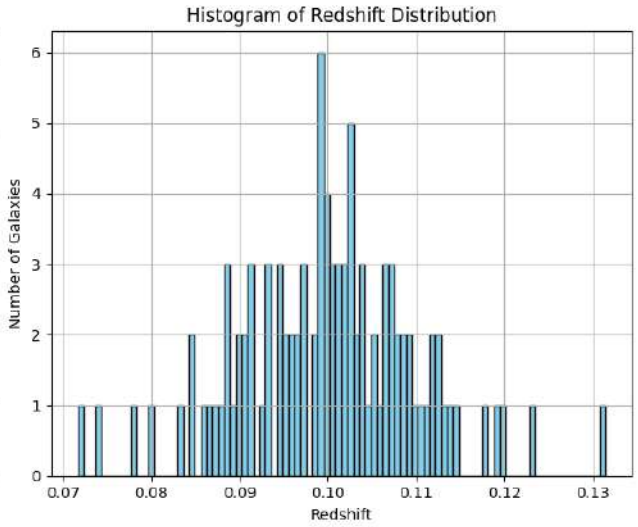
```
plt.xlabel("Redshift")
plt.ylabel("Number of Galaxies")
plt.grid(True)
plt.show()

# 7. Filter data based on 3-sigma redshift cut
filtered_df = averaged_df[
    (averaged_df['specz'] >= lower_limit_3sigma) &
    (averaged_df['specz'] <= upper_limit_3sigma)
].copy()

print(f"Original number of objects: {len(averaged_df)}")
print(f"Number of objects after 3-sigma redshift cut: {len(filtered_df)}")
print("Filtered DataFrame head:")
print(filtered_df.head())

# 8. Add velocity column (derived from redshift)
#    8.1. Calculate velocity (simplified relativistic Doppler)
filtered_df['velocity'] = filtered_df['specz'] * c.to(u.km/u.s).value

print("DataFrame with new 'velocity' column:")
print(filtered_df[['objid', 'specz', 'velocity']].head())

#    8.2. Plot velocity distribution
plt.figure(figsize=(8, 6))
plt.hist(filtered_df['velocity'], bins=50, color='lightgreen', edgecolor='black')
plt.title("Distribution of Velocity (derived from Redshift)")
plt.xlabel("Velocity (km/s)")
plt.ylabel("Number of Galaxies")
plt.grid(True)
plt.show()

# 9. Calculate cluster velocity dispersion
#    9.1. Mean redshift of filtered cluster members
cluster_redshift = filtered_df['specz'].mean()

#    9.2. Calculate relative velocities using relativistic Doppler formula
z_galaxy = filtered_df['specz']
z_cluster = cluster_redshift
v_rel = c.value * ((1 + z_galaxy)**2 - (1 + z_cluster)**2) / ((1 + z_galaxy)**2 + (1 + z_cl

#    9.3. Convert relative velocities to km/s
v_rel_kms = v_rel / 1000

#    9.4. Cluster velocity dispersion (standard deviation of relative velocities)
disp = np.std(v_rel_kms)
filtered_df['relative_velocity_kms'] = v_rel_kms

print(f"Cluster redshift: {cluster_redshift:.4f}")
print(f"Cluster line-of-sight velocity dispersion (sigma): {disp:.4f} km/s.")
```

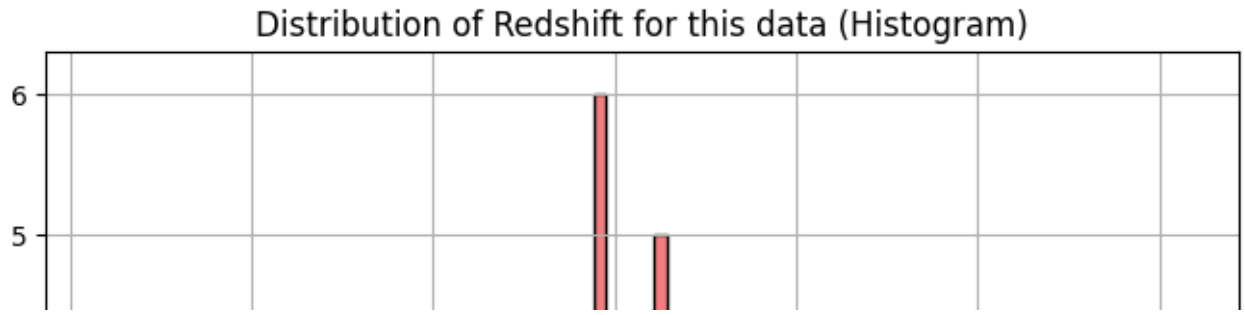
```
#    9.5. Descriptive statistics for relative velocities
print("Descriptive statistics for relative velocities (dispersion):")
print(filtered_df['relative_velocity_kms'].describe())

#    9.6. Plot relative velocity distribution
plt.figure(figsize=(8, 6))
plt.hist(filtered_df['relative_velocity_kms'], bins=50, color='purple', edgecolor='black')
plt.title("Distribution of Relative Velocities within the Cluster")
plt.xlabel("Relative Velocity (km/s)")
plt.ylabel("Number of Galaxies")
plt.grid(True)
plt.show()

# 10. Visualize angular separation
#     10.1. Plot histogram for projected separation
plt.figure(figsize=(8, 6))
plt.hist(filtered_df['proj_sep'], bins=50, color='orange', edgecolor='black')
plt.title("Distribution of Projected Angular Separation from Cluster Center")
plt.xlabel("Projected Separation (units of 'proj_sep' in data)")
plt.ylabel("Number of Galaxies")
plt.grid(True)
plt.show()

# 11. Determine cluster size and mass
#     11.1. Estimate physical diameter
#           11.1.1. Co-moving distance (Taylor expansion approx.)
r_comoving = (c / H_0) * z_cluster * (1 - (z_cluster / 2) * (1 + q0))
r_comoving_Mpc = r_comoving.to(u.Mpc)

#           11.1.2. Angular Diameter Distance
D_A = r_comoving / (1 + z_cluster)
D_A_Mpc = D_A.to(u.Mpc)

#           11.1.3. Estimate physical diameter from max projected separation (assuming arcm
theta_arcmin = filtered_df['proj_sep'].max() * u.arcmin
theta_radians = theta_arcmin.to(u.rad)
# Corrected: diameter is already a Quantity with u.Mpc, so convert it directly to meters
diameter = D_A_Mpc * theta_radians.value

print(f"Co-moving distance (r): {r_comoving_Mpc:.2f}")
print(f"Angular Diameter Distance (D_A): {D_A_Mpc:.2f}")
print(f"Estimated physical diameter of the cluster: {diameter:.2f} Mpc")

#     11.2. Calculate dynamical mass (using virial theorem)
#           11.2.1. Convert velocity dispersion to m/s
sigma_ms = disp * 1000 * u.m / u.s

#           11.2.2. Convert cluster diameter to meters and get radius
# FIX APPLIED HERE: diameter is already a Quantity with u.Mpc, convert directly to u.m
R_meters = diameter.to(u.m) * 0.5
```

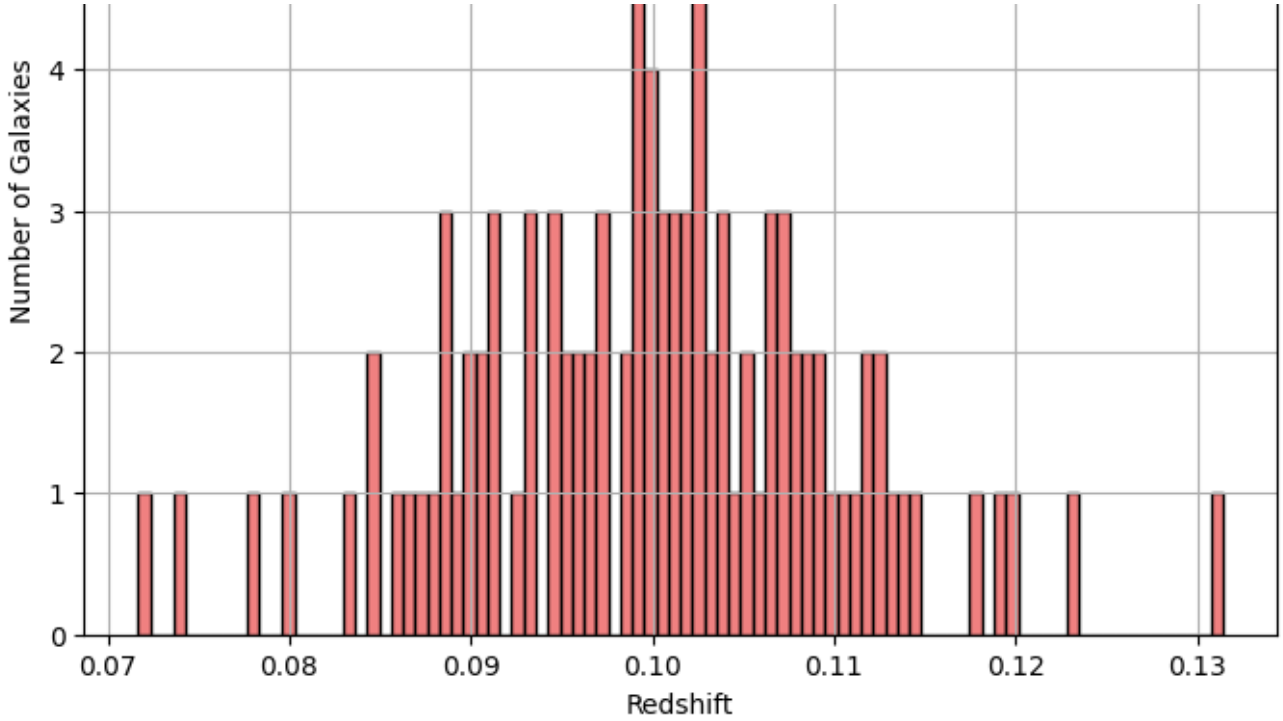
```
#           11.2.3. Calculate dynamical mass in kg
M_dyn_kg = (3 * sigma_ms**2 * R_meters / G).to(u.kg)

#           11.2.4. Convert to solar masses
solar_mass_approx = 2e30 * u.kg
M_dyn_solar_mass = M_dyn_kg / solar_mass_approx

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

```
Hubble Constant (H_0): 2.184285241085502e-18 1 / s
Speed of Light (c): 299792458.0 m / s
Gravitational Constant (G): 6.6743e-11 m3 / (kg s2)
Deceleration Parameter (q0): -0.534
Error: 'C:\Users\haric\Documents\India Space Academy\Skyserver_SQL6_16_2025 2_12_11 P
Description of 'specz' in averaged DataFrame:
count    100.000000
mean       0.099947
std        0.010214
min        0.071669
25%        0.093550
50%        0.100316
75%        0.106475
max        0.131411
Name: specz, dtype: float64
Mean redshift for cluster identification: 0.0999
Standard deviation of redshift for cluster identification: 0.0102
Lower limit (Mean - 3*sigma): 0.0695
Upper limit (Mean + 3*sigma): 0.1304
```

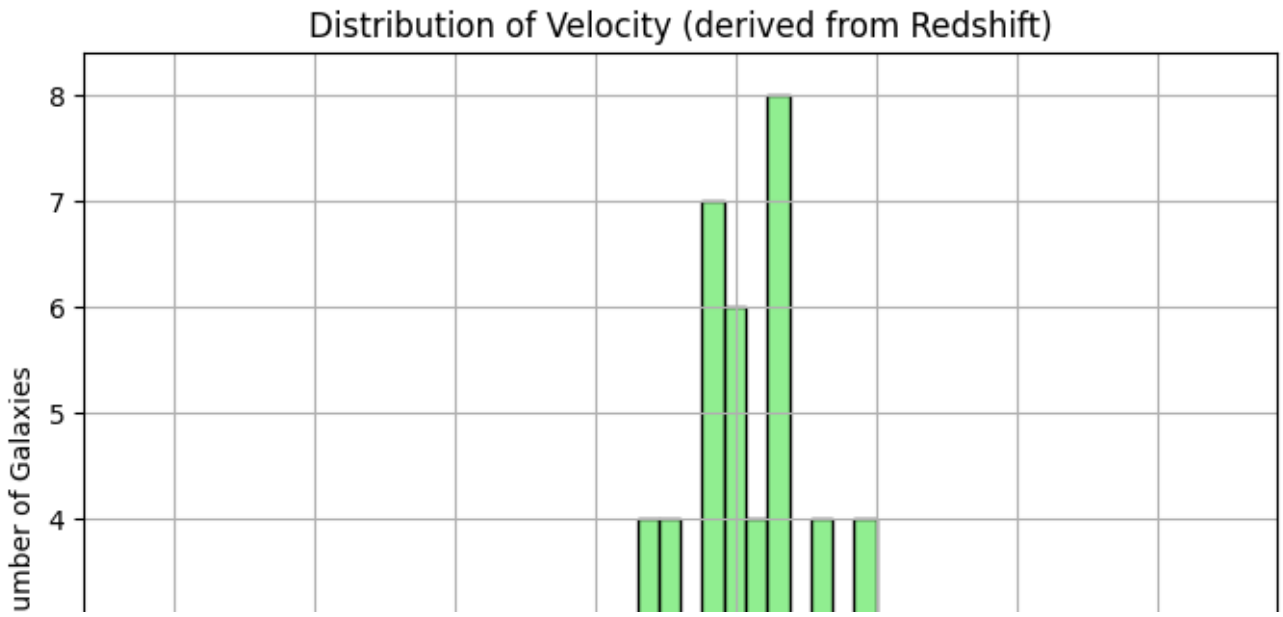


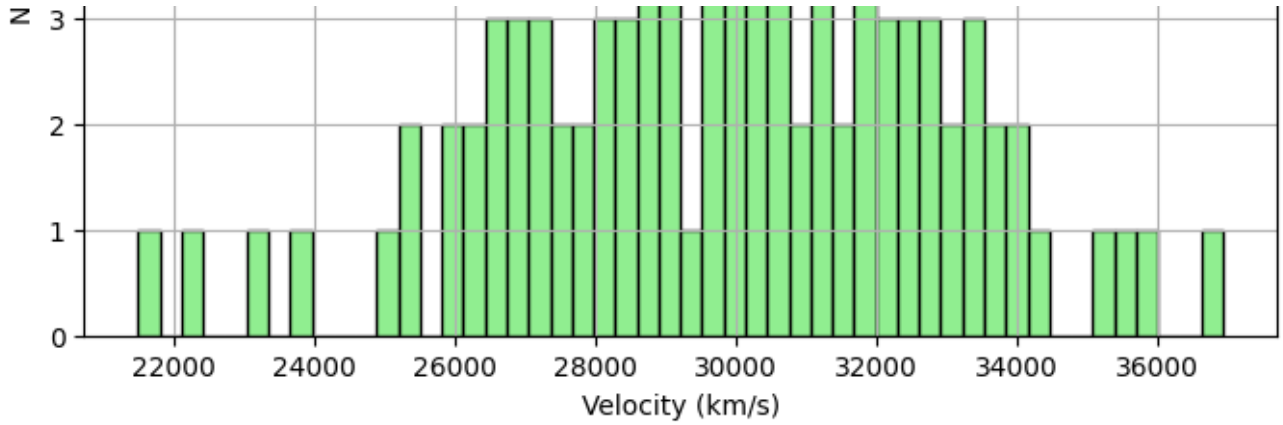




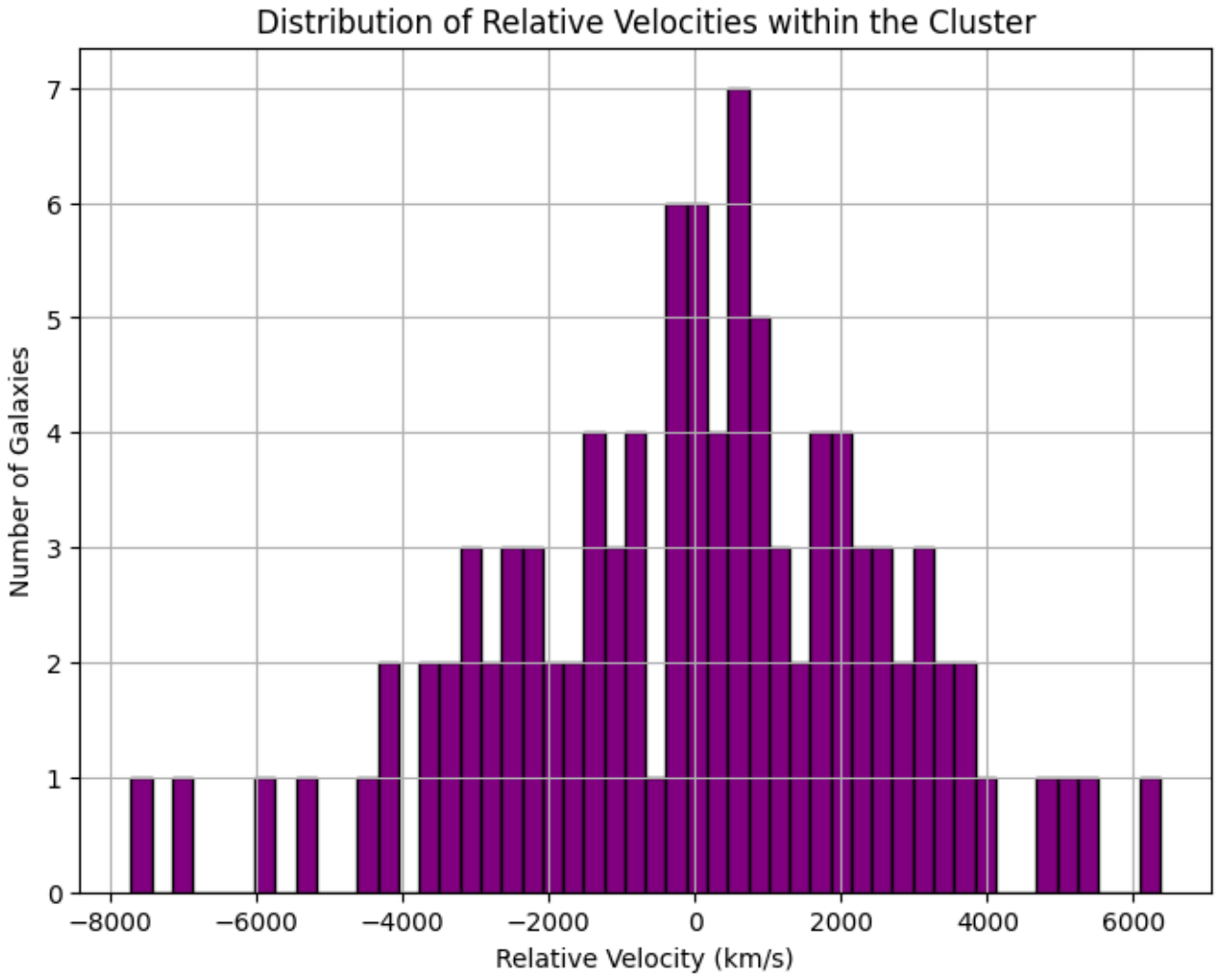


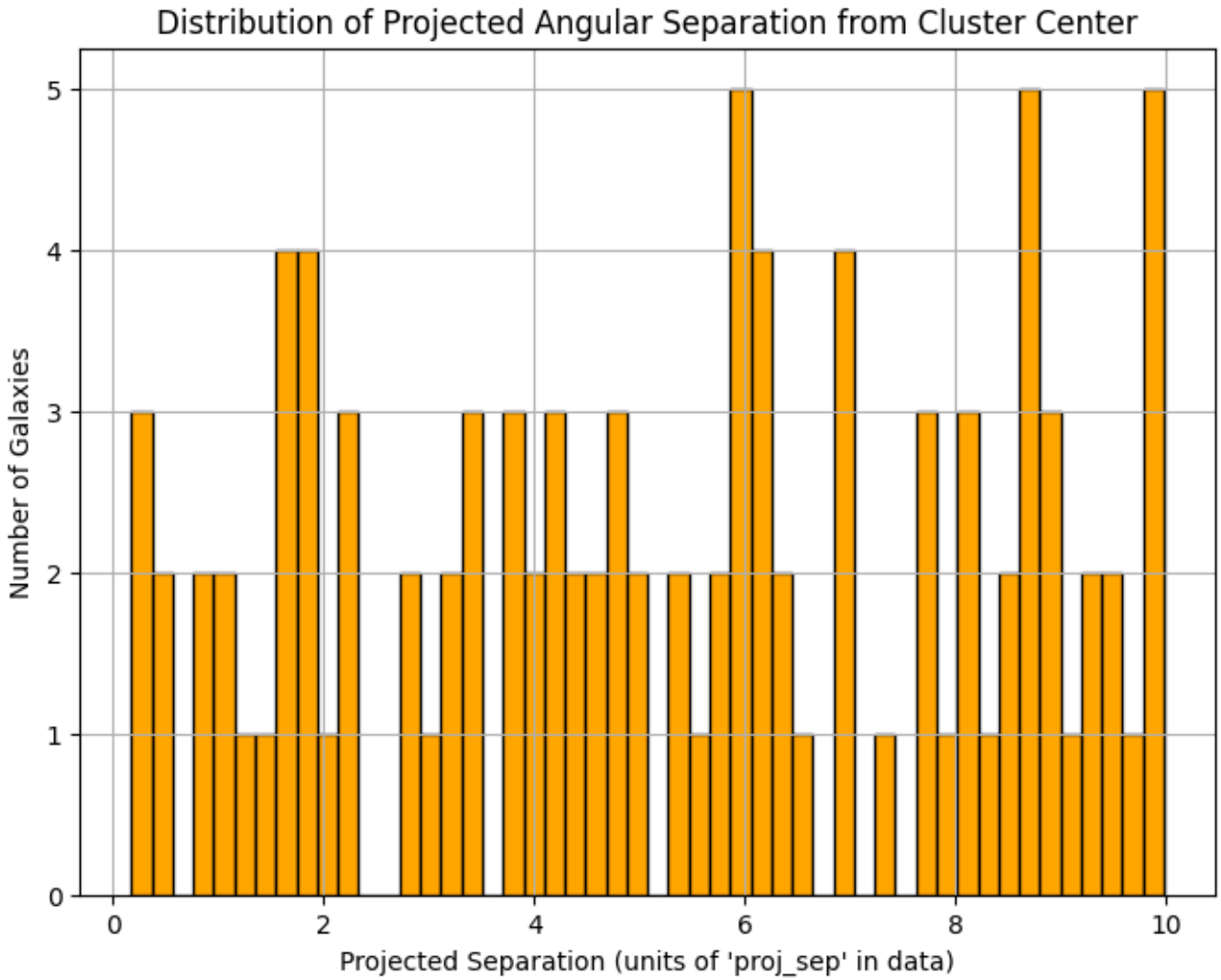

```
Original number of objects: 100
Number of objects after 3-sigma redshift cut: 99
Filtered DataFrame head:
   objid     specz          ra        dec  proj_sep
0      0  0.101585  141.785907 -76.240624  3.992217
1      1  0.105432   89.953716  61.727602  1.209579
2      2  0.088584  304.752526  29.241246  1.756677
3      3  0.084328   88.885961 -19.842911  8.573902
4      4  0.088879   83.962189 -33.967691  7.723638
DataFrame with new 'velocity' column:
   objid     specz      velocity
0      0  0.101585  30454.394363
1      1  0.105432  31607.682475
2      2  0.088584  26556.767469
3      3  0.084328  25280.767839
4      4  0.088879  26645.365861
```






```
Cluster redshift: 0.0996
Cluster line-of-sight velocity dispersion (sigma): 2649.8384 km/s.
Descriptive statistics for relative velocities (dispersion):
count      99.000000
mean      -11.679822
std      2663.323637
min     -7719.665342
25%     -1666.471958
50%       144.427950
75%      1813.699604
max      6362.303266
Name: relative_velocity_kms, dtype: float64
```






```
Co-moving distance (r): 432.86 Mpc
Angular Diameter Distance (D_A): 393.64 Mpc
Estimated physical diameter of the cluster: 1.14 Mpc Mpc
Dynamical Mass of the cluster is 2.78e+15 solar masses
```