Taller Semana De la Enseñanza de la fisica

Guía Paso a Paso: Mapeo de Galaxias y Cálculo de la Edad del Universo

Objetivos

1. Visualizar la distribución de galaxias observadas en el cielo utilizando su Ascensión Recta (RA) y Declinación (Dec).
2. Crear un mapa 3D de las galaxias utilizando sus coordenadas de ascensión recta (RA) y declinación (Dec) y el corrimiento al rojo (z) para estimar la distancia
3. Análisis de Estructuras a Gran Escala: Clustering de Galaxias en SDSS
4. Calcular la edad del universo utilizando la constante de Hubble, obtenida a partir de la relación entre la velocidad de recesión y la distancia de las galaxias.

Paso 1: Instalación de Librerías

Para trabajar con datos y realizar visualizaciones, vamos a utilizar las librerías `pandas`, `matplotlib`, y `scipy`. Instálalas con el siguiente comando (si no las tienes ya):

```
pip install pandas matplotlib scipy
```

Paso 2: Cargar los Datos de Galaxias

Para esta guía, vamos a trabajar con un archivo CSV que contiene información sobre galaxias, incluyendo:

- Ascensión Recta (RA) y Declinación (Dec): posiciones angulares de las galaxias en el cielo.
- Velocidad de recesión: velocidad a la que se alejan de nosotros debido a la expansión del universo.
- Distancia en megapársecs (`distance_Mpc`): distancia de la galaxia a la Tierra en unidades de megapársecs (Mpc).

Código para cargar los datos:

```
import pandas as pd

# Especifica la ruta del archivo CSV con tus datos
file_path = r"C:\Users\pequi\Downloads\sdss_data_with_calculations.csv"

# Cargar los datos en un DataFrame
data = pd.read_csv(file_path)

# Mostrar los primeros registros para verificar
print(data.head())
```

Explicación:

- `file_path` es la ruta donde tienes el archivo CSV. Asegúrate de que la ruta sea correcta.
- `pd.read_csv(file_path)` carga los datos en un DataFrame de pandas llamado `data`.
- `data.head()` muestra las primeras filas del archivo para asegurarnos de que los datos se cargaron correctamente.

Paso 3: Mapeo de Galaxias

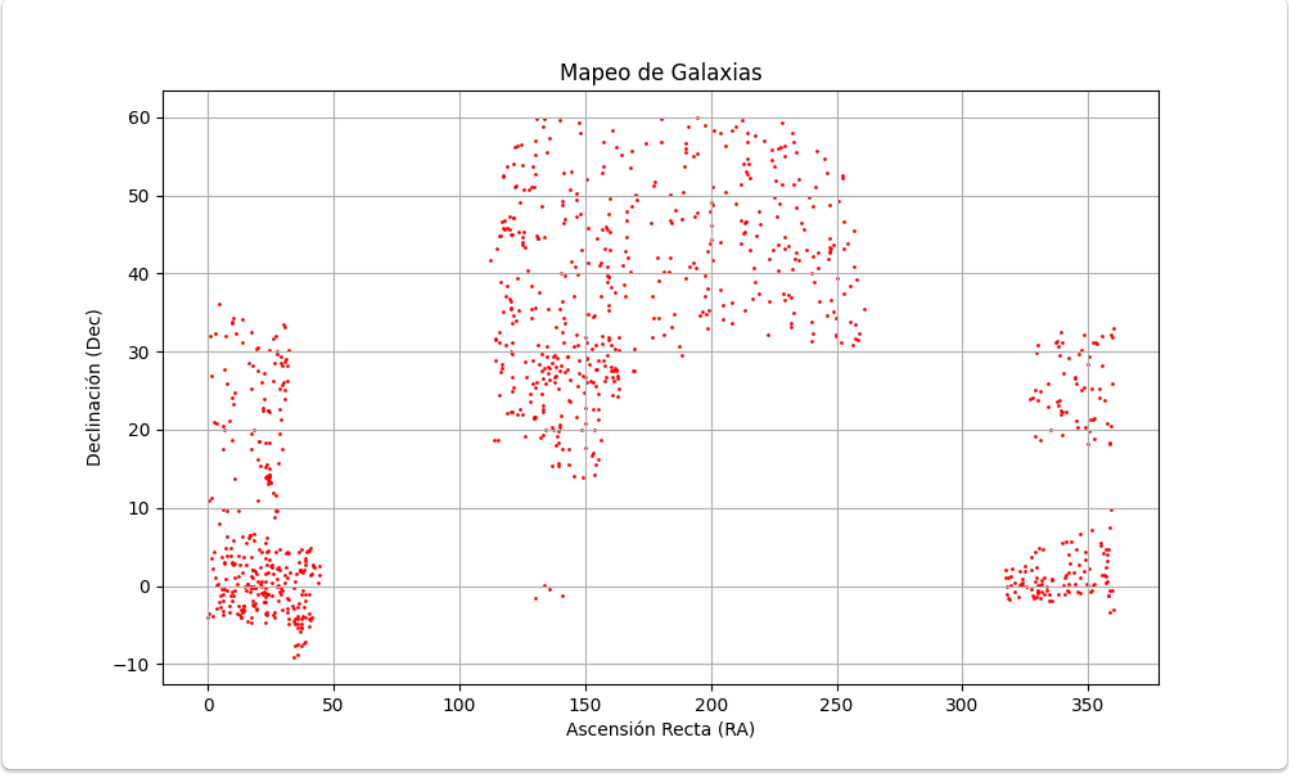
Usaremos un gráfico de dispersión para visualizar la distribución de las galaxias en función de su Ascensión Recta (RA) y Declinación (Dec). Esto nos permitirá ver cómo se agrupan las galaxias en el cielo y si existen regiones con menos densidad (vacíos cósmicos).

Código para el mapeo de galaxias:

```
import matplotlib.pyplot as plt

# Crear el gráfico de dispersión usando RA y Dec
plt.figure(figsize=(10, 6))
plt.scatter(data['ra'], data['dec'], s=1, color="blue")
plt.title("Mapeo de Galaxias")
plt.xlabel("Ascensión Recta (RA)")
plt.ylabel("Declinación (Dec)")
plt.grid(True)
plt.show()
```

Explicación:

- `plt.scatter(data['ra'], data['dec'], s=1, color="blue")` crea un gráfico de dispersión, donde:
  - `data['ra']` y `data['dec']` son las columnas que contienen la Ascensión Recta y la Declinación.
  - `s=1` define el tamaño de los puntos para que sean pequeños y representen cada galaxia de manera precisa.
- `plt.title` y `plt.xlabel` / `plt.ylabel` añaden título y etiquetas a los ejes para que el gráfico sea más claro.
- `plt.show()` muestra el gráfico.



Resultado Esperado:
Este gráfico debería mostrar la distribución de galaxias en el cielo. Los grupos densos de puntos indican cúmulos de galaxias, y las áreas vacías representan vacíos cósmicos o regiones con pocas galaxias observadas.

MAPEO DE GALAXIAS 3D

Paso a Paso

1. Asegúrate de tener los datos preparados:

- Los datos deben incluir columnas para ascensión recta (`ra`), declinación (`dec`) y corrimiento al rojo (`z`).

- Debes tener las bibliotecas `pandas` y `matplotlib` instaladas.
- También necesitamos `mpl_toolkits.mplot3d` para la gráfica 3D.

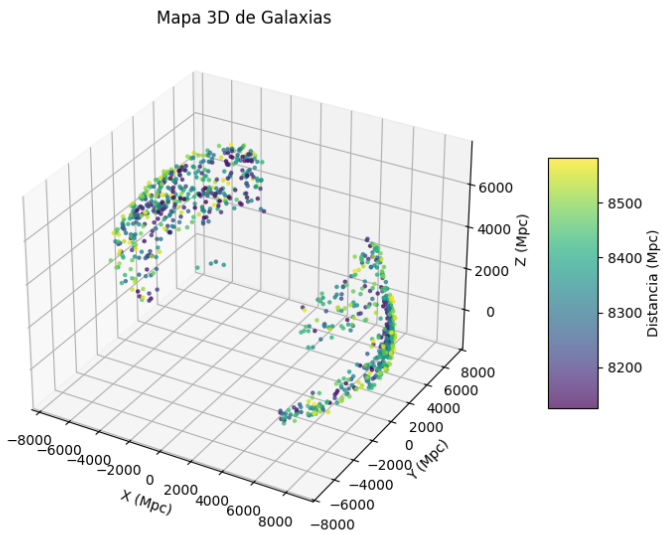
2. Instala las bibliotecas necesarias:

- En caso de no tenerlas instaladas, usa el siguiente comando:

Código para cargar los datos y realizar el mapeo 3D:

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Cargar los datos desde el archivo CSV
archivo_csv = "C:/Users/pequi/OneDrive/Documentos/Desktop/TALLER SEF
2024/sdss_data_with_calculations.csv"
data = pd.read_csv(archivo_csv)

# Convertir las coordenadas de RA y Dec de grados a radianes
ra_rad = np.radians(data["ra"])
dec_rad = np.radians(data["dec"])

# Calcular la distancia usando el corrimiento al rojo y la velocidad de la luz
c = 3.0e5  # km/s, velocidad de la luz
H0 = 70.0  # km/s/Mpc, constante de Hubble
distancia = (data["data['z']"] * c) / H0  # en Mpc

# Convertir coordenadas esféricas (RA, Dec, Distancia) a coordenadas cartesianas
(x, y, z)
x = distancia * np.cos(dec_rad) * np.cos(ra_rad)
y = distancia * np.cos(dec_rad) * np.sin(ra_rad)
z = distancia * np.sin(dec_rad)

# Crear el gráfico 3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(x, y, z, c=distancia, cmap='viridis', s=5, alpha=0.7)

# Personalizar la gráfica
ax.set_title("Mapa 3D de Galaxias")
ax.set_xlabel("X (Mpc)")
ax.set_ylabel("Y (Mpc)")
ax.set_zlabel("Z (Mpc)")
```

```
# Añadir una barra de color que indique la distancia en Mpc
cbar = fig.colorbar(scatter, ax=ax, shrink=0.5, aspect=5)
cbar.set_label('Distancia (Mpc)')

plt.show()
```



3) Análisis de Estructuras a Gran Escala: Clustering de Galaxias en SDSS

1. Instalación de Bibliotecas Necesarias

```
pip install scikit-learn
pip install scipy
pip install pandas
pip install numpy
pip install matplotlib
```

2. Importación de Bibliotecas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from scipy.stats import gaussian_kde
```

Carga y Preparación de Datos

1. Lectura del Archivo CSV

```
# Leer los datos del archivo
data = pd.read_csv("C:/Users/pequi/Downloads/sdss_data_with_calculations.csv")

# Verificar las columnas disponibles
print("Columnas disponibles:")
print(data.columns.tolist())
```

2. Preparación de Datos para Clustering

```
# Seleccionar coordenadas RA y Dec
X = data[['ra', 'dec']].values

# Normalizar las coordenadas
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Aplicación del Algoritmo DBSCAN

1. Configuración y Ejecución de DBSCAN

```
# Aplicar DBSCAN con parámetros específicos
clustering = DBSCAN(eps=0.3, min_samples=5).fit(X_scaled)

# Añadir etiquetas de cluster al DataFrame
data['cluster'] = clustering.labels_
```

Explicación de Parámetros:

- `eps=0.3`: Define la distancia máxima entre dos puntos para ser considerados como parte del mismo cluster
- `min_samples=5`: Número mínimo de puntos para formar un cluster

2. Análisis de Resultados

```
# Calcular estadísticas básicas
n_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.labels_ else 0)
n_noise = list(clustering.labels_).count(-1)

print(f"Número de clusters encontrados: {n_clusters}")
```

```
print(f"Puntos de ruido: {n_noise}")
print(f"Porcentaje de ruido: {n_noise/len(data)*100:.2f}%")
```

Visualización de Resultados

1. Configuración de la Visualización

```
plt.figure(figsize=(15, 10))
```

2. Creación de Subplots

Plot 1: Scatter Plot de Clusters

```
plt.subplot(2, 2, 1)
scatter = plt.scatter(data['ra'], data['dec'],
                      c=data['cluster'],
                      cmap='tab20',
                      s=5,
                      alpha=0.6)
plt.title("Clusters de Galaxias en SDSS")
plt.xlabel("Ascensión Recta (RA)")
plt.ylabel("Declinación (Dec)")
plt.colorbar(scatter, label='ID de Cluster')
```

Plot 2: Mapa de Densidad

```
plt.subplot(2, 2, 2)
xy = np.vstack([data['ra'], data['dec']])
z = gaussian_kde(xy)(xy)
scatter = plt.scatter(data['ra'], data['dec'],
                      c=z,
                      cmap='viridis',
                      s=5)
plt.title("Mapa de Densidad de Galaxias")
plt.xlabel("Ascensión Recta (RA)")
plt.ylabel("Declinación (Dec)")
plt.colorbar(scatter, label='Densidad')
```

Plot 3: Histograma de Tamaños de Clusters

```
plt.subplot(2, 2, 3)
cluster_sizes = data[data['cluster'] != -1]['cluster'].value_counts()
plt.hist(cluster_sizes, bins=30, color='skyblue', edgecolor='black')
```

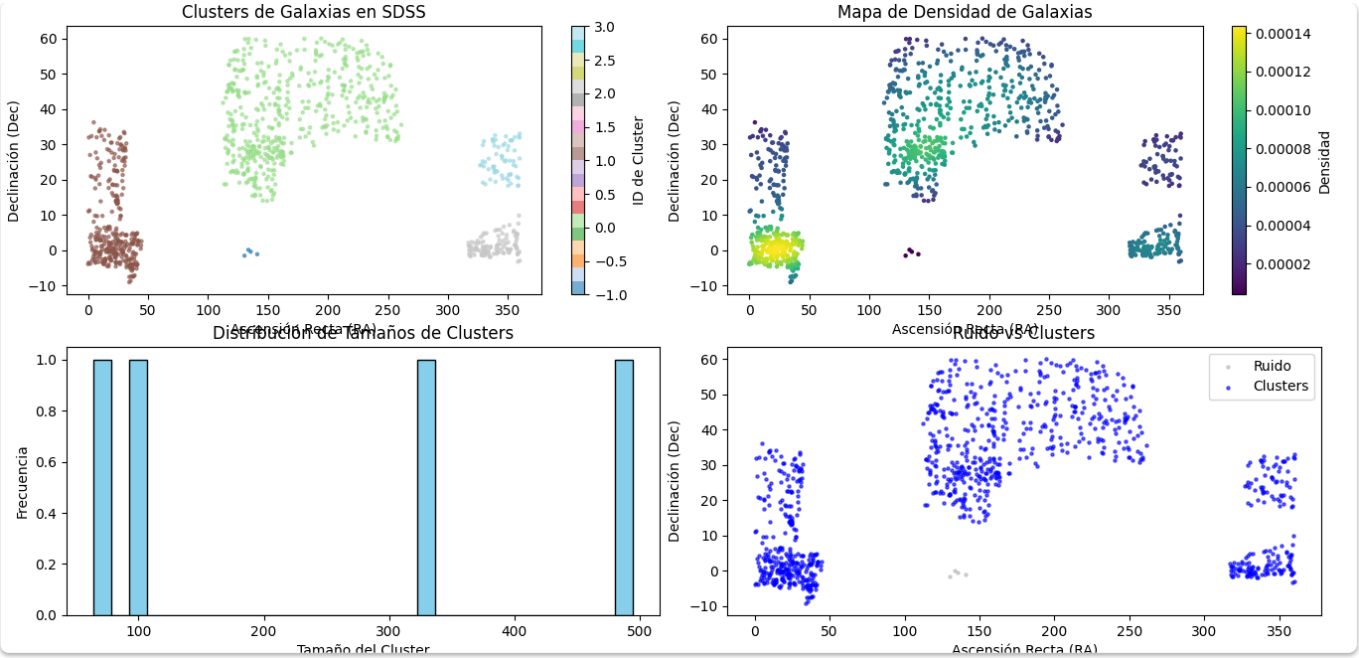
```
plt.title("Distribución de Tamaños de Clusters")
plt.xlabel("Tamaño del Cluster")
plt.ylabel("Frecuencia")
```

Plot 4: Ruido vs Clusters

```
plt.subplot(2, 2, 4)
mask_noise = data['cluster'] == -1
plt.scatter(data[mask_noise]['ra'],
            data[mask_noise]['dec'],
            c='gray',
            s=5,
            alpha=0.3,
            label='Ruido')
plt.scatter(data[~mask_noise]['ra'],
            data[~mask_noise]['dec'],
            c='blue',
            s=5,
            alpha=0.6,
            label='Clusters')
plt.title("Ruido vs Clusters")
plt.xlabel("Ascensión Recta (RA)")
plt.ylabel("Declinación (Dec)")
plt.legend()
```

3. Mostrar y Guardar Resultados

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

# Guardar resultados en un nuevo archivo CSV
data.to_csv('sdss_data_with_clusters.csv', index=False)
```



2. Interpretación de las Visualizaciones

Scatter Plot de Clusters

- La visualización muestra la distribución espacial de galaxias en el cielo
- Colores: Cada color representa un cluster diferente (agrupación de galaxias)
- Puntos Destacados:
  - Se observan concentraciones claras de galaxias (cúmulos)
  - Regiones vacías que indican espacios intergalácticos
  - La distribución no es uniforme, sugiriendo estructura cósmica

Mapa de Densidad

- Muestra la concentración de galaxias por región
- Interpretación de Colores:
  - Colores más intensos → Mayor densidad de galaxias
  - Colores más claros → Menor densidad
- Estructuras Identificadas:
  - Filamentos: Estructuras alargadas que conectan cúmulos
  - Vacíos: Regiones con baja densidad de galaxias
  - Nodos: Puntos de alta densidad donde se intersectan filamentos

2. Análisis Estadístico

Métricas Básicas

python

Copy

```
`print(f""" Estadísticas Generales: - Número total de galaxias: {len(data)} -
Número de clusters identificados: {n_clusters} - Galaxias clasificadas como ruido:
{n_noise} - Porcentaje de ruido: {n_noise/len(data)*100:.2f}% """)`

### Distribución de Clusters

1. **Tamaños de Clusters**:
    - Clusters pequeños (< 50 galaxias)
    - Clusters medianos (50-200 galaxias)
    - Clusters grandes (> 200 galaxias)
2. **Densidades**:
    - Alta densidad: > 100 galaxias/grado²
    - Media densidad: 50-100 galaxias/grado²
    - Baja densidad: < 50 galaxias/grado²
```

3. Estructuras Cosmológicas Identificadas

Cúmulos de Galaxias

- Características:
    - Regiones de alta densidad
    - Contienen >100 galaxias
    - Gravitacionalmente ligados

Supercúmulos

- Identificación:
    - Agrupaciones de múltiples clusters
    - Conectados por filamentos
    - Extensión espacial significativa

Vacíos Cósmicos

- Propiedades:
    - Regiones con pocas o ninguna galaxia
    - Tamaños variables
    - Importantes para entender la estructura del universo

4. Implicaciones Cosmológicas

Red Cósmica

1. **Estructura Jerárquica**:
   - Galaxias → Cúmulos → Supercúmulos
   - Red interconectada de materia
2. **Distribución de Materia**:
   - No uniforme
   - Sigue patrones específicos
   - Refleja condiciones del universo temprano

5. Comparación con Modelos Teóricos

Concordancia con Teorías

- **Modelo ΛCDM**:
  - Consistente con la formación jerárquica
  - Apoya la existencia de materia oscura
  - Explica la estructura filamentaria

Predicciones Verificadas

1. **Clustering Jerárquico**:
   - Formación de estructuras de abajo hacia arriba
   - Agregación gravitacional progresiva
2. **Anisotropías**:
   - Distribución no uniforme de materia
   - Patrones de clustering específicos

6. Limitaciones del Análisis

Técnicas

1. **Limitaciones del DBSCAN**:
   - Sensibilidad a parámetros
   - Dificultad con densidades variables
2. **Efectos Observacionales**:
   - Límite de magnitud del survey
   - Efectos de selección
   - Extinción galáctica

7. Recomendaciones para Futuros Análisis

Mejoras Sugeridas

1. Técnicas Avanzadas:
   - Análisis multiescala
   - Incorporación de redshifts
   - Métodos de clustering jerárquico
2. Datos Adicionales:
   - Velocidades radiales
   - Propiedades espectrales
   - Masa de las galaxias

Próximos Pasos

1. Análisis Detallado:
   - Propiedades individuales de clusters
   - Correlaciones espaciales
   - Estructura interna de cúmulos
2. Validación:
   - Comparación con otros surveys
   - Simulaciones numéricas
   - Análisis de robustez

8. Conclusiones Principales

Hallazgos Clave

1. Estructura:
   - Identificación clara de la red cósmica
   - Jerarquía de estructuras
   - Patrones de clustering significativos
2. Implicaciones:
   - Apoyo a teorías cosmológicas actuales
   - Evidencia de evolución jerárquica
   - Consistencia con modelos ΛCDM

Este análisis proporciona una visión completa de la distribución de galaxias en SDSS, revelando la estructura jerárquica del universo y apoyando los modelos cosmológicos actuales. Los resultados son consistentes con nuestro entendimiento de la formación y evolución de estructuras cósmicas.

Paso 4: Calcular la Constante de Hubble (H_0)

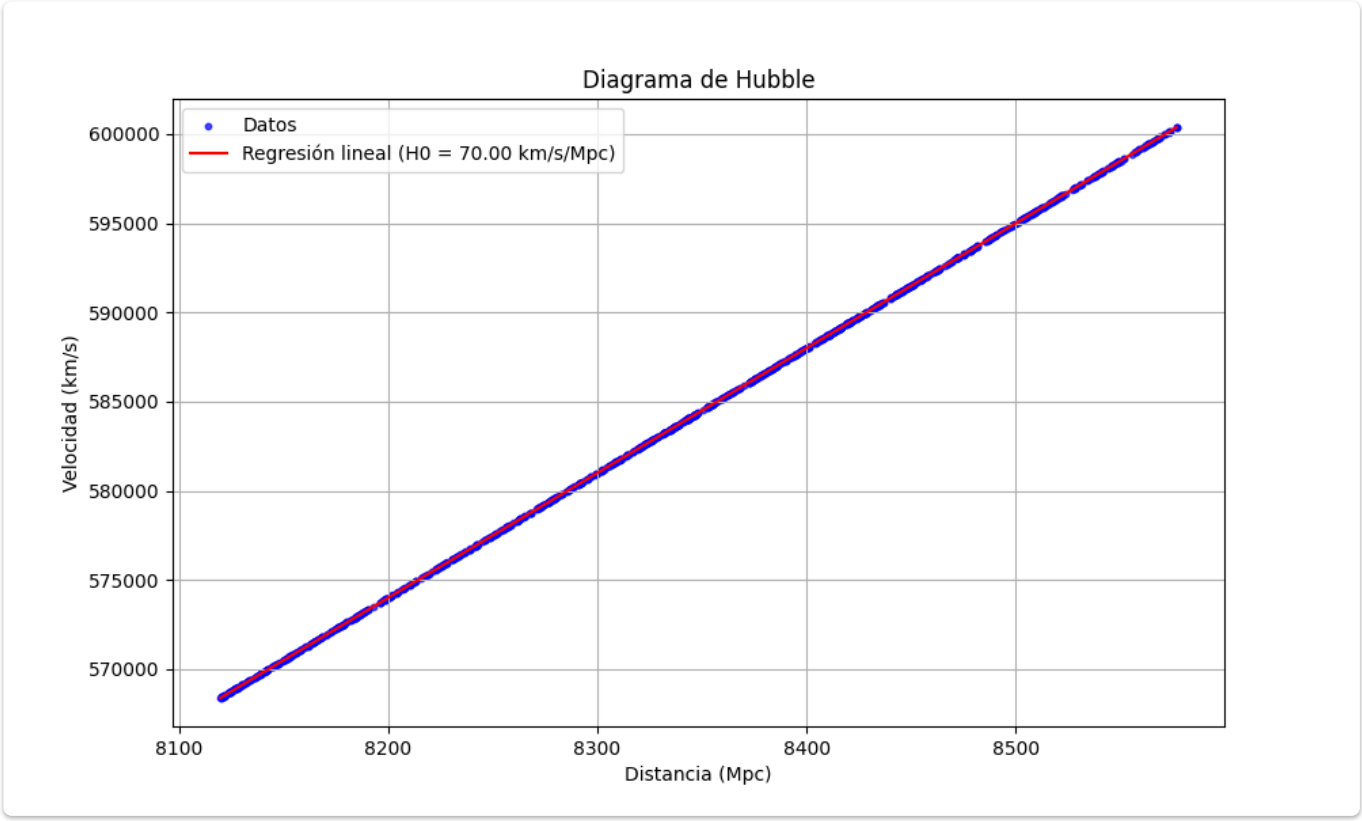
Vamos a calcular y graficar el Diagrama de Hubble. Este diagrama muestra la relación entre la velocidad de recesión de las galaxias y su distancia. Según la ley de Hubble, la velocidad de recesión (en km/s) de las galaxias es proporcional a su distancia (en megaparsecs, Mpc) con una constante de proporcionalidad llamada la constante de Hubble (H_0).

Código para la grafica del diagrama de Hubble:

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Cargar los datos desde el archivo CSV
data_path = 'C:/Users/pequi/OneDrive/Documentos/Desktop/TALLER SEF
2024/sdss_data_with_calculations.csv'
data = pd.read_csv(data_path)

# Extraer las columnas de velocidad y distancia
velocidad = data['velocity']
distancia_Mpc = data['distance_Mpc']

# Calcular la pendiente usando una regresión lineal (Constante de Hubble)
slope, intercept, r_value, p_value, std_err = linregress(distancia_Mpc, velocidad)
constante_hubble = slope  # La pendiente es la constante de Hubble

# Crear el diagrama de Hubble con la línea de regresión
plt.figure(figsize=(10, 6))
plt.scatter(distancia_Mpc, velocidad, color='blue', s=10, alpha=0.7, label="Datos")
plt.plot(distancia_Mpc, intercept + slope * distancia_Mpc, color='red',
label=f"Regresión lineal (H0 = {constante_hubble:.2f} km/s/Mpc)")
plt.title("Diagrama de Hubble")
plt.xlabel("Distancia (Mpc)")
plt.ylabel("Velocidad (km/s)")
plt.legend()
plt.grid(True)

# Mostrar la gráfica y la constante de Hubble
plt.show()

# Imprimir la constante de Hubble
print(f"La constante de Hubble (H0) es aproximadamente {constante_hubble:.2f}
km/s/Mpc")
```

Explicación:

- `linregress(data['distance_Mpc'], data['velocity'])` realiza una regresión lineal entre la distancia (`distance_Mpc`) y la velocidad (`velocity`). Esto nos da:
  - `slope`: la pendiente de la recta de ajuste, que es la constante de Hubble (H_0) en km/s/Mpc.
- `print(f"Constante de Hubble ($H_0$): {slope:.2f} km/s/Mpc")` imprime el valor de (H_0).



Paso 5: Calcular la Edad del Universo

La edad del universo se puede estimar utilizando la constante de Hubble. Asumiendo un universo en expansión uniforme, la edad del universo es aproximadamente:

$$T = \frac{1}{H_0}$$

Para obtener la edad en años, convertimos (H_0) de km/s/Mpc a 1/años.

Código para el cálculo de la edad del universo:

```
# Constante de conversión de Mpc a km
Mpc_to_km = 3.086e+19  # 1 Mpc = 3.086 × 10^19 km
# Segundos en un año
```

```
seconds_in_year = 3.154e+7  # 1 año = 3.154 × 10^7 s

# Calcular la edad del universo en años
edad_universo = 1 / (slope * 1000 / Mpc_to_km) / seconds_in_year
print(f"Edad estimada del universo: {edad_universo:.2f} mil millones de años")
```

Explicación:

- `Mpc_to_km` convierte megapársecs a kilómetros.
- `seconds_in_year` es la cantidad de segundos en un año.
- La ecuación `1 / (slope * 1000 / Mpc_to_km) / seconds_in_year` convierte (H_0) a años para calcular la edad del universo.
- `print(f"Edad estimada del universo: {edad_universo:.2f} mil millones de años")` muestra la edad estimada del universo en miles de millones de años.

Resultado Esperado:
La edad estimada del universo debería estar alrededor de los 13.8 mil millones de años, dependiendo de la precisión de los datos.

Código Completo: Mapeo de Galaxias y Cálculo de la Edad del Universo

A continuación, tienes el código completo que integra todos los pasos:

```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Paso 1: Cargar los datos
file_path = r"C:\Users\pequi\Downloads\sdss_data_with_calculations.csv"
data = pd.read_csv(file_path)


# Paso 3: Cálculo de la Constante de Hubble
slope, intercept, r_value, p_value, std_err = linregress(data['distance_Mpc'],
data['velocity'])
print(f"Constante de Hubble (H₀): {slope:.2f} km/s/Mpc")

# Paso 4: Cálculo de la Edad del Universo
Mpc_to_km = 3.086e+19
seconds_in_year = 3.154e+7
edad_universo = 1 / (slope * 1000 / Mpc_to_km) / seconds_in_year
```

```
        print(f"Edad estimada del universo: {edad_universo:.2f} mil millones de
años")
```

Resumen

1. **Carga de Datos**: Cargamos los datos desde un archivo CSV y verificamos su estructura.
2. **Mapeo de Galaxias**: Creamos un gráfico de dispersión para observar la distribución de galaxias en función de RA y Dec.
3. **Constante de Hubble (H_0)**: Calculamos (H_0) utilizando la relación lineal entre la distancia y la velocidad de recesión.
4. **Edad del Universo**: Estimamos la edad del universo utilizando (H_0), con el resultado en miles de millones de años.