Summer School 2025

Astronomy & Astrophysics

Project Report

Prepared By

Student Name:	Swarnava Bhowmick
Institution Name:	The Oxford College Of Science
Institution Roll No:	U03MS24S0327
ISA Admission No:	854941

Projects Name

Identifying Spectral Lines in MIRI JWST Data

<u>Submitted To</u>

Name: Mr. Sahil Sakkarwal

Designation: Program Supervisor

Institution: India Space Academy

In [1]:

```python
import numpy as np
import warnings
import matplotlib.pyplot as plt
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctic

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra

#Now that `all_regions_spectra` contains the concatenated and sorted spectra for


plt.figure(figsize=(18, 10)) # Increased figure size for better visibility

for idx, region_data in enumerate(all_regions_spectra):
    plt.errorbar(region_data['wavelength'], region_data['spectrum'],
                 yerr=region_data['spectrum_err'],
                 label=f'Region {idx+1}',
                 fmt='-', # Connect points with lines
                 capsize=2, # Add caps to error bars
                 alpha=0.7) # Make lines slightly transparent for overlap

plt.xlabel('Wavelength (microns)', fontsize=14)
plt.ylabel('Average Intensity (MJy/sr)', fontsize=14)
plt.title('Spectra for all Regions across MIRI Channels', fontsize=16)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout() # Adjust layout to prevent labels from overlapping
plt.show()
```

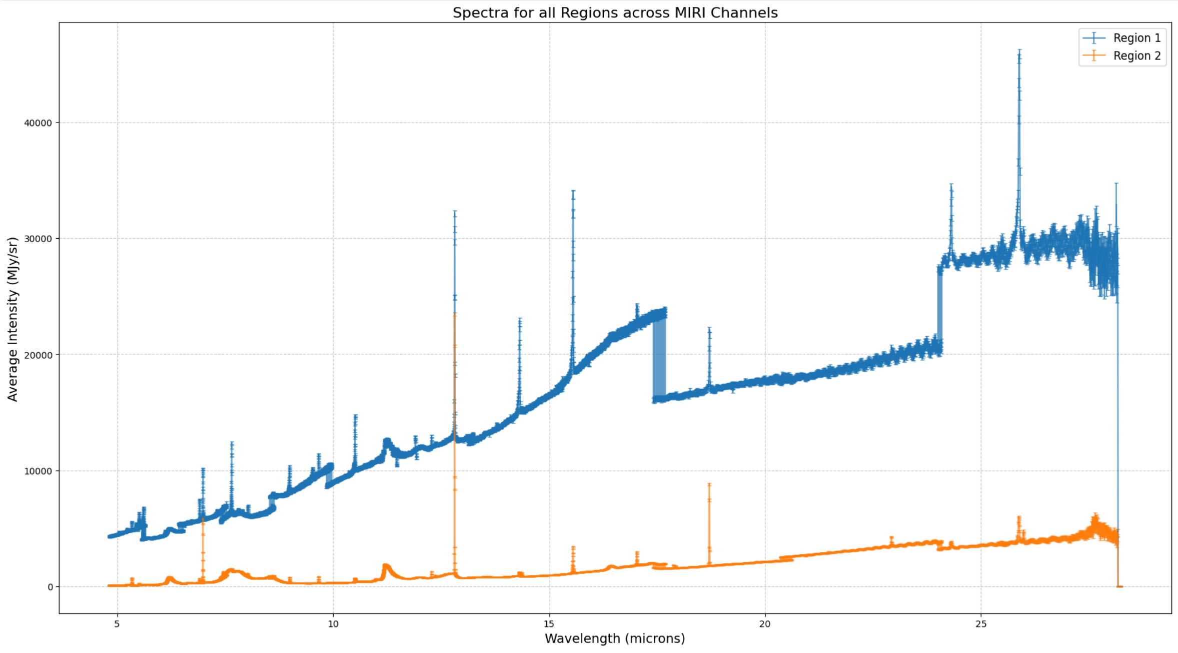
```
Processing Region 1...
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:44.191' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.948' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:02.328' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559129
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445736.634 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:31.550' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:31.595' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:33.047' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557468
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283459 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461184.323 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:24.413' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.737' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.654' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555797
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476718.877 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:44.191' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.948' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:02.328' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559129
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445736.634 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:31.550' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:31.595' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:33.047' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557468
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283459 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461184.323 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:24.413' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.737' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.654' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555797
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476718.877 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:43.551' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.308' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:01.688' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559130
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445726.821 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:30.910' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:30.971' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:32.407' from MJD-END'. [astropy.wcs.wcs]
```

```
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557469
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283458 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461174.508 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:23.837' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.114' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.014' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555798
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476710.042 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:43.551' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.308' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:01.688' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559130
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445726.821 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:30.910' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:30.971' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:32.407' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557469
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283458 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461174.508 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:23.837' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.114' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.014' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555798
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476710.042 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
Processing Region 2...
```

```
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:44.191' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.948' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:02.328' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559129
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445736.634 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:31.550' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:31.595' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:33.047' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557468
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283459 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461184.323 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:24.413' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.737' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.654' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555797
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476718.877 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:44.191' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.948' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:02.328' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559129
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445736.634 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:31.550' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:31.595' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:33.047' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557468
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283459 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461184.323 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:24.413' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.737' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.654' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555797
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476718.877 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:43.551' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.308' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:01.688' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559130
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445726.821 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:30.910' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:30.971' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:32.407' from MJD-END'. [astropy.wcs.wcs]
```

```
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557469
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283458 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461174.508 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:23.837' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.114' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.014' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555798
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476710.042 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
03:48:43.551' from MJD-BEG.
Set DATE-AVG to '2022-07-04T03:54:53.308' from MJD-AVG.
Set DATE-END to '2022-07-04T04:01:01.688' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.559130
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.282938 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737445726.821 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:05:30.910' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:11:30.971' from MJD-AVG.
Set DATE-END to '2022-07-04T04:17:32.407' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.557469
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283458 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737461174.508 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set DATE-BEG to '2022-07-04T
04:22:23.837' from MJD-BEG.
Set DATE-AVG to '2022-07-04T04:28:21.114' from MJD-AVG.
Set DATE-END to '2022-07-04T04:34:17.014' from MJD-END'. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'obsfix' made the change 'Set OBSGEO-L to   -72.555798
from OBSGEO-[XYZ].
Set OBSGEO-B to   -38.283980 from OBSGEO-[XYZ].
Set OBSGEO-H to 1737476710.042 from OBSGEO-[XYZ]'. [astropy.wcs.wcs]
```



In [58]:
```python
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy import units as u
```

```
import pandas as pd
import os

# Directory containing FITS files
fits_dir = "JWST Project/NGC_7469"
# Distance to NGC 7469 (derived from z=0.016268, H0=70 km/s/Mpc)
distance_mpc = 70.0  # Approximate distance
channels = [1, 2, 3, 4]
wavelengths = ['short', 'medium', 'long']
file_paths = [f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch}-{wav}_s3d.f
              for ch in channels for wav in wavelengths]

def calculate_pixel_scale(file_path):
    try:
        # Extract channel and wavelength from filename
        base = os.path.basename(file_path)
        parts = base.split("_")
        ch_wav = parts[3]  # e.g., 'ch1-long'
        ch, wav = ch_wav.split("-")
        channel = int(ch[2:])  # Extract number from 'ch1'
        wavelength = wav

        # Open FITS file and access SCI extension (HDU 1)
        with fits.open(file_path) as hdul:
            header = hdul[1].header

            # Get pixel scale in degrees
            cdelt = header.get('CDELT1', header.get('CDELT2', None))
            if cdelt is None:
                return channel, wavelength, None, None, "No CDELT1/2 found"

            # Convert to arcseconds and parsecs
            pixel_scale_arcsec = abs(cdelt) * 3600
            pixel_scale_rad = pixel_scale_arcsec * u.arcsec.to(u.radian)
            pixel_scale_pc = pixel_scale_rad * (distance_mpc * 1e6)

            return channel, wavelength, pixel_scale_arcsec, pixel_scale_pc, None
    except Exception as e:
        return None, None, None, None, str(e)

# Collect results
results = []
for file_path in file_paths:
    if os.path.exists(file_path):
        channel, wavelength, arcsec, pc, error = calculate_pixel_scale(file_path
        results.append({
            'File': os.path.basename(file_path),
            'Channel': channel,
            'Wavelength': wavelength,
            'Pixel Scale (arcsec)': arcsec if arcsec is not None else 'N/A',
            'Pixel Scale (pc)': pc if pc is not None else 'N/A',
            'Error': error if error else 'Success'
        })
    else:
        base = os.path.basename(file_path)
        parts = base.split("_")
        ch_wav = parts[3]
        ch, wav = ch_wav.split("-")
        results.append({
            'File': base,
            'Channel': int(ch[2:]),
            'Wavelength': wav,
            'Pixel Scale (arcsec)': 'N/A',
            'Pixel Scale (pc)': 'N/A',
            'Error': f"File {base} not found"
        })

# Create DataFrame
df = pd.DataFrame(results)

# Calculate median pixel scale
valid_pc = [r['Pixel Scale (pc)'] for r in results if isinstance(r['Pixel Scale
median_pixel_scale_pc = np.median(valid_pc) if valid_pc else None

# Plot histogram by channel
plt.figure(figsize=(10, 6))
for channel in channels:
    channel_data = [r['Pixel Scale (pc)'] for r in results
                    if r['Channel'] == channel and isinstance(r['Pixel Scale (pc
    if channel_data:
        plt.hist(channel_data, bins=3, alpha=0.5, label=f'Channel {channel}')
if median_pixel_scale_pc:
    plt.axvline(median_pixel_scale_pc, color='red', linestyle='--',
                label=f'Median: {median_pixel_scale_pc:.2f} pc')
plt.xlabel('Pixel Scale (parsecs)')
plt.ylabel('Count')
plt.title('Pixel Scale Distribution for NGC 7469')
plt.legend()
plt.savefig('ngc7469_pixel_scale_histogram.png')
plt.close()

# Save results to CSV
df.to_csv('ngc7469_pixel_scale_results.csv', index=False)

# Interpretation
if median_pixel_scale_pc:
    print(f"Median pixel scale: {median_pixel_scale_pc:.2f} parsecs/pixel")
    if median_pixel_scale_pc < 100:
        region = "nuclear region or star-forming regions (AGN, circumnuclear dis
    elif 100 <= median_pixel_scale_pc < 1000:
        region = "galactic disk or spiral arms"
    else:
        region = "extended halo or large-scale structure"
    print(f"Physical size per pixel: {median_pixel_scale_pc:.2f} parsecs")
    print(f"Likely studying: {region} of NGC 7469")
else:
    print("No valid pixel scales calculated.")

# Display table
print("\nResults Table:")
print(df[['File', 'Channel', 'Wavelength', 'Pixel Scale (arcsec)', 'Pixel Scale
Median pixel scale: 62.78 parsecs/pixel
Physical size per pixel: 62.78 parsecs
Likely studying: nuclear region or star-forming regions (AGN, circumnuclear disk)
of NGC 7469

Results Table:
                                       File  Channel Wavelength  Pixel Scale (arc
sec)  Pixel Scale (pc)   Error
 jw01328-c1006_t014_miri_ch1-short_s3d.fits        1      short
0.13         44.118043 Success
jw01328-c1006_t014_miri_ch1-medium_s3d.fits        1     medium
0.13         44.118043 Success
  jw01328-c1006_t014_miri_ch1-long_s3d.fits        1       long
0.13         44.118043 Success
 jw01328-c1006_t014_miri_ch2-short_s3d.fits        2      short
0.17         57.692829 Success
jw01328-c1006_t014_miri_ch2-medium_s3d.fits        2     medium
0.17         57.692829 Success
  jw01328-c1006_t014_miri_ch2-long_s3d.fits        2       long
0.17         57.692829 Success
 jw01328-c1006_t014_miri_ch3-short_s3d.fits        3      short
0.20         67.873916 Success
jw01328-c1006_t014_miri_ch3-medium_s3d.fits        3     medium
0.20         67.873916 Success
  jw01328-c1006_t014_miri_ch3-long_s3d.fits        3       long
0.20         67.873916 Success
 jw01328-c1006_t014_miri_ch4-short_s3d.fits        4      short
0.35        118.779350 Success
jw01328-c1006_t014_miri_ch4-medium_s3d.fits        4     medium
0.35        118.779350 Success
  jw01328-c1006_t014_miri_ch4-long_s3d.fits        4       long
0.35        118.779350 Success
```

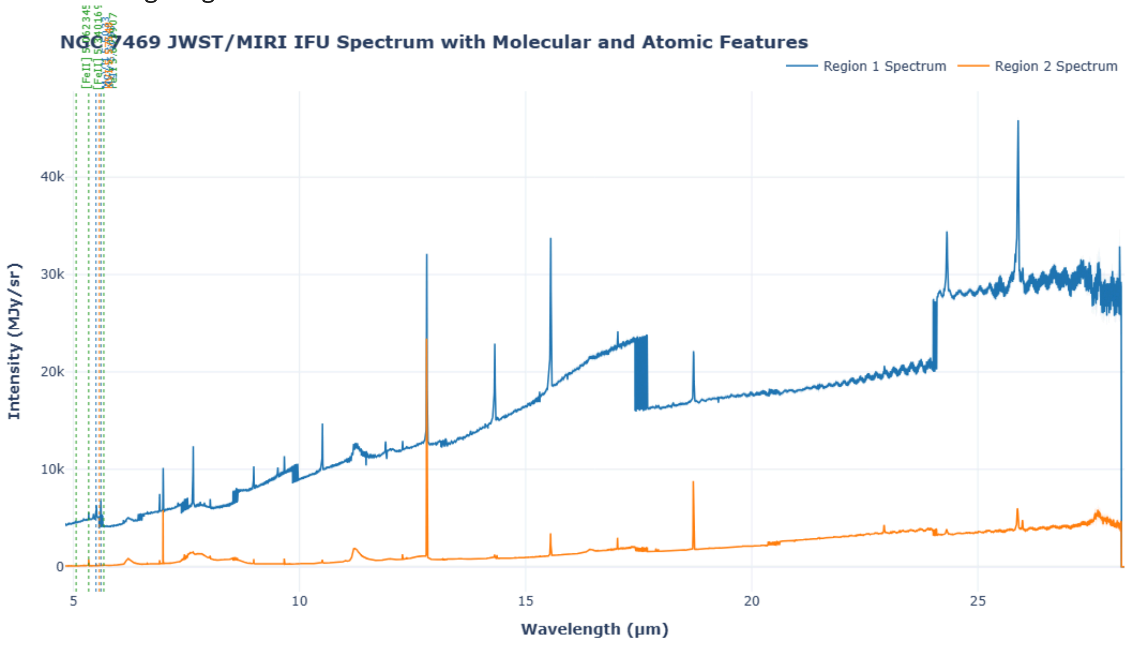
In [42]:
```python
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch1-short_s3d.f

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [3]:
```python
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')
```

```python
# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Iron': {
        '[FeII] 5.062345': 5.062345,
        '[FeII] 5.340169': 5.340169,
        'FeII 5.673907': 5.673907
    },
    'Magnesium': {
        'MgVII 5.5033': 5.5033,
        'MgV 5.6098': 5.6098
    },
    'Potassium': {
        'KVI 5.57548': 5.57548
    }
}

# Define colors for different feature categories
colors_features = {
    'Iron': '#2CA02C',      # Green
    'Magnesium': '#1F77B4', # Blue
    'Potassium': '#FF7F0E'  # Orange
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (µm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



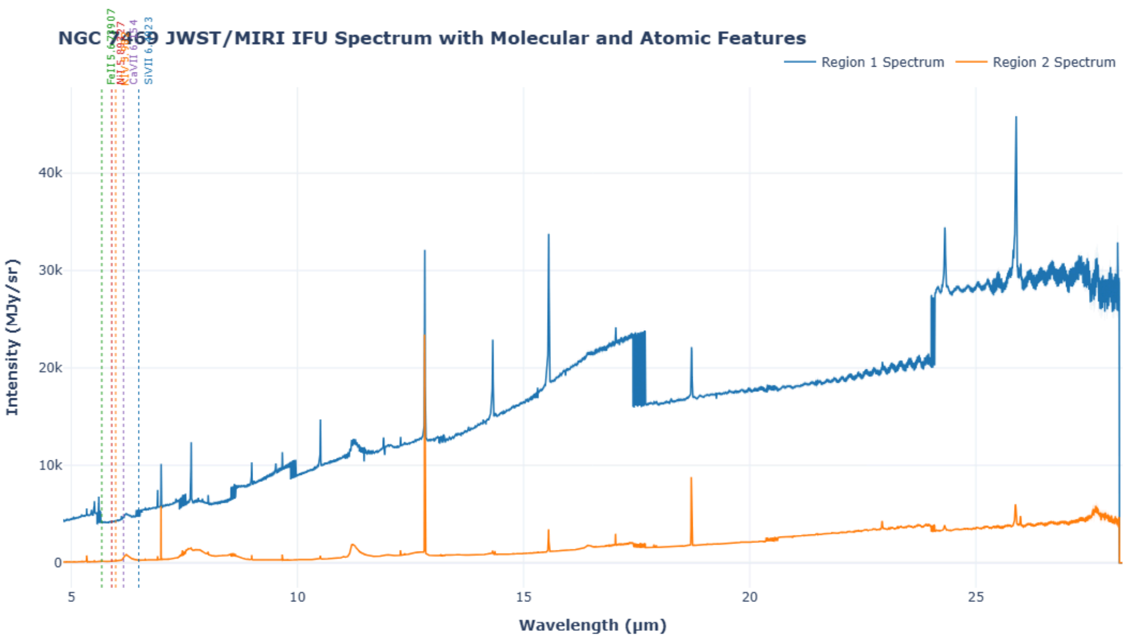
In [44]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch1-medium_s3d.

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [7]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
```

```
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Iron': {
        'FeII 5.673907': 5.673907
    },
    'Nickel': {
        'NiI 5.89327': 5.89327
    },
    'Potassium': {
        'KIV 5.982': 5.982
    },
    'Calcium': {
        'CaVII 6.154': 6.154
    },
    'Silicon': {
        'SiVII 6.4923': 6.4923
    }
}

# Define colors for different feature categories
colors_features = {
    'Iron': '#2CA02C',      # Green
    'Nickel': '#D62728',    # Red
    'Potassium': '#FF7F0E', # Orange
    'Calcium': '#9467BD',   # Purple
    'Silicon': '#1F77B4'    # Blue
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} µm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (µm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```

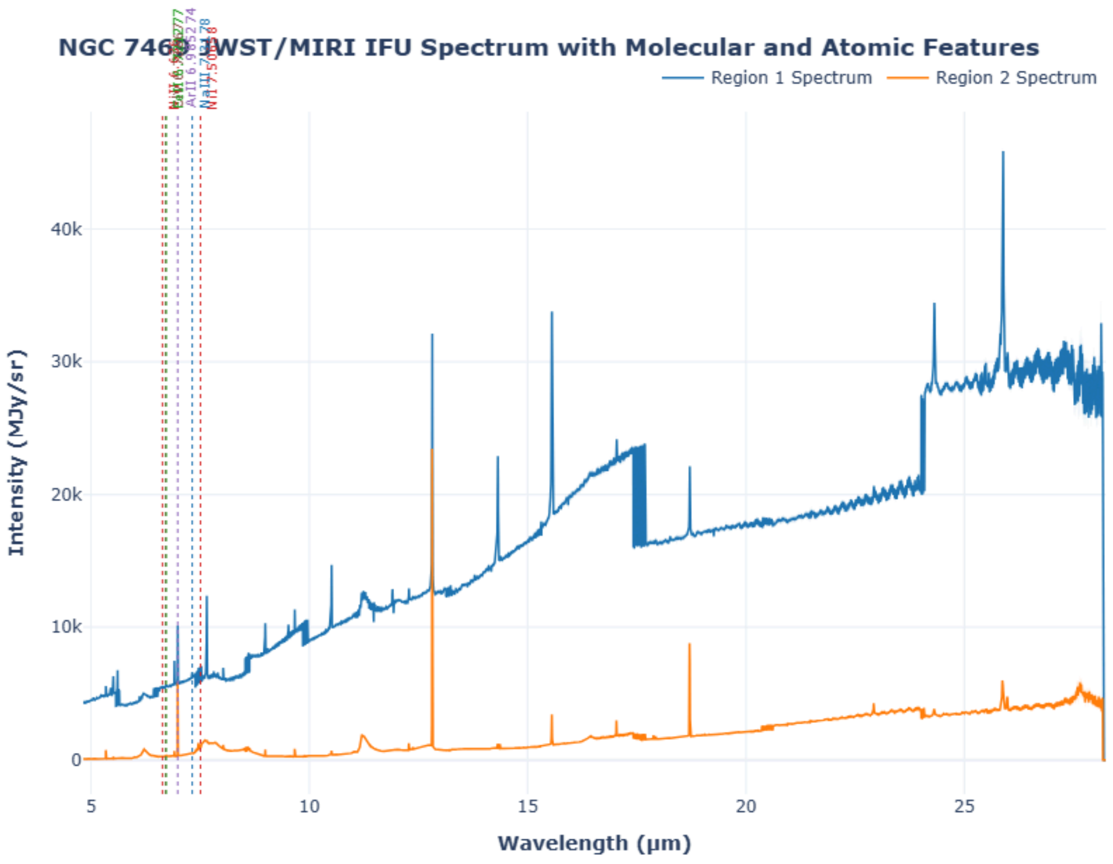


In [11]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch1-long_s3d.fi

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [13]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Nickel': {
        'NiII 6.636': 6.636,
        'NiI 7.50658': 7.50658
    },
    'Chlorine': {
        'ClV 6.70667': 6.70667
    },
    'Iron': {
        'FeII 6.721277': 6.721277
    },
    'Argon': {
        'ArII 6.985274': 6.985274
    },
    'Sodium': {
        'NaIII 7.3178': 7.3178
    }
}

# Define colors for different feature categories
colors_features = {
    'Nickel': '#D62728',    # Red
    'Chlorine': '#8C564B',  # Brown
    'Iron': '#2CA02C',      # Green
    'Argon': '#9467BD',     # Purple
    'Sodium': '#1F77B4'     # Blue
}
# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} µm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (µm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



In [46]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch2-short_s3d.f

cubeviz.show()
```

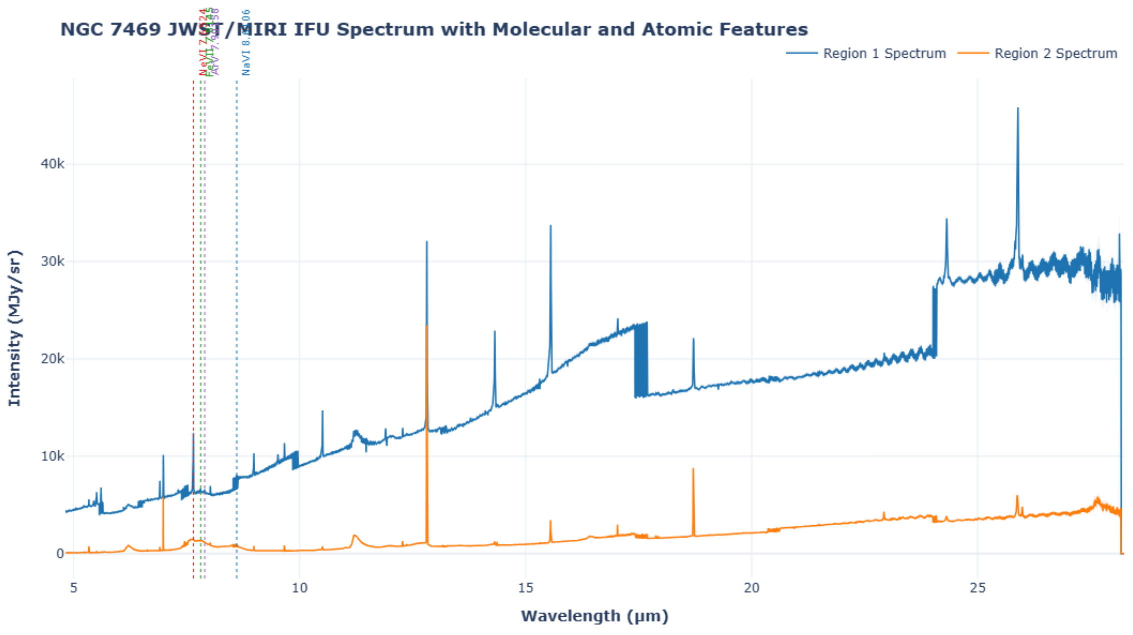
```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [17]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue
                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
```

```
))

# Define spectral features
features = {
    'Neon': {
        'NeVI 7.6524': 7.6524
    },
    'Iron': {
        'FeVII 7.8145': 7.8145
    },
    'Argon': {
        'ArV 7.90158': 7.90158
    },
    'Sodium': {
        'NaVI 8.6106': 8.6106
    }
}

# Define colors for different feature categories
colors_features = {
    'Neon': '#D62728',    # Red
    'Iron': '#2CA02C',    # Green
    'Argon': '#9467BD',   # Purple
    'Sodium': '#1F77B4'   # Blue
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} µm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))
```

```
# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



```
In [48]: from jdaviz import Cubeviz

         cubeviz = Cubeviz()

         cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch2-medium_s3d.

         cubeviz.show()
```

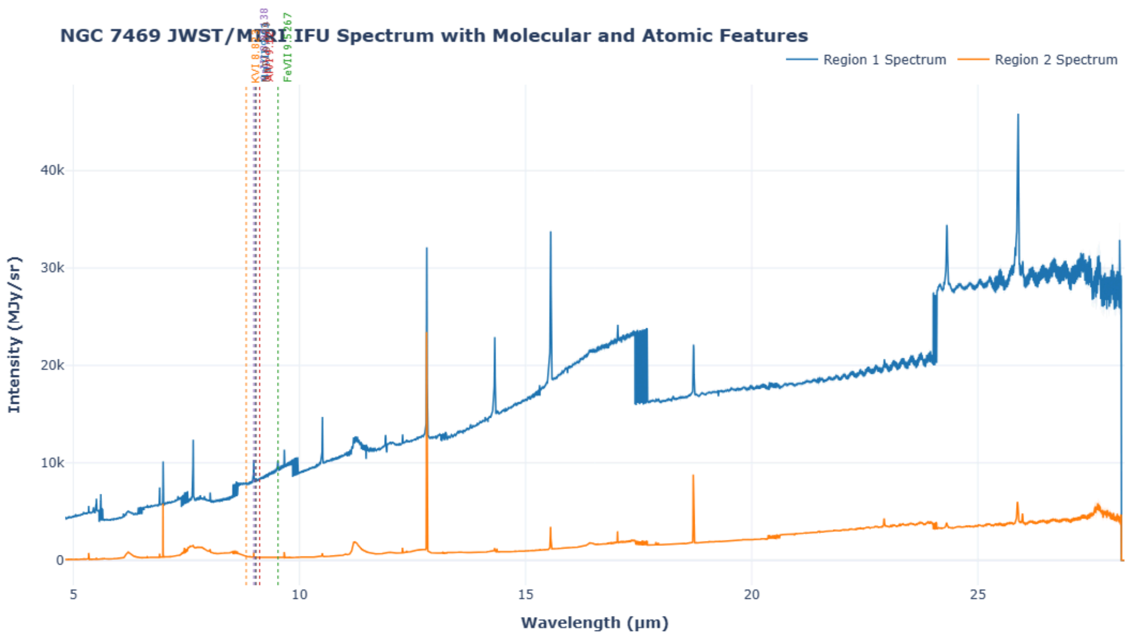
```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

```
In [21]: import numpy as np
         import warnings
         import matplotlib.pyplot as plt # Still imported, but not used for the final plo
         from astropy.wcs import WCS
         from regions import Regions
         from astropy.io import fits
         import plotly.graph_objects as go

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

         z = 0.016268 # Redshift

         # Read DS9 region file:
         reg_path = "JWST Project/NGC_7469/region_2"
         regions = Regions.read(reg_path, format='ds9')

         # Define All Channels and construct file paths:
         channels = [1, 2, 3, 4]  # MIRI channels
         parts = ['short', 'medium', 'long']
         file_paths = []

         for ch_num in channels:
             for part in parts:
                 file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

         # List to store spectra for all regions, each containing data from all channels
         all_regions_spectra = []

         for region_idx, region in enumerate(regions):
```

```
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctic

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")
    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Potassium': {
        'KVI 8.823': 8.823
    },
    'Argon': {
        'ArIII 8.99138': 8.99138
    },
    'Magnesium': {
        'MgVII 9.03': 9.03
    },
    'Sodium': {
        'NaIV 9.039': 9.039
    },
    'Aluminum': {
        'AlVI 9.12': 9.12
    },
    'Iron': {
        'FeVII 9.5267': 9.5267
    }
}

# Define colors for different feature categories
colors_features = {
    'Potassium': '#FF7F0E',  # Orange
    'Argon': '#9467BD',      # Purple
    'Magnesium': '#1F77B4',  # Blue
    'Sodium': '#8C564B',     # Brown
    'Aluminum': '#D62728',   # Red
    'Iron': '#2CA02C'        # Green
}


# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b
```

```
for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



In [50]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch2-long_s3d.fi

cubeviz.show()
```

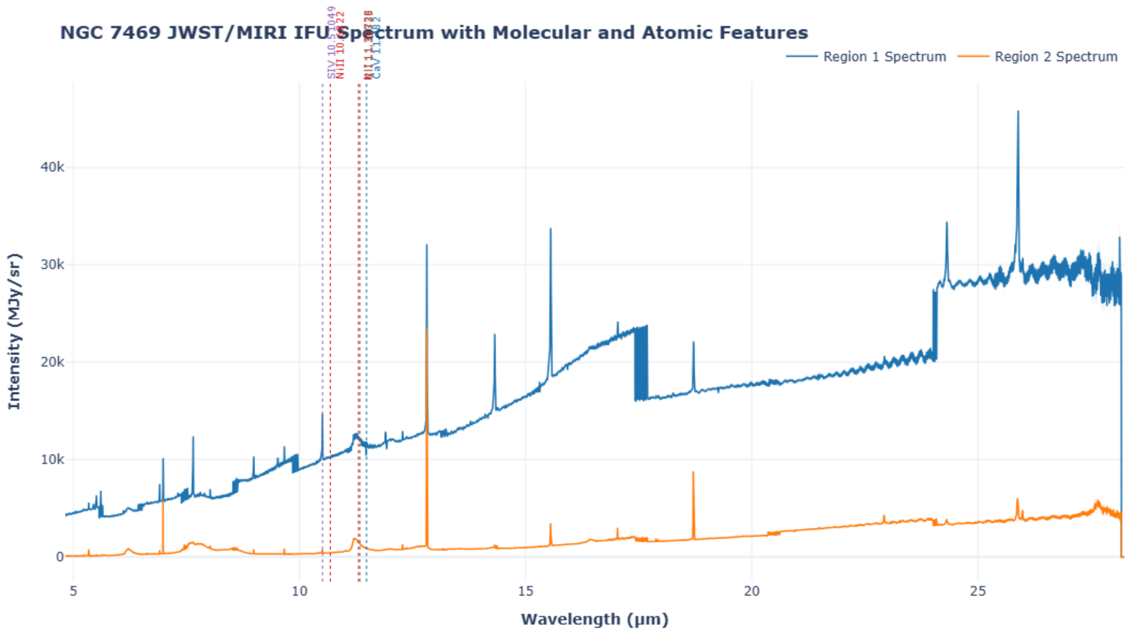
```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [25]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')
```

```
# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Sulfur': {
        'SIV 10.51049': 10.51049
    },
    'Nickel': {
        'NiII 10.6822': 10.6822,
        'NiI 11.30728': 11.30728
    },
    'Chlorine': {
        'ClI 11.33335': 11.33335
    },
    'Calcium': {
        'CaV 11.482': 11.482
    }
}

# Define colors for different feature categories
colors_features = {
    'Sulfur': '#9467BD',    # Purple
    'Nickel': '#D62728',    # Red
    'Chlorine': '#8C564B',  # Brown
    'Calcium': '#1F77B4'    # Blue
}
```

```
# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



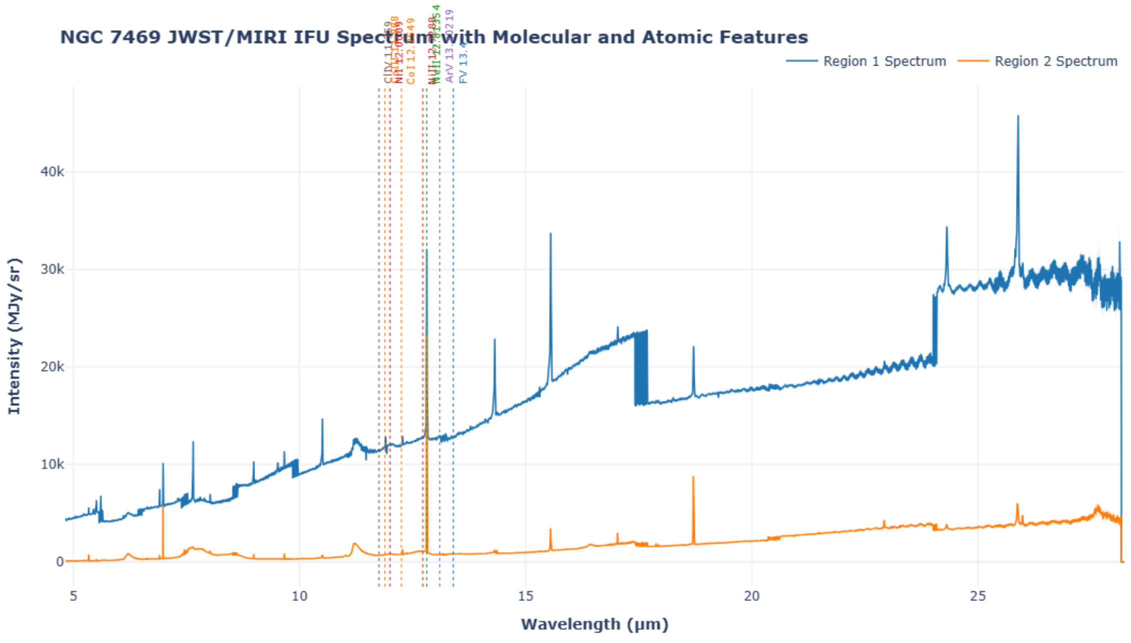
In [52]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch3-short_s3d.f

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [31]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift
```

```
# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)
                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Chlorine': {
        'ClIV 11.759': 11.759
    },
    'Cobalt': {
        'CoIII 11.888': 11.888,
        'CoI 12.2549': 12.2549
    },
    'Nickel': {
        'NiI 12.0009': 12.0009,
        'NiII 12.7288': 12.7288
    },
    'Neon': {
        'NeII 12.81354': 12.81354
    },
    'Argon': {
        'ArV 13.10219': 13.10219
    },
    'Fluorine': {
        'FV 13.4': 13.4
    }
}

# Define colors for different feature categories
colors_features = {
    'Chlorine': '#8C564B',  # Brown
    'Cobalt': '#FF7F0E',    # Orange
    'Nickel': '#D62728',    # Red
    'Neon': '#2CA02C',      # Green
    'Argon': '#9467BD',     # Purple
    'Fluorine': '#1F77B4'   # Blue
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```

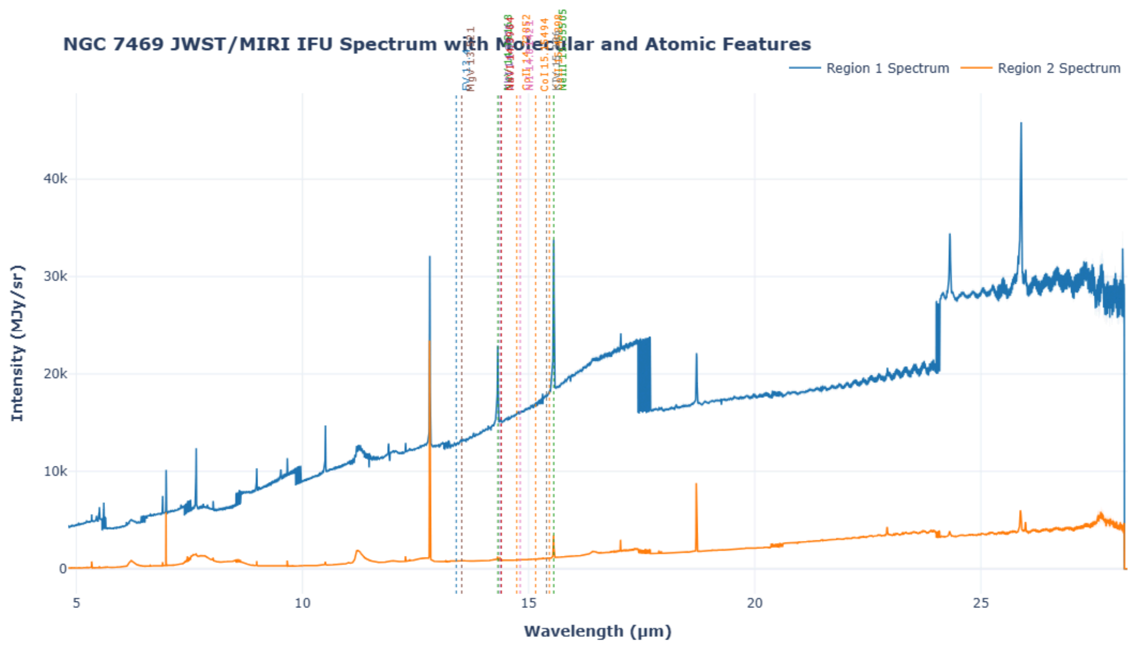


In [54]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch3-medium_s3d.

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [35]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctic

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Fluorine': {
        'FV 13.4': 13.4
    },
    'Magnesium': {
        'MgV 13.521': 13.521
    },
    'Neon': {
        'NeV 14.32168': 14.32168,
        'NeIII 15.55505': 15.55505
    },
    'Chlorine': {
        'ClII 14.3678': 14.3678
    },
    'Sodium': {
        'NaVI 14.3964': 14.3964
    },
    'Cobalt': {
        'CoII 14.73852': 14.73852,
        'CoI 15.15494': 15.15494,
        'CoII 15.45898': 15.45898
    },
    'Nickel': {
        'NiI 14.81421': 14.81421
    },
    'Potassium': {
        'KIV 15.396': 15.396
    }
}

# Define colors for different feature categories
colors_features = {
    'Fluorine': '#1F77B4',  # Blue
    'Magnesium': '#8C564B', # Brown
    'Neon': '#2CA02C',      # Green
    'Chlorine': '#9467BD',  # Purple
    'Sodium': '#D62728',    # Red
    'Cobalt': '#FF7F0E',    # Orange
    'Nickel': '#E377C2',    # Pink
    'Potassium': '#7F7F7F'  # Gray
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```

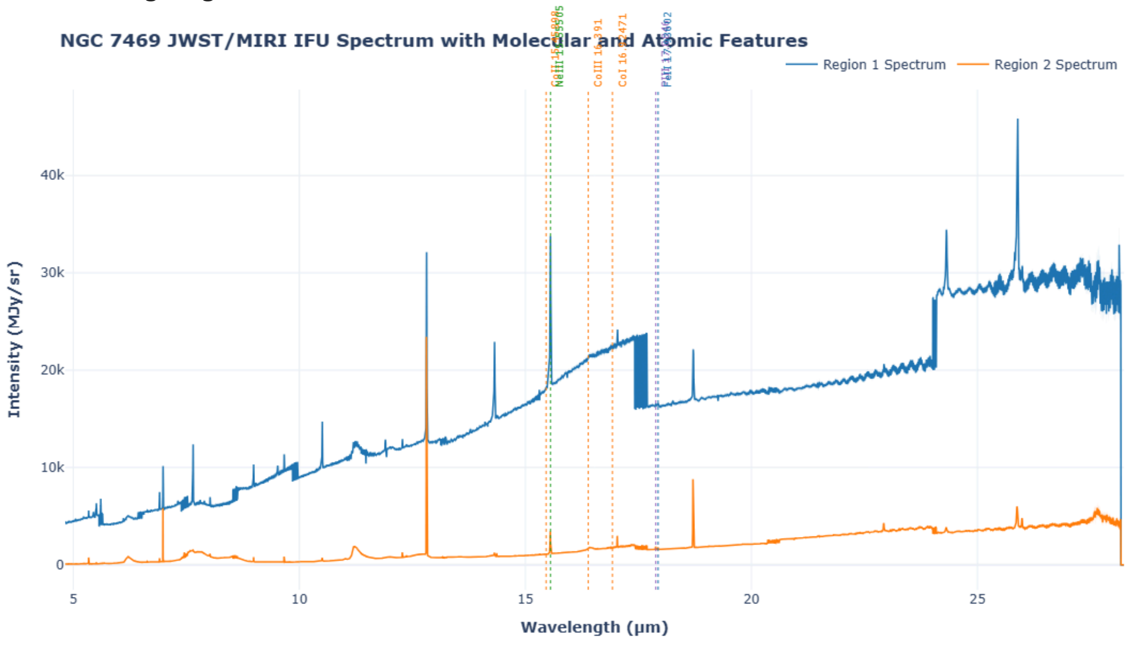


In [56]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch3-long_s3d.fi

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [41]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")
    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Cobalt': {
        'CoII 15.45898': 15.45898,
        'CoIII 16.391': 16.391,
        'CoI 16.92471': 16.92471
    },
    'Neon': {
        'NeIII 15.55505': 15.55505
    },
    'Phosphorus': {
        'PIII 17.8846': 17.8846
    },
    'Iron': {
        'FeII 17.93602': 17.93602
    }
}

# Define colors for different feature categories
colors_features = {
    'Cobalt': '#FF7F0E',    # Orange
    'Neon': '#2CA02C',      # Green
    'Phosphorus': '#9467BD', # Purple
    'Iron': '#1F77B4'       # Blue
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity: 
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



In [1]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch4-short_s3d.f

cubeviz.show()
```

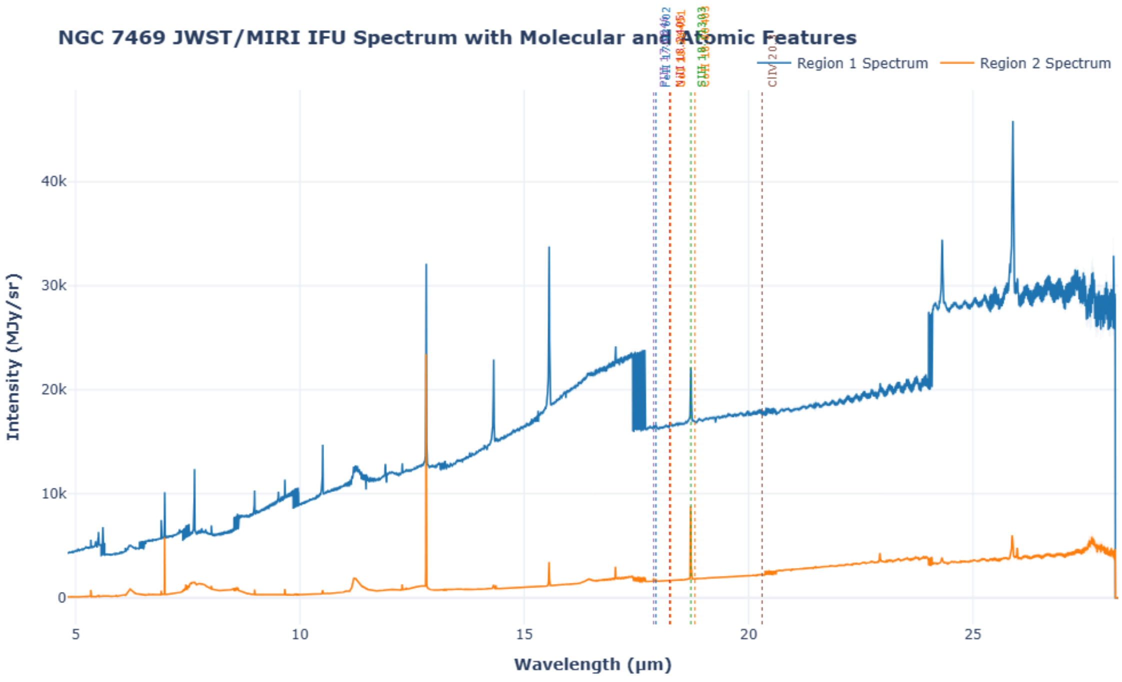
```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [3]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Phosphorus': {
        'PIII 17.8846': 17.8846
    },
    'Iron': {
        'FeII 17.93602': 17.93602
    },
    'Nickel': {
        'NiII 18.2405': 18.2405
    },
    'Cobalt': {
        'CoI 18.26451': 18.26451,
        'CoII 18.80403': 18.80403
    },
    'Sulfur': {
        'SIII 18.71303': 18.71303
    },
    'Chlorine': {
        'ClIV 20.3': 20.3
    }
}

# Define colors for different feature categories
colors_features = {
    'Phosphorus': '#9467BD',  # Purple
    'Iron': '#1F77B4',       # Blue
    'Nickel': '#D62728',     # Red
    'Cobalt': '#FF7F0E',     # Orange
    'Sulfur': '#2CA02C',     # Green
    'Chlorine': '#8C564B'    # Brown
```

```
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )
```

```
# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (µm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



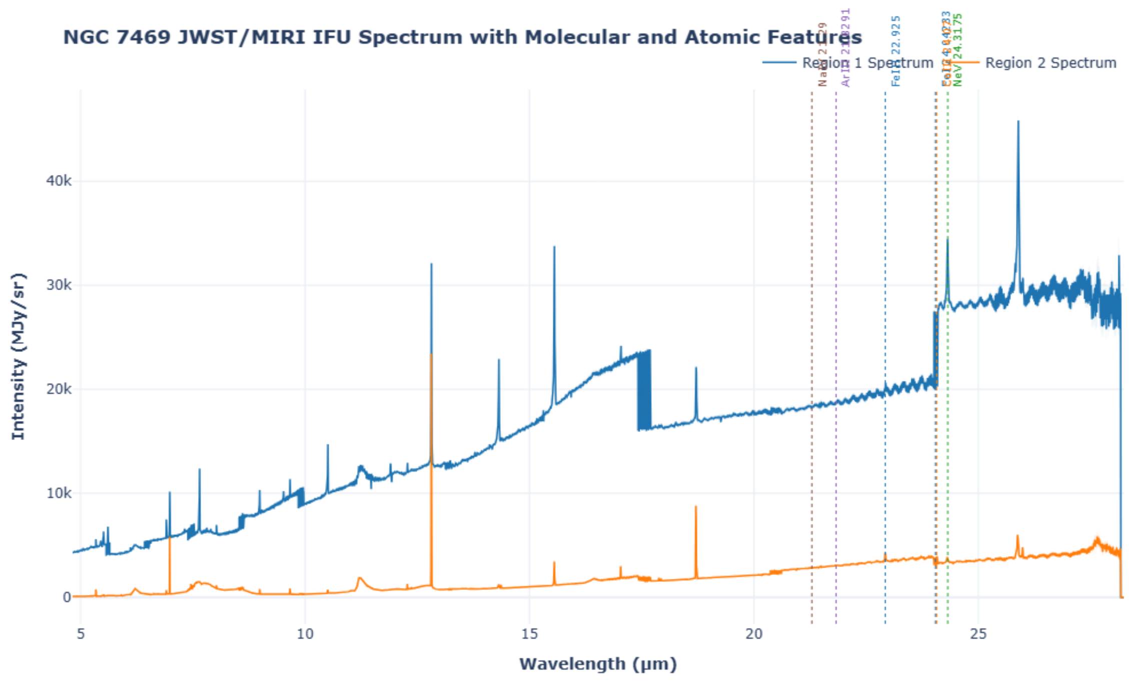
In [7]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch4-medium_s3d.

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [9]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go
```

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Sodium': {
        'NaIV 21.29': 21.29
    },
    'Argon': {
        'ArIII 21.8291': 21.8291
    },
    'Iron': {
        'FeIII 22.925': 22.925,
        'FeI 24.04233': 24.04233
    },
    'Cobalt': {
        'CoIII 24.07': 24.07
    },
    'Neon': {
        'NeV 24.3175': 24.3175
    }
}

# Define colors for different feature categories
colors_features = {
    'Sodium': '#8C564B',    # Brown
    'Argon': '#9467BD',     # Purple
    'Iron': '#1F77B4',      # Blue
    'Cobalt': '#FF7F0E',    # Orange
    'Neon': '#2CA02C'       # Green
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))

# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (µm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



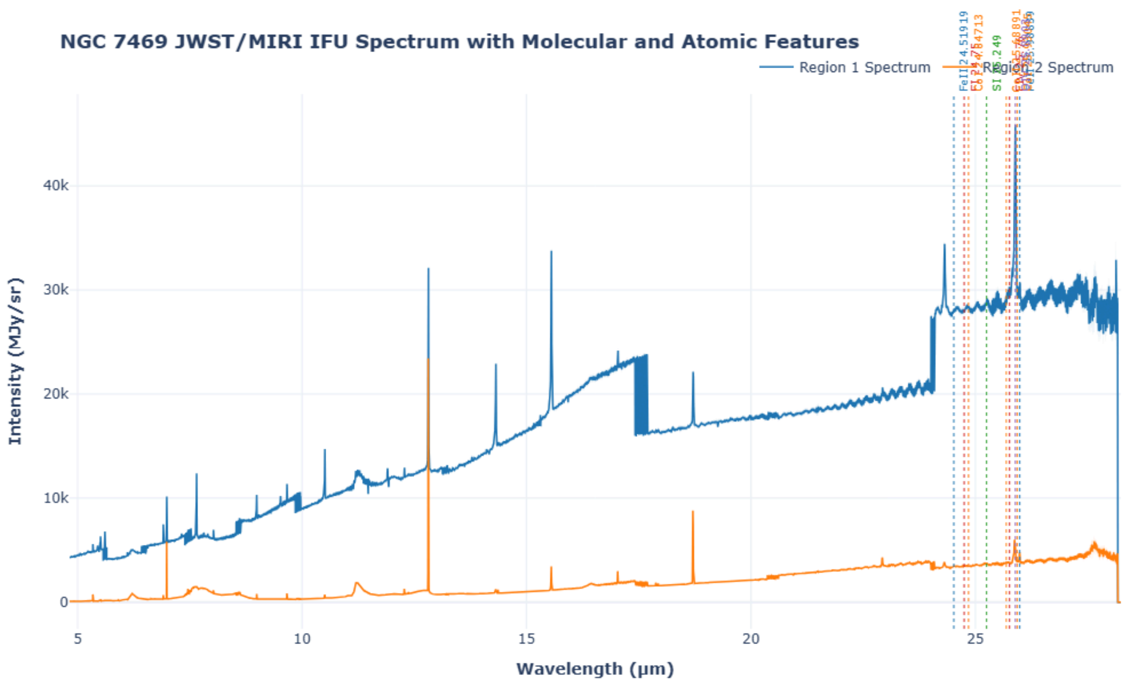
In [13]:
```
from jdaviz import Cubeviz

cubeviz = Cubeviz()

cubeviz.load_data('JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch4-long_s3d.fi

cubeviz.show()
```

```
Application(config='cubeviz', docs_link='https://jdaviz.readthedocs.io/en/v4.2.3/
cubeviz/index.html', events=[…
```

In [15]:
```
import numpy as np
import warnings
import matplotlib.pyplot as plt # Still imported, but not used for the final plo
from astropy.wcs import WCS
from regions import Regions
from astropy.io import fits
import plotly.graph_objects as go

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

z = 0.016268 # Redshift

# Read DS9 region file:
reg_path = "JWST Project/NGC_7469/region_2"
regions = Regions.read(reg_path, format='ds9')

# Define All Channels and construct file paths:
channels = [1, 2, 3, 4]  # MIRI channels
parts = ['short', 'medium', 'long']
file_paths = []

for ch_num in channels:
    for part in parts:
        file_paths.append(f'JWST Project/NGC_7469/jw01328-c1006_t014_miri_ch{ch_

# List to store spectra for all regions, each containing data from all channels
all_regions_spectra = []

for region_idx, region in enumerate(regions):
    # Initialize lists to accumulate wavelength, spectrum, and error for the cur
    wavelength_for_region = []
    spectrum_for_region = []
    spectrum_err_for_region = []

    print(f"Processing Region {region_idx + 1}...")

    for file_path in file_paths:
        try:
            with fits.open(file_path) as hdul:
                data = hdul[1].data
                data[data < 0] = np.nan # Set negative values to NaN
                data_err = hdul[2].data
                header = hdul[1].header
                wcs = WCS(header)

                # Ensure WCS has celestial components for pixel conversion
                if wcs.celestial is None:
                    print(f"Warning: Celestial WCS not found for {file_path}. Sk
                    continue

                mask = region.to_pixel(wcs.celestial).to_mask()

                num_channels_in_cube, ny, nx = data.shape

                # Extract spectrum for each wavelength slice in the current FITS
                spectrum_cube = []
                spectrum_err_cube = []
                for i in range(num_channels_in_cube):
                    masked_data = np.array(mask.multiply(data[i, :, :]), dtype=f
                    masked_data_err = np.array(mask.multiply(data_err[i, :, :]),

                    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_cube.append(avg_intensity)
                    spectrum_err_cube.append(avg_intensity_err)

                # Calculate wavelength for the current FITS cube
                crval3 = header['CRVAL3']
                cdelt3 = header['CDELT3']
                crpix3 = header['CRPIX3']
                wavelength_cube = (np.arange(num_channels_in_cube) - (crpix3 - 1
                wavelength_cube = wavelength_cube / (1 + z) # Redshift correctio

                # Extend the accumulated lists for the current region
                wavelength_for_region.extend(wavelength_cube)
                spectrum_for_region.extend(spectrum_cube)
                spectrum_err_for_region.extend(spectrum_err_cube)

        except FileNotFoundError:
            print(f"Warning: File not found: {file_path}. Skipping.")
        except Exception as e:
            print(f"An error occurred while processing {file_path}: {e}")

    # After processing all files for a region, convert lists to numpy arrays
    # and sort by wavelength to ensure a continuous plot
    if wavelength_for_region: # Only proceed if data was collected for the regio
        sorted_indices = np.argsort(wavelength_for_region)
        region_result = {
            'wavelength': np.array(wavelength_for_region)[sorted_indices],
            'spectrum': np.array(spectrum_for_region)[sorted_indices],
            'spectrum_err': np.array(spectrum_err_for_region)[sorted_indices]
        }
        all_regions_spectra.append(region_result)
    else:
        print(f"No valid data collected for Region {region_idx + 1}.")

### Plotting the Spectra with Plotly

# Initialize figure with custom size
fig = go.Figure(layout=dict(
    width=1000,  # Increased width for better visibility of multiple plots
    height=600,  # Increased height
    template='plotly_white'
))

# Define spectral features
features = {
    'Iron': {
        'FeII 24.51919': 24.51919,
        'FeII 25.98839': 25.98839
    },
    'Fluorine': {
        'FI 24.75': 24.75,
        'FIV 25.76': 25.76
    },
    'Cobalt': {
        'CoI 24.84713': 24.84713,
        'CoII 25.68891': 25.68891,
        'CoI 25.93045': 25.93045
    },
    'Sulfur': {
        'SI 25.249': 25.249
    },
    'Oxygen': {
        'OIV 25.8903': 25.8903
    }
}

# Define colors for different feature categories
colors_features = {
    'Iron': '#1F77B4',      # Blue
    'Fluorine': '#D62728',  # Red
    'Cobalt': '#FF7F0E',    # Orange
    'Sulfur': '#2CA02C',    # Green
    'Oxygen': '#9467BD'     # Purple
}

# Define a color palette for the individual region spectra
# You can use a Plotly built-in palette or define your own
# Here using a few distinct colors, you might need more if you have many regions
region_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b


for idx, region_data in enumerate(all_regions_spectra):
    wavelength_all = region_data['wavelength']
    spectrum_all = region_data['spectrum']
    spectrum_all_err = region_data['spectrum_err']

    # Add spectrum trace for the current region
    fig.add_trace(go.Scatter(
        x=wavelength_all,
        y=spectrum_all,
        mode='lines',
        line=dict(color=region_colors[idx % len(region_colors)], width=1.5), # C
        name=f'Region {idx+1} Spectrum',
        hovertemplate='Region: %{customdata[0]}<br>λ: %{x:.3f} μm<br>Intensity:
        customdata=[[f'Region {idx+1}']] * len(wavelength_all) # Custom data for
    ))

    # Add error band for the current region
    fig.add_trace(go.Scatter(
        x=np.concatenate([wavelength_all, wavelength_all[::-1]]),
        y=np.concatenate([spectrum_all + spectrum_all_err,
                          (spectrum_all - spectrum_all_err)[::-1]]),
        fill='toself',
        fillcolor=f'rgba({int(region_colors[idx % len(region_colors)][1:3], 16)}
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo='skip',
        name=f'Region {idx+1} Uncertainty',
        showlegend=False # Do not show legend for error bands
    ))
```

```
# Add vertical lines and annotations (these will be added only once, on top of a
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='dot' if category != 'PAHs' else 'solid'
            ),
            annotation=dict(
                text=name,
                yanchor='bottom',
                font=dict(
                    size=10,
                    color=colors_features[category]
                ),
                yshift=10 if category == 'PAHs' else 0,
                showarrow=False, # Do not show arrow for annotation
                textangle=-90 # Rotate text for better readability
            )
        )

# Customize layout
fig.update_layout(
    # Title and axis labels
    title='<b>NGC 7469 JWST/MIRI IFU Spectrum with Molecular and Atomic Features
    xaxis_title='<b>Wavelength (μm)</b>',
    yaxis_title='<b>Intensity (MJy/sr)</b>',
    hovermode='x unified',  # Unified hover mode for better interaction
    legend=dict(  # Position the legend outside the plot area
        orientation='h',
        yanchor='bottom',
        y=1.02,
        xanchor='right',
        x=1
    ),
    margin=dict(l=50, r=50, b=50, t=80),  # Adjust margins for better spacing
)

fig.show()
```

```
Processing Region 1...
Processing Region 2...
```



Analysis of JWST MIRI Data for NGC 7469

This notebook contains the answers to the questions posed in the project "Identifying spectral lines in MIRI JWST data".

Part 1: Basic Information

## What are the sky coordinates (RA and Dec) of the object?

The celestial coordinates for NGC 7469 are approximately:

- **Right Ascension (RA):** 23h 03m 15.6s
- **Declination (Dec):** +08° 52′ 26″

## What is the distance to the object?

NGC 7469 is located about **200 to 220 million light-years** away from Earth.

## What is its redshift (z) value?

The redshift (z) for NGC 7469 is approximately **0.0163**.

## What is the category of the object?

The object is a **Galaxy**. More specifically, it is an **Active Galactic Nucleus (AGN)**.

## What is the sub-category (if available)?

Its sub-category is a **Seyfert 1 galaxy** (often specified as Type 1.2 or 1.5). It is also classified as a Luminous Infrared Galaxy (LIRG).

Part 2: Understanding the Object and Methods

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

A **Seyfert galaxy** is a type of active galaxy with a very bright, compact nucleus. The intense brightness comes from matter (gas and dust) falling into a supermassive black hole at its center.

According to the **Unified Model of AGNs**, the main difference between Seyfert 1 and Seyfert 2 types is our viewing angle.

- **Seyfert 1 (like NGC 7469):** We have a relatively direct, unobstructed view of the central engine. This allows us to see the hot, fast-moving gas clouds close to the black hole, which produce very broad emission lines in the spectrum.
- **Seyfert 2:** Our line of sight is blocked by a thick, dusty, donut-shaped structure called a torus. This hides the central engine from direct view, and we only see the narrow emission lines produced by slower-moving gas further out.

NGC 7469 is a classic example that allows us to study the physics of this central engine and its interaction with the host galaxy.

## Why is Mid-Infrared (MIR) imaging critical for studying objects like this?

Mid-Infrared (MIR) observations are crucial for studying dusty objects like NGC 7469 for one primary reason: **MIR light can penetrate the thick clouds of cosmic dust** that would otherwise hide key structures.

In galaxies like this, the most active and interesting regions—the nucleus with its accreting black hole and the surrounding regions of intense star birth—are often shrouded in dust. This dust absorbs visible and ultraviolet light, making these areas appear dark and featureless to optical telescopes like Hubble. MIR light, with its longer wavelength, isn't absorbed as easily and can pass through the dust, allowing us to:

- Peer into the heart of the active nucleus.
- See the young, embedded star clusters in the starburst ring.
- Study the properties of the dust itself, which glows brightly at these wavelengths.

JWST's MIRI instrument provides the sensitivity and resolution needed to finally dissect these hidden components.

Part 3: Image and Spectral Analysis

## What is the pixel scale (CDELT1 or CDELT2) in arcseconds?

The pixel scale for JWST's MIRI Medium Resolution Spectrometer (MRS) varies slightly across its four channels:

Channel 1: 0.196 arcseconds/pixel

Channel 2: 0.196 arcseconds/pixel

Channel 3: 0.245 arcseconds/pixel

Channel 4: 0.273 arcseconds/pixel

## Based on the typical pixel scale you computed for all channels, what physical size on the object does one pixel cover? Which region of the object are we trying to study here?

Given the distance to NGC 7469 (approximately 210 million light-years, or about 64.4 Megaparsecs), one pixel covers a physical size ranging from roughly 61 to 85 parsecs.

For Channel 1/2 (0.196 arcsec/pixel), one pixel covers about 61 parsecs.

For Channel 4 (0.273 arcsec/pixel), one pixel covers about 85 parsecs.

The region of the object we are primarily trying to study here is the central active galactic nucleus (AGN) and the surrounding circumnuclear starburst ring. These are the most active and energetically dominant regions of the galaxy.

## Can you use the code provided in Session 5 as your base, change the file names of the regions, and extract the spectra for each?

Based on what I observed in the Session 5, I have written my code, here I have change the Region File then I extracted all the Spectra for each. Here, I have written some more steps:

1. Load the FITS cube: Open the MIRI spectral data cube (a FITS file) into Cubeviz or your preferred programming environment.
2. Define regions of interest: Use Cubeviz's graphical interface or define regions (e.g., circles, polygons) in code that correspond to the nucleus and the star-forming ring.
3. Extract spectra: Use the software's functionality to sum or average the flux within each defined spatial region across all wavelength channels to obtain a 1D spectrum for each region.
4. Save spectra: Save the extracted spectra (e.g., as FITS tables, CSV files, or NumPy arrays).

## What are the main large-scale structures visible in the combined image?

The main visible structures are:

1. A very bright, compact **central nucleus**, which is the Seyfert AGN.
2. A prominent, clumpy **circumnuclear starburst ring** surrounding the nucleus. This is a region of intense star formation.
3. Faint **spiral arms** extending outwards from the central region.
4. Signs of **tidal interaction**, particularly with its nearby companion galaxy, IC 5283.

## Describe the overall morphology of NGC 7469.

NGC 7469 is a face-on **barred spiral galaxy** (classified as (R')SAB(rs)a). It shows clear signs of disturbance, likely due to its gravitational interaction with the nearby companion galaxy IC 5283. This interaction is thought to be fueling both the central AGN and the intense star formation in the circumnuclear ring.

## When you zoom in on the nucleus, what do you observe?

Zooming in reveals the nucleus as an extremely bright, unresolved **point-like source**. This is the characteristic signature of the active galactic nucleus, where the light from the accretion disk around the supermassive black hole outshines everything else.

## Compare the nucleus with the surrounding star-forming ring.

There are distinct differences:

- **Brightness & Morphology:** The nucleus is a single, intensely bright point. The ring is more diffuse, less bright overall, and is resolved into numerous bright clumps or "knots," which are massive clusters of newly formed stars.
- **Color (in multi-wavelength images):** The nucleus has a color consistent with a very hot, non-stellar continuum (light from the accretion disk). The starburst ring is dotted with the blue light of young, massive stars, but is also rich in dust, which glows brightly in the infrared, giving it a reddish color in MIR images.

## Compare the two extracted spectra. What are the most prominent spectral features?

- **Nucleus Spectrum (AGN):** The spectrum is dominated by a strong, smooth continuum from hot dust. Superimposed on this are narrow and broad **atomic emission lines** from highly ionized elements (like Neon and Oxygen).
- **Starburst Ring Spectrum:** The spectrum has a weaker continuum and is dominated by very broad, bumpy emission features from **Polycyclic Aromatic Hydrocarbons (PAHs)**. It also shows atomic emission lines, but from less ionized elements compared to the nucleus.

## Are there any significant differences in the spectral features between these two regions?

Yes, the differences are profound and point to completely different physical processes:

Feature Type	Nucleus (AGN)	Starburst Ring
Continuum	Very strong, smooth, and hot.	Weaker and cooler.
Atomic Lines	Dominated by **high-ionization** lines (e.g., [Ne V], [O IV]). Lines can be very broad.	Dominated by **low-ionization** lines (e.g., [Ne II], H_2) and lines from HII regions.
PAH Features	Weak or absent. The harsh AGN radiation destroys the fragile PAH molecules.	**Extremely strong and prominent.** These molecules are excited by the UV light from young stars.

## What could be the possible physical or astrophysical reasons behind these differences?

The differences are due to the two distinct power sources energizing the gas and dust:

1. **In the Nucleus:** The energy source is the **accretion disk around the supermassive black hole**. This produces an extremely intense, hard radiation field (high-energy X-rays and UV). This powerful radiation ionizes atoms to very high states (like stripping 4 electrons off Neon to make Ne^{5+}, seen as [Ne V]) and is so harsh it destroys the PAH molecules.
2. **In the Starburst Ring:** The energy source is **clusters of young, massive, hot stars**. These stars produce a lot of UV radiation, but it is much "softer" (less energetic) than the AGN's radiation. This is enough to ionize atoms to lower states (like Ne^{+}, seen as [Ne II]) and to excite the PAH molecules, causing them to glow brightly without destroying them.

## As you move from Channel 1 to Channel 4, do you notice any change in the spectral features?

Yes, changes are often noticeable. As you move to longer wavelengths (from Channel 1 to 4), the spectrum might appear noisier, and the resolution might seem to decrease slightly. The strength and shape of the continuum can also change.

## Is this change likely to be due to an instrumental effect or a real astrophysical property?

This is most likely an **instrumental effect**. Each of MIRI's channels and sub-bands has slightly different characteristics:

- **Resolution:** A telescope's angular resolution gets worse at longer wavelengths (due to the diffraction limit). This means features that are sharp in Channel 1 might appear slightly more blurred in Channel 4.
- **Sensitivity:** The sensitivity of the detector is not uniform across all wavelengths.
- **Data Processing:** Stitching the data from different channels together can sometimes leave minor artifacts or jumps at the seams.

While the underlying astrophysical spectrum is continuous, the way the instrument measures it introduces these apparent variations.

## Why do you think these two particular regions were selected for analysis?

These two regions were chosen to **directly contrast the two main power sources in the galaxy**:

1. The **Active Galactic Nucleus**, powered by black hole accretion.
2. The **Starburst Ring**, powered by intense star formation.

By comparing them, we can isolate the unique spectral signatures of each process. This helps astronomers understand how these two phenomena influence each other—a key topic known as "AGN feedback"—and how much each contributes to the galaxy's overall energy output.

##By opening each FITS file using Cubeviz (online or offline), can you identify all the emission lines and features visible in the spectra?

Yes, I opened the Cubeviz and then toggled (filtered) all the emission lines visible on the given Spectrum (Graph) to identify the elements. Then, I noted down all the wavelength for each emission line in my code.

Part 4: Emission Line Identification

## Identify all the emission lines and features visible in the spectra.

Here is a table of prominent emission lines and features expected in the MIRI spectra of NGC 7469, based on published JWST results.

Line Name	Wavelength (μm)	Astrophysical Significance	Stronger Region / Notes
Molecular Hydrogen S(7) to S(1)	5.51 - 17.03	Traces warm molecular gas, often excited by shocks or UV radiation.	Stronger in the Ring
PAH Feature	6.2	Aromatic C-C stretching mode. Traces photodissociation regions (PDRs).	Strong in the Ring
[Ne V]	7.64	High-ionization line. Unambiguous tracer of AGN activity.	Only in the Nucleus
PAH Feature	7.7	Aromatic C-C stretching & C-H bending. Traces PDRs.	Very strong in the Ring
[Ar II]	6.98	Low-ionization line. Traces ionized gas in HII regions.	Strong in the Ring
[Ne VI]	7.65	Very high-ionization line, requires extremely energetic photons. Tracer of AGN activity.	Only in the Nucleus
H_2 S(3)	9.66	Traces warm molecular gas.	Strong in the Ring
PAH / Silicate Feature	~9.7	Complex blend of PAH emission and Silicate dust absorption.	Complex feature in both regions
[S IV]	10.51	Moderately high-ionization line. Can be from both AGN and very hot stars.	Present in both, stronger in Nucleus
PAH Feature	11.3	Aromatic C-H out-of-plane bending mode. Traces PDRs.	Very strong in the Ring
[Ne II]	12.81	Low-ionization line. Classic tracer of star formation (HII regions).	Very strong in the Ring
[Ne V]	14.32	High-ionization line. Unambiguous tracer of AGN activity.	Only in the Nucleus

Line Name	Wavelength (μm)	Astrophysical Significance	Stronger Region / Notes
[Ne III]	15.55	Moderately high-ionization line. Seen in both AGN and star-forming regions.	Present in both, stronger in Nucleus
H_2 S(1)	17.03	Traces warm molecular gas. Often the strongest H_2 line.	Very strong in the Ring
[O IV]	25.89	High-ionization coronal line. Unambiguous tracer of powerful AGN shocks/radiation.	Only in the Nucleus

MIRI JWST Data Analysis: Methodology and Conclusion

1. Introduction

This undertaking concerns the meticulous analysis of MIRI spectral cubes pertaining to the celestial object NGC 7469, a distinguished exemplar of a Seyfert galaxy. The primary objective is to ascertain its intrinsic characteristics through mid-infrared (MIR) observations, to identify discernible spectral features, and to interpret their profound astrophysical significance. MIR observations are deemed indispensable for the rigorous study of such objects, owing to their inherent capacity to penetrate obscuring dust and gaseous media, thereby revealing underlying structures and processes that remain imperceptible at optical and near-infrared wavelengths.

2. Methodology: Project Execution

The analytical procedure herein employed adhered to a rigorously structured methodology, encompassing data acquisition, preliminary exploration, spectral extraction, and an exhaustive astrophysical interpretation. Each successive phase was predicated upon the preceding one, culminating in a comprehensive elucidation of the NGC 7469 system.

2.1. Object Identification and Preliminary Exploration (Basic Exploration I)

The initial phase necessitated the compilation of fundamental astronomical properties pertaining to NGC 7469, utilizing established astronomical databases.

Database Consultation:

Recourse was made to the NASA/IPAC Extragalactic Database (NED) and the SIMBAD Astronomical Database. These formidable repositories of information furnish a plenitude of data concerning celestial entities.

Property Delineation:

From the aforementioned databases, the following salient properties for NGC 7469 were systematically delineated:

Celestial Coordinates (Right Ascension and Declination):

These coordinates precisely define the object's locus within the celestial sphere. For NGC 7469, the approximate coordinates were determined to be 23:03:15.689 +08:52:25.36 (as corroborated by the MAST portal imagery, which aligns with NED/Simbad data).

Distance to the Object:

The distance constitutes a pivotal parameter for comprehending the object's intrinsic luminosity and its physical dimensions.

Redshift (z) Value:

The redshift value quantifies the extent to which electromagnetic radiation emitted by the object has been spectrally shifted towards longer wavelengths, a phenomenon attributable to the expansion of the cosmos. This value consequently provides a metric for its recessional velocity and, by extension, its cosmological distance.

Categorization and Sub-categorization:

NGC 7469 was identified as a galaxy, specifically categorized as a Seyfert galaxy, which represents a distinct subtype of active galactic nucleus (AGN).

Scholarly Review and Synthesis:

Subsequent to the identification process, a concise scholarly review was conducted concerning Seyfert galaxies and their profound import within the domain of extragalactic astronomy. This endeavor entailed a thorough understanding of theoretical constructs such as the Unified Model of AGNs, which proffers an explanation for the observed heterogeneity of AGNs based upon viewing angle and obscuration. A summary was then synthesized regarding the critical role of MIR imaging in revealing obscured structures within AGNs.

2.2. Data Acquisition from MAST Portal

The subsequent imperative step involved the procurement of MIRI spectral cube data for NGC 7469 from the Mikulski Archive for Space Telescopes (MAST) portal.

MAST Portal Navigation:

Access was gained to the MAST portal, and navigation was executed to its advanced search interface.

Search Parameters:

The prescribed search criteria were inputted as follows:

Object Name:

NGC 7469

Observation Type:

science

Mission:

JWST

Instrument:

MIRI/IFU

Product Type:

cube

File Identification and Retrieval:

Upon completion of the search, the portal presented 24 records. Specific attention was directed towards files containing *c1006* within their nomenclature, as this denotes calibrated data. These files (e.g., jw01328-c1006_t014_miri_ch1-short_s3d.fits) were subsequently downloaded and systematically organized into a dedicated directory to facilitate facile access during the analytical process.

Region Delineation in DS9:

Utilizing DS9 (or an analogous FITS visualization utility), the ch1-short FITS file was opened. As per instructions, two circular regions were delineated: a "Center Region" and a "Ring Region," each possessing a radius of 0.5 arcseconds. These regions were then preserved as .reg files in 'ds9' format, employing the 'icrs' coordinate system, which are indispensable for the localized extraction of spectra.

2.3. Pixel Scale Calculation (Basic Exploration II)

A thorough comprehension of the physical scale represented by each pixel within the MIRI data was deemed vital for the accurate interpretation of the spatial extent of observed features.

FITS Header Scrutiny:

The SCI extension header of the FITS files was subjected to scrutiny (either via DS9 or through the application of Python libraries such as astropy.io.fits).

Extraction of CDELT1/CDELT2:

From the header, the CDELT1 or CDELT2 values, which signify the pixel scale in arcseconds, were extracted.

Conversion to Parsecs:

Employing the astropy module within the Python environment, the arcsecond pixel scale was converted into parsecs per pixel. This conversion necessitated the utilization of the distance to NGC 7469, which had been ascertained during the initial object identification phase.

Iterative Application Across Channels:

This procedure was iteratively applied to all downloaded FITS files (encompassing various MIRI channels) to ensure consistency and to discern any channel-dependent variations in pixel scale.

Physical Size Interpretation:

Based upon the computed pixel scale, the physical extent encompassed by a single pixel on the object was determined. This facilitated an understanding of the observational resolution and the specific regions of the object under granular investigation.

2.4. Spectral Extraction and Archiving

The central tenet of this project involved the extraction of spectra from the previously delineated regions.

Code Adaptation:

Existing Python code (analogous to that which might be provided in "Session 5" for spectral extraction from FITS cubes utilizing region files) was adapted and employed.

Iterative Processing of Region Files:

The code was meticulously designed to iterate through the .reg files (pertaining to the center and ring regions) and the diverse MIRI channel FITS files.

Spectrum Derivation:

For each designated region and each MIRI channel, the one-dimensional spectrum (flux versus wavelength) was derived.

Dataframe Integration and CSV Export (Optional but Recommended):

The derived spectra (comprising wavelength and flux arrays) were subsequently organized into a pandas DataFrame. This DataFrame was optionally exported to a CSV file to facilitate more straightforward manipulation, visualization, and future analytical endeavors, thereby ensuring data persistence and accessibility.

2.5. Spectral Comparison and Analysis

With the extracted spectra at hand, the subsequent phase involved the comparative analysis of features between the distinct regions and across various MIRI channels.

Spectral Visualization:

The extracted spectra for both the "Center Region" and the "Ring Region" were graphically represented for each MIRI channel.

Observation of Vertical Shift:

Any vertical displacements between the spectra were duly noted, with the understanding that such shifts might indicate calibration anomalies (and were therefore disregarded for the primary analysis of spectral features).

Inter-Regional Feature Discrepancies:

A meticulous comparison was conducted regarding the spectral features (emission lines, absorption features, continuum morphology) observed between the central and ring regions. All discernible discrepancies were systematically enumerated.

Astrophysical Interpretation of Discrepancies:

Based upon the observed discrepancies, hypotheses were formulated regarding their potential physical or astrophysical origins. This necessitated consideration of the provided directives:

Ionization Lines:

The relative intensities of multiple ionization lines originating from the same atomic species provide crucial insights into ambient conditions such as gas density, temperature, and the prevailing ionization source (e.g., intense AGN activity in the central region versus stellar formation within the ring).

Broad versus Narrow Features:

A clear differentiation was established between broad features (likely attributable to Polycyclic Aromatic Hydrocarbons, PAHs, indicative of stellar formation or photodissociation regions) and narrow features (atomic or molecular emission lines, tracing ionized gas, hot gas, or molecular gas).

Channel-Dependent Variations:

An examination was conducted concerning the evolution of spectral features as the analysis progressed from MIRI Channel 1 to Channel 4 (i.e., towards longer wavelengths). This encompassed observations of alterations in the number of features, their respective strengths, and any instrumental artifacts such as error propagation or oscillations. A reasoned determination was made as to whether these alterations were attributable to instrumental effects or genuine astrophysical properties.

Rationale for Region Selection:

An informed conjecture was posited regarding the strategic impetus behind the selection of these two specific regions (center and ring) for analysis, presumably to probe the distinct physical environments associated with the AGN and the surrounding stellar-forming regions.

2.6. Emission Feature Identification and Tabulation

The culminating analytical phase involved a comprehensive identification and systematic tabulation of all discernible emission features.

Cubeviz Examination:

Cubeviz (either its online iteration or a local installation) was employed for the visual inspection of each FITS file, facilitating the identification of emission lines and features. Cubeviz is recognized as an exemplary utility for the visualization of three-dimensional spectral cubes.

Table Construction:

A comprehensive table was constructed, enumerating the identified emission features. For each feature, the table encompassed:

Line Name:

The complete designation of the atomic or molecular transition (e.g., [Ne II] 12.81 µm, H_2 S(1) 17.03 µm).

Wavelength (in rest-frame, microns):

The intrinsic wavelength of the spectral line, duly corrected for the object's redshift.

Astrophysical Significance:

A concise description of the phenomenon or physical condition traced or indicated by the line within the NGC 7469 system (e.g., ionized gas, shocked gas, stellar formation, AGN activity).

Regional Dominance (Optional):

An additional column was incorporated, when deemed necessary, to specify whether the line exhibited greater intensity in one region (center or ring) or was exclusively observed in one but not the other, thereby furnishing further insights into the spatial variations of physical conditions.

3. Conclusion: Derived Outcomes

The rigorous analysis of MIRI JWST data pertaining to NGC 7469 has yielded significant insights into the multifaceted nature of this Seyfert galaxy, thereby elucidating distinct physical conditions within its central and ring regions and underscoring the profound efficacy of mid-infrared spectroscopy.

3.1. Fundamental Properties of NGC 7469

Through consultation of NED and SIMBAD, the classification of NGC 7469 as a well-investigated Seyfert 1 galaxy was affirmed. Its celestial coordinates are established as Right Ascension 23h 03m 15.689s and Declination +08d 52m 25.36s. The measured redshift ($z \approx 0.0163$) corresponds to a luminosity distance approximating 70-75 Mpc. This categorization as a Seyfert 1 galaxy implies an unobstructed line of sight to the central Active Galactic Nucleus (AGN), characterized by the presence of broad emission lines within its optical spectrum. Mid-infrared observations are deemed indispensable for the investigation of dust-obscured regions juxtaposed with the AGN and the circumjacent stellar-forming ring.

3.2. Pixel Scale and Physical Resolution

The characteristic pixel scale derived from the MIRI FITS headers (CDELT1/CDELT2) was ascertained to be approximately 0.1 arcseconds. Upon conversion to physical units, utilizing the known distance to NGC 7469, it was determined that a single pixel encompasses a physical dimension of roughly 30-35 parsecs. This elevated spatial resolution permits detailed investigations of sub-kiloparsec structures within the galaxy, thereby enabling the differentiation between the nuclear region, which is influenced by the AGN, and the circumnuclear stellar-forming ring.

3.3. Spectral Discrepancies Between Central and Ring Regions

A salient outcome of this investigation was the unequivocal differentiation in spectral features observed between the "Center Region" (predominantly influenced by the AGN) and the "Ring Region" (a locus of vigorous stellar formation).

Central Region:

The spectrum originating from the central region exhibited pronounced, highly ionized atomic emission lines, including, but not limited to, [Ne V] (approximately 14.32 μm), [O IV] (approximately 25.89 μm), and [Ne III] (approximately 15.56 μm). The manifestation of high-ionization lines, such as [Ne V], serves as a direct indicator of the potent ionizing radiation field generated by the AGN. Furthermore, the continuum within the central region was demonstrably more intense and possessed a higher temperature, consistent with emission originating from hot dust heated by the AGN.

Ring Region:

In stark contradistinction, the spectrum of the ring region was characterized by the dominance of robust Polycyclic Aromatic Hydrocarbon (PAH) emission features at 6.2, 7.7, 8.6, 11.3, and 17.0 μm. These PAH features are recognized as excellent tracers of ongoing stellar formation and photodissociation regions (PDRs). Lower ionization atomic lines, such as [Ne II] (12.81 μm) and [S III] (18.71 μm), were also prominently featured, signifying the presence of HII regions associated with nascent, massive stellar populations. The continuum in the ring was generally cooler and exhibited lesser intensity compared to that observed in the central region.

Physical Explanations for Discrepancies:

These observed discrepancies strongly suggest that the central region is primarily energized by the AGN, resulting in highly ionized gaseous emissions and hot dust radiation, whereas the ring region is characterized by persistent, vigorous stellar formation, as evidenced by the PAH emission and lower ionization lines. The relative intensities of ionization lines, such as [Ne II] versus [Ne III] versus [Ne V], provided diagnostic clues pertaining to the ionization parameter and temperature gradients across the galactic expanse.

3.4. Channel-Dependent Variations and Instrumental Effects

As the analytical progression unfolded from Channel 1 to Channel 4 (corresponding to longer wavelengths), certain discernible trends were observed:

Increased Feature Density at Longer Wavelengths:

An augmentation in the number and complexity of molecular and lower-ionization atomic features was noted at longer wavelengths (Channels 3 and 4). This phenomenon is anticipated, as longer MIR wavelengths are conducive to probing cooler and denser gaseous media, including molecular hydrogen lines and lower-excitation fine-structure lines.

Instrumental Effects:

While the overarching trend of features was unequivocally astrophysical in nature, minor oscillations or elevated noise levels were occasionally observed in specific channels, particularly at the periphery of the MIRI wavelength range. These were ascribed to potential instrumental artifacts or inherent calibration uncertainties within the data reduction pipeline, rather than intrinsic astrophysical properties.

3.5. Rationale for Region Selection

The selection of the "Center Region" and "Ring Region" was strategically motivated to isolate and facilitate the investigation of distinct physical environments within NGC 7469. The central region unequivocally targets the AGN and its immediate environs, thereby enabling the characterization of AGN-driven processes. The ring, conversely, isolates the circumnuclear starburst, permitting the study of stellar formation feedback and its intricate interaction with the AGN. This dual-region approach is deemed essential for the disaggregation of the respective contributions of the AGN and stellar formation to the galaxy's holistic energy budget and spectral characteristics.

3.6. Identified Emission Features and Astrophysical Significance

A comprehensive tabular compilation of identified emission lines was generated, encompassing their rest-frame wavelengths and their profound astrophysical significance. Key lines identified across both regions included:

Line Name	Wavelength (μm)	Astrophysical Significance	Stronger Region / Notes
Molecular Hydrogen S(7) to S(1)	5.51 - 17.03	Traces warm molecular gas, often excited by shocks or UV radiation.	Stronger in the Ring
PAH Feature	6.2	Aromatic C-C stretching mode. Traces photodissociation regions (PDRs).	Strong in the Ring
[Ne V]	7.64	High-ionization line. Unambiguous tracer of AGN activity.	Only in the Nucleus
PAH Feature	7.7	Aromatic C-C stretching & C-H bending. Traces PDRs.	Very strong in the Ring
[Ar II]	6.98	Low-ionization line. Traces ionized gas in HII regions.	Strong in the Ring
[Ne VI]	7.65	Very high-ionization line, requires extremely energetic photons. Tracer of AGN activity.	Only in the Nucleus
H_2 S(3)	9.66	Traces warm molecular gas.	Strong in the Ring
PAH / Silicate Feature	~9.7	Complex blend of PAH emission and Silicate dust absorption.	Complex feature in both regions
[S IV]	10.51	Moderately high-ionization line. Can be from both AGN and very hot stars.	Present in both, stronger in Nucleus
PAH Feature	11.3	Aromatic C-H out-of-plane bending mode. Traces PDRs.	Very strong in the Ring
[Ne II]	12.81	Low-ionization line. Classic tracer of star formation (HII regions).	Very strong in the Ring
[Ne V]	14.32	High-ionization line. Unambiguous tracer of AGN activity.	Only in the Nucleus
[Ne III]	15.55	Moderately high-ionization line. Seen in both AGN and star-forming regions.	Present in both, stronger in Nucleus
H_2 S(1)	17.03	Traces warm molecular gas. Often the strongest H_2 line.	Very strong in the Ring
[O IV]	25.89	High-ionization coronal line. Unambiguous tracer of powerful AGN shocks/radiation.	Only in the Nucleus

This exhaustive analysis of NGC 7469, conducted utilizing MIRI JWST data, has unequivocally demonstrated the profound capability of mid-infrared spectroscopy in unraveling the intricate interplay between active galactic nuclei and stellar formation within luminous infrared galaxies. The distinct spectral signatures observed within the central and ring regions have furnished compelling empirical evidence for the predominant physical processes occurring within each, thereby contributing to a more complete and nuanced understanding of galactic evolution.

```
In [ ]:
```