Code Documentation: read_Data.py

1. Reading PNG Image Files

The `read_dataset` function processes PNG files and prepares image data and corresponding labels.

Code:

```
def read_dataset(lispath, target, resize_shape=(256, 256)):
    imgs = []
    for file in os.listdir(lispath):
        if file[-3:] == 'png':
            img = cv2.imread(os.path.join(lispath, file))
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, resize_shape)
            imgs.append(np.array(img))
    labels = len(imgs) * [target]
    return np.array(imgs), np.array(labels)
```

Explanation:

- **Inputs**:
  - `lispath`: Directory containing PNG files.
  - `target`: Label to assign to all images.
  - `resize_shape`: Tuple defining the image resizing dimensions (default is `(256, 256)`).
- **Process**:
  - Loops through all files in the specified directory.
  - Checks for files ending with `.png`.
  - Reads the image, converts its color from BGR to RGB, and resizes it.
- **Outputs**:
  - `imgs`: Numpy array of image data.
  - `labels`: Numpy array containing the specified `target` label repeated for all images.

2. Reading FITS Files

The `read_fits` function extracts data from FITS files, commonly used in astronomy.

Code:

```
def read_fits(filename):
    try:
        with fits.open(filename) as hdul:
            data_matrix = hdul[0].data
            if len(hdul) > 1 and isinstance(hdul[1], fits.BinTableHDU):
```

```
            binary_table = hdul[1].data
            freqs = binary_table['FREQUENCY'][0]
            time = binary_table['TIME'][0]
        else:
            raise ValueError("Binary table not found or invalid structure.")
        if data_matrix.shape[0] != len(freqs) or data_matrix.shape[1] != len(time):
            raise ValueError("Data matrix dimensions do not match frequency and time axes.")
        return data_matrix, freqs, time
    except Exception as e:
        print(f"Warning: Error reading FITS file '{filename}': {e}")
        return None, None, None
```

Explanation:

- **Inputs**:
  - `filename`: Path to a FITS file.
- **Process**:
  - Opens the FITS file and extracts:
    - `data_matrix`: 2D data matrix.
    - `freqs`: Frequency axis.
    - `time`: Time axis.
  - Ensures the matrix dimensions align with the axes.
- **Outputs**:
  - `data_matrix`: 2D array of data values.
  - `freqs`: Array of frequency values.
  - `time`: Array of time values.

3. Plotting FITS Data

The `plot_fits_data` function visualizes FITS data before and after background filtering.

Code:

```
def plot_fits_data(data_matrix, freqs, time, title_raw='Raw Spectrum', title_filtered='Filtered Spectrum', vmin=None, vmax=None):
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.imshow(data_matrix, aspect='auto', extent=[time[0], time[-1], freqs[0], freqs[-1]], origin='lower')
    plt.xlabel('Time (s)')
    plt.ylabel('Frequency (MHz)')
    plt.colorbar()
    plt.title(title_raw)

    background = np.mean(data_matrix, axis=1)
    background = np.tile(background, (data_matrix.shape[1], 1)).T
```

```
    filtered_data = data_matrix - background
    filtered_data = np.clip(filtered_data, -1, 25)

    plt.subplot(1, 2, 2)
    plt.imshow(filtered_data, aspect='auto', extent=[time[0], time[-1], freqs[0], freqs[-1]],
origin='lower')
    plt.xlabel('Time (s)')
    plt.ylabel('Frequency (MHz)')
    plt.colorbar()
    plt.title(title_filtered)

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

Explanation:

- **Inputs**:
  - `data_matrix`: Raw 2D data to visualize.
  - `freqs` and `time`: Frequency and time axes.
- **Process**:
  - Displays two plots:
    1. **Raw Data**: Original data matrix.
    2. **Filtered Data**: Background-subtracted data, clipped to a specific range.
- **Features**:
  - Customizable titles (`title_raw` and `title_filtered`).
  - Clipping of filtered data improves visualization.

4. Processing FITS Files in a Directory

The `read_fits_dataset` function processes all `.gz` compressed FITS files in a directory.

Code:

```
def read_fits_dataset(path):
    images = []
    times = []
    freqs = []
    first_image_shape = None

    for file in tqdm(os.listdir(path)):
        try:
            if file.endswith("gz"):
                img, freq, time = read_fits(os.path.join(path, file))

                if img is not None:
                    if first_image_shape is None:
```

```
                first_image_shape = img.shape

            img_resized = cv2.resize(img, (first_image_shape[1], first_image_shape[0]))
            images.append(img_resized)
            freqs.append(freq)
            times.append(time)
        except Exception as e:
            print(f"Warning: Error reading FITS file '{file}': {e}")

    return images, pd.DataFrame({'freqs': freqs, 'times': times})
```

Explanation:

- **Inputs**:
  - `path`: Directory containing `.gz` compressed FITS files.
- **Process**:
  - Reads and processes each FITS file:
    - Resizes images to match the first image's shape.
    - Extracts frequency and time data.
- **Outputs**:
  - `images`: List of processed FITS image data.
  - `data`: Pandas DataFrame containing the frequency and time data.