```
# STEP 1: Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
# Imports
import os
import pandas as pd
from astropy.io import fits
from astropy.cosmology import Planck18 as cosmo
import astropy.units as u

# Set redshift
z = 0.016268

# Root folder containing subfolders
root_path = '/content/drive/MyDrive/JWST MIRI NGC 7469/'

# Find all FITS files recursively
fits_files = []
for dirpath, dirnames, filenames in os.walk(root_path):
    for file in filenames:
        if file.endswith('s3d.fits'):
            fits_files.append(os.path.join(dirpath, file))

print(f"Found {len(fits_files)} FITS files.")

# Libraries
import os
import pandas as pd
from astropy.io import fits
import astropy.units as u

# Set path to JWST MIRI data in your Google Drive
root_path = "/content/drive/MyDrive/JWST MIRI NGC 7469"

# Distance to NGC 7469
distance_mpc = 70 * u.Mpc

# Conversion constant from arcsec to radians
rad_per_arcsec = u.arcsec.to(u.rad)
```

```
# Store results
pixel_scales = []

# Recursively look for *_s3d.fits files
for dirpath, _, filenames in os.walk(root_path):
    for filename in filenames:
        if filename.endswith('_s3d.fits'):
            filepath = os.path.join(dirpath, filename)
            try:
                with fits.open(filepath) as hdul:
                    header = hdul[1].header  # SCI extension

                    cdelt1 = header.get('CDELT1')  # degrees/pixel
                    cdelt2 = header.get('CDELT2')
                    print(f"CDELT1: {cdelt1}, CDELT2: {cdelt2}")

                    if cdelt1 is None or cdelt2 is None:
                        print(f"CDELT not found in {filename}")
                        continue

                    # Convert to arcsec
                    arcsec_x = abs(cdelt1) * 3600
                    arcsec_y = abs(cdelt2) * 3600

                    # Convert to parsecs
                    pc_per_pixel_x = (arcsec_x * rad_per_arcsec *
distance_mpc).to(u.pc).value
                    pc_per_pixel_y = (arcsec_y * rad_per_arcsec *
distance_mpc).to(u.pc).value

                    pixel_scales.append({
                        "FITS File": filename,
                        "Arcsec/Pixel X": round(arcsec_x, 4),
                        "Arcsec/Pixel Y": round(arcsec_y, 4),
                        "Parsec/Pixel X": round(pc_per_pixel_x, 4),
                        "Parsec/Pixel Y": round(pc_per_pixel_y, 4)
                    })
            except Exception as e:
                print(f"Error reading {filename}: {e}")
```

```
# Display results
df = pd.DataFrame(pixel_scales)
df.sort_values("FITS File", inplace=True)
df.reset_index(drop=True, inplace=True)
df
```

CDELT1: 5.55555563833978e-05, CDELT2: 5.55555563833978e-05
CDELT1: 9.72222205665376e-05, CDELT2: 9.72222205665376e-05
CDELT1: 5.55555563833978e-05, CDELT2: 5.55555563833978e-05
CDELT1: 4.72222227189275e-05, CDELT2: 4.72222227189275e-05
CDELT1: 4.72222227189275e-05, CDELT2: 4.72222227189275e-05
CDELT1: 5.55555563833978e-05, CDELT2: 5.55555563833978e-05
CDELT1: 9.72222205665376e-05, CDELT2: 9.72222205665376e-05
CDELT1: 3.61111097865634e-05, CDELT2: 3.61111097865634e-05
CDELT1: 9.72222205665376e-05, CDELT2: 9.72222205665376e-05
CDELT1: 4.72222227189275e-05, CDELT2: 4.72222227189275e-05
CDELT1: 3.61111097865634e-05, CDELT2: 3.61111097865634e-05
CDELT1: 3.61111097865634e-05, CDELT2: 3.61111097865634e-05

	FITS File	Arcsec/Pixel X	Arcsec/Pixel Y	Parsec/Pixel X	Parsec/Pixel Y
0	jw01328-c1006_t014_miri_ch1-long_s3d.fits	0.13	0.13	44.1180	44.1180
1	jw01328-c1006_t014_miri_ch1-medium_s3d.fits	0.13	0.13	44.1180	44.1180
2	jw01328-c1006_t014_miri_ch1-short_s3d.fits	0.13	0.13	44.1180	44.1180
3	jw01328-c1006_t014_miri_ch2-long_s3d.fits	0.17	0.17	57.6928	57.6928
4	jw01328-c1006_t014_miri_ch2-medium_s3d.fits	0.17	0.17	57.6928	57.6928
5	jw01328-c1006_t014_miri_ch2-short_s3d.fits	0.17	0.17	57.6928	57.6928

6	jw01328-c1006_t014_miri_ch3-long_s3d.fits	0.20	0.20	67.8739	67.8739
7	jw01328-c1006_t014_miri_ch3-medium_s3d.fits	0.20	0.20	67.8739	67.8739
8	jw01328-c1006_t014_miri_ch3-short_s3d.fits	0.20	0.20	67.8739	67.8739
9	jw01328-c1006_t014_miri_ch4-long_s3d.fits	0.35	0.35	118.7793	118.7793
10	jw01328-c1006_t014_miri_ch4-medium_s3d.fits	0.35	0.35	118.7793	118.7793
11	jw01328-c1006_t014_miri_ch4-short_s3d.fits	0.35	0.35	118.7793	118.7793

```
#Region-wise line plotting
!pip install regions --quiet

# 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)
from regions import Regions
from io import BytesIO

# Load your uploaded file path
reg_path = "/content/drive/MyDrive/JWST MIRI NGC 7469/region_ds9.reg"

# Step 1: Read and decode the file
with open(reg_path, 'r', encoding='utf-8', errors='ignore') as f:
```

```
    content = f.read()

# Step 2: Wrap it in BytesIO so Regions can parse it
region_bytes = BytesIO(content.encode('utf-8'))

# Step 3: Parse regions
regions = Regions.read(region_bytes, format='ds9')

# Step 4: Confirm loaded regions
for i, region in enumerate(regions):
    print(f"Region {i+1}: {region}")
```

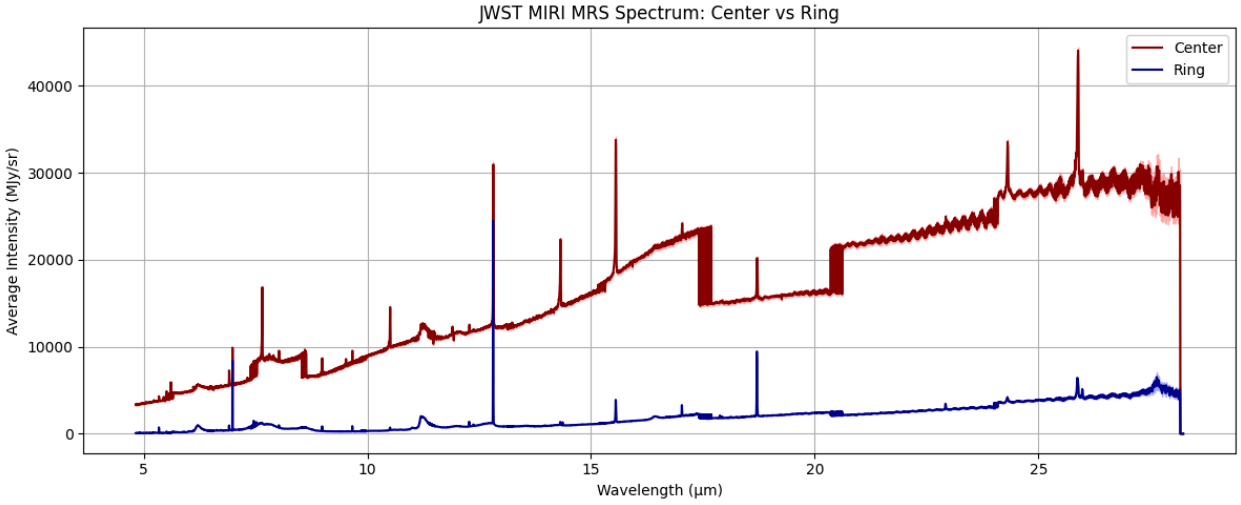
```
Region 1: Region: CircleSkyRegion
center: <SkyCoord (FK5: equinox=J2000.000): (ra, dec) in deg
    (345.8151044, 8.8739304)>
radius: 0.5 arcsec
Region 2: Region: CircleSkyRegion
center: <SkyCoord (FK5: equinox=J2000.000): (ra, dec) in deg
    (345.8154286, 8.8737281)>
radius: 0.5 arcsec
```

```
import numpy as np
import pandas as pd
import os
from astropy.io import fits
from astropy.wcs import WCS
from regions import Regions
from io import BytesIO
import matplotlib.pyplot as plt

# --- PARAMETERS ---
base_path = "/content/drive/MyDrive/JWST MIRI NGC 7469"
region_path = f"{base_path}/region_ds9.reg"
z = 0.0164  # Redshift of NGC 7469

# --- LOAD DS9 REGION FILE ---
from regions import Regions
from io import BytesIO

# Path to your region file in Google Drive
```

```
reg_path = "/content/drive/MyDrive/JWST MIRI NGC 7469/region_ds9.reg"

# Read binary and decode using fallback encoding
with open(reg_path, 'rb') as f:
    content = f.read().decode('latin1')  # safer for non-UTF-8 characters

# Encode back to bytes and wrap in BytesIO
region_bytes = BytesIO(content.encode('utf-8'))  # Regions.read wants a
bytes-like object

# Read the regions
for i, region in enumerate(regions):
    print(f"Region {i+1}: {region}")



assert len(regions) >= 2, "Need at least two regions defined: center and
ring."

# --- DEFINE FILE PATHS ---
file_paths = []
for ch in range(1, 5):
    for part in ['short', 'medium', 'long']:
        fname = f"jw01328-c1006_t014_miri_ch{ch}-{part}_s3d.fits"
        fpath =
f"{base_path}/jw01328-c1006_t014_miri_ch{ch}-{part}/{fname}"
        file_paths.append(fpath)

# --- FUNCTION TO EXTRACT SPECTRUM FROM ONE REGION ---
def extract_region_spectrum(region):
    spectrum_all = []
    spectrum_all_err = []
    wavelength_all = []

    for file_path in file_paths:
        with fits.open(file_path) as hdul:
            data = hdul[1].data.astype(float)
            data[data < 0] = np.nan
            data_err = hdul[2].data.astype(float)
            header = hdul[1].header
            wcs = WCS(header)
```

```
        mask = region.to_pixel(wcs.celestial).to_mask()
        num_channels, ny, nx = data.shape

        spectrum = []
        spectrum_err = []

        for i in range(num_channels):
            masked_data = mask.multiply(data[i, :, :])
            masked_data_err = mask.multiply(data_err[i, :, :])

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

            spectrum.append(avg_intensity if not np.isnan(avg_intensity)
else 0)
            spectrum_err.append(avg_intensity_err if not
np.isnan(avg_intensity_err) else 0)

        crval3 = header['CRVAL3']
        cdelt3 = header['CDELT3']
        crpix3 = header['CRPIX3']
        wavelength = (np.arange(num_channels) - (crpix3 - 1)) * cdelt3 +
crval3
        wavelength /= (1 + z)

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

    df = pd.DataFrame({
        'Wavelength (μm)': wavelength_all,
        'Flux': spectrum_all,
        'Error': spectrum_all_err
    })
    df.sort_values('Wavelength (μm)', inplace=True)
    df.reset_index(drop=True, inplace=True)
    return df

# --- EXTRACT FOR BOTH REGIONS ---
```

```
df_center = extract_region_spectrum(regions[0])
df_ring = extract_region_spectrum(regions[1])

# --- PLOT BOTH SPECTRA ---
plt.figure(figsize=(12, 5))
plt.plot(df_center['Wavelength (μm)'], df_center['Flux'], label='Center',
color='darkred')
plt.fill_between(df_center['Wavelength (μm)'],
                 df_center['Flux'] - df_center['Error'],
                 df_center['Flux'] + df_center['Error'],
                 alpha=0.3, color='red')

plt.plot(df_ring['Wavelength (μm)'], df_ring['Flux'], label='Ring',
color='darkblue')
plt.fill_between(df_ring['Wavelength (μm)'],
                 df_ring['Flux'] - df_ring['Error'],
                 df_ring['Flux'] + df_ring['Error'],
                 alpha=0.3, color='blue')

plt.xlabel("Wavelength (μm)")
plt.ylabel("Average Intensity (MJy/sr)")
plt.title("JWST MIRI MRS Spectrum: Center vs Ring")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
!pip install plotly
```

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

# Convert to numpy arrays for plotting
wavelength_center = df_center['Wavelength (μm)'].values
flux_center = df_center['Flux'].values
err_center = df_center['Error'].values

wavelength_ring = df_ring['Wavelength (μm)'].values
flux_ring = df_ring['Flux'].values
err_ring = df_ring['Error'].values

# Initialize figure
fig = go.Figure(layout=dict(
    width=900,
    height=600,
    template='plotly_white'
))

# --- CENTER REGION ---
fig.add_trace(go.Scatter(
    x=wavelength_center,
    y=flux_center,
    mode='lines',
    line=dict(color='darkred', width=1.5),
    name='Center',
    hovertemplate='λ: %{x:.3f} μm<br>Center: %{y:.2f}
MJy/sr<extra></extra>'
))

fig.add_trace(go.Scatter(
    x=np.concatenate([wavelength_center, wavelength_center[::-1]]),
    y=np.concatenate([flux_center + err_center, (flux_center -
err_center)[::-1]]),
    fill='toself',
    fillcolor='rgba(139, 0, 0, 0.2)',
    line=dict(color='rgba(255,255,255,0)'),
    hoverinfo='skip',
    name='Center Uncertainty'
))
```

```
# --- RING REGION ---
fig.add_trace(go.Scatter(
    x=wavelength_ring,
    y=flux_ring,
    mode='lines',
    line=dict(color='darkblue', width=1.5),
    name='Ring',
    hovertemplate='λ: %{x:.3f} µm<br>Ring: %{y:.2f} MJy/sr<extra></extra>'
))

fig.add_trace(go.Scatter(
    x=np.concatenate([wavelength_ring, wavelength_ring[::-1]]),
    y=np.concatenate([flux_ring + err_ring, (flux_ring -
err_ring)[::-1]]),
    fill='toself',
    fillcolor='rgba(0, 0, 139, 0.2)',
    line=dict(color='rgba(255,255,255,0)'),
    hoverinfo='skip',
    name='Ring Uncertainty'
))

# --- FEATURES ---
features = {
    'PAHs': {'PAH 7.7': 7.7, 'PAH 8.6': 8.6, 'PAH 11.3': 11.3},
    'Neon': {'[Ne VI]': 7.65},
    'Other': {'[Ar III]': 8.991, '[S IV]': 10.51},
    'H₂': {'S(3)': 9.66, 'S(4)': 8.03}
}

colors = {
    'PAHs': '#FF7F0E',
    'Neon': '#D62728',
    'Other': '#9467BD',
    'H₂': '#8C564B'
}

# 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[category],
                width=1.5 if category == 'PAHs' else 1,
                dash='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(size=10, color=colors[category]),
                yshift=10 if category == 'PAHs' else 0
            )
        )

# Add shaded vertical bands for PAHs
for wl in [7.7, 8.6, 11.3]:
    fig.add_vrect(
        x0=wl - 0.15, x1=wl + 0.15,
        fillcolor=colors['PAHs'],
        opacity=0.1,
        line_width=0
    )

# --- Layout ---
fig.update_layout(
    title='<b>JWST/MIRI IFU Spectra of NGC 7469: Center vs Ring</b>',
    xaxis_title='<b>Rest-frame Wavelength (μm)</b>',
    yaxis_title='<b>Average Intensity (MJy/sr)</b>',
    hovermode='x unified',
    xaxis=dict(range=[7.5, 11]),  # <- Limit x-axis from ~6.5 to 11 μm
    legend=dict(
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80)
)
```

```
fig.update_layout(
    title='<b>JWST/MIRI IFU Spectra of NGC 7469: Center vs Ring</b>',
    xaxis_title='<b>Rest-frame Wavelength (μm)</b>',
    yaxis_title='<b>Average Intensity (MJy/sr)</b>',
    hovermode='x unified',
    xaxis=dict(range=[5.5, 12]),
    yaxis=dict(range=[0, 20000]),  # <- Set y-axis limit
    legend=dict(
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80)
)
```