HETDEX Source Catalog 1: Catalog description and access

In [ ]:
```
# Required python modulules for opening FITS files and plotting
import numpy as np
import os.path as op
import glob

import matplotlib.pyplot as plt
from matplotlib import gridspec

from astropy.io import fits
from astropy.table import Table, hstack
import astropy.units as u
from astropy.coordinates import SkyCoord
from astropy.wcs import WCS

from astropy.visualization import make_lupton_rgb, ZScaleInterval
```

In [ ]:
```
%matplotlib inline
```

In [ ]:
```
# fill in the path to the HETDEX Source Catalog 1 files
path_to_scl = "hetdex_source_catalog_1"
```

Catalog Organization

Two catalogs make up HETDEX Source Catalog 1:

1. The Source Observation Table: hetdex_sc1_vX.dat/.fits/.ecsv
   With SPECTRA arrays included: hetdex_sc1_spec_vX.fits

   One row per source observation. The table provides basic coordinates/redshift/source information for each observation of a unique astronomical source. The larger file hetdex_sc1_spec_vX.fits contains the same info from the first table plus addition data units of spectral array data.

2. The Detection Information Table: hetdex_sc1_detinfo_vX.fits
   One row per line or continuum detection. Bright sources can be comprised of multiple line or continuum emission. This catalog provides specific detection information such as line parameter info (S/N, line flux, line width)

How to Open the Source Observation Table without Spectra

Multiple formats are provided for the source table. As suggested by Astropy v5.1 developers, we provide the recommended .ecsv format which contains masking and data quantity info. We also provide a simple .dat ascii file.

In [ ]:
```
source_table = Table.read(op.join( path_to_scl, 'hetdex_scl_v3.1.ecsv') )
```

How to Open the Source Observation Table with Spectra

If spectral data is desired, a fits file contains spectral array data matching each row of the Source Observation Table

In [ ]:
```
hdu = fits.open( op.join( path_to_scl, 'hetdex_scl_spec_v3.1.fits'))
```

In [ ]:
```
# The source table can also be accessed by astropy Table class or through HDU1
source_table = Table.read( op.join( path_to_scl, 'hetdex_scl_spec_v3.1.fits'))
```

In [ ]:
```
hdu.info()
```

In [ ]:
```
spec = hdu['SPEC'].data
spec_err = hdu['SPEC_ERR'].data
wave_rect = hdu['WAVELENGTH'].data
```

In [ ]:
```
#spec unit is:
u.Unit( hdu['SPEC'].header['BUNIT'])
```

In [ ]:
```
# wavelength unit is:
u.Unit( hdu['WAVELENGTH'].header['BUNIT'])
```

Query by source type example

In [ ]:
```
sel_lae = source_table['source_type'] == 'lae'
print('There are {} LAES in the catalog'.format(np.sum(sel_lae)))
```

In [ ]:
```
source_ids_lae = source_table['source_id'][sel_lae]
```

In [ ]:
```
# Plot of all LAE spectra in redshift order as an example
lae_spec_table = hstack( [source_table[sel_lae], spec[sel_lae], spec_err[sel_lae]])
```

In [ ]:
```
lae_spec_table.rename_column('col0_2', 'spec')
lae_spec_table.rename_column('col0_3', 'spec_err')
```

In [ ]:
```
lae_spec_table.sort('z_hetdex')
```

In [ ]:
```
plt.figure(figsize=(4, 10))
plt.imshow( lae_spec_table['spec'], aspect=0.05, vmin=-0.01, vmax=2)
plt.title('HETDEX LAE spectra in increasing redshift order')
plt.xlabel('wavelength/z_hetdex')
plt.ylabel('HETDEX 1D LAE spectra')
```

Query by Sky Coordinate Example

In [ ]:
```
# Create array of coordinates for all HETDEX source members
source_coords = SkyCoord(ra = source_table['RA'], dec= source_table['DEC'])
```

In [ ]:
```
coord = SkyCoord(ra=220.21432*u.deg, dec=52.095898*u.deg)
```

In [ ]:
```
sel_match = source_coords.separation(coord) < 1.*u.arcsec
```

Access and Plot Spectra

In [ ]:
```
redshift = source_table['z_hetdex'][sel_match][0]

lya_wave = 1216* (redshift + 1)
source_id = source_table['source_id'][sel_match][0]
name = source_table['source_name'][sel_match][0]
plt.plot( wave_rect, spec[sel_match][0])

plt.xlim(lya_wave-50, lya_wave+50)
plt.ylabel(r'spec (10$^-17$ ergs/s/cm$^2$/$\AA$)')
plt.xlabel(r'wavelength ($\AA$)')
plt.title('{}  source_id={}'.format(name, source_id))
```

How to Open the Detection Info Table

In [ ]:
```
det_table = Table.read(op.join(path_to_scl, 'hetdex_scl_detinfo_v3.1.fits'))
```

Plot up all detections for a single source_id

Let's query the detection info table for a relatively nearby, bright galaxy that is composed of serveral line detections to demonstrate the content of the Detection Info Table. Nearby galaxies and bright sources such as AGN can be composed of many detections. This is because line emission at different spatial regions and wavelengths will result in multiple detections in the detection search. The source will also likely have a complementary continuum detetion if it is bright (g < 21)

In [ ]:
```
sel_big_oii = (det_table['source_type'] == 'oii') & (det_table['major'] > 6)
sel_center_ifu = (np.abs( det_table['x_ifu']) < 5) & (np.abs(det_table['y_ifu']) < 5)
selected_det = (det_table['selected_det'] == True) & (det_table['n_members'] >6)
```

In [ ]:
```
sel = sel_big_oii & sel_center_ifu & selected_det

print(' There are {} matches '.format(np.sum(sel)))
```

In [ ]:
```
# Pick random object in list and plot up the source_id

index = np.where(sel)[0][0]

sid = det_table['source_id'][index]

coords = SkyCoord(ra=det_table['RA'][index]*u.deg, dec=det_table['DEC'][index]*u.deg)
```

In [ ]:
```
# Get Imaging data from Legacy Survey API
fits_file = 'https://www.legacysurvey.org/viewer/fits-cutout?ra={}&dec={}&layer=ls-dr9&width=80&height=80&pixscale=0.2
hdu_ls = fits.open(fits_file)
wcs_ls = WCS( hdu_ls[0].header).dropaxis(2)
```

In [ ]:
```
redshift = det_table['z_hetdex'][index]
source_type = det_table['source_type'][index]

grp = det_table[ det_table['source_id'] == sid]
grp.sort('gmag')
```

In [ ]:
```
# Get Spectrum from source_table
hdu = fits.open('hetdex_source_catalog_1/hetdex_scl_spec_v3.1.fits')
source_table = hdu['INFO'].data
spec = hdu['SPEC'].data
sel_source = source_table['source_id'] == sid
spectra = spec[sel_source][0]
```

In [ ]:
```
plt.figure(figsize=(13,3))
gs = gridspec.GridSpec(1, 2, width_ratios=[1,3])

ax1 = plt.subplot(gs[0], projection=wcs_ls)
im_zscale = ZScaleInterval(contrast=0.5, krej=1.1)
im_vmin, im_vmax = im_zscale.get_limits(values=hdu_ls[0].data[2])
plt.imshow(hdu_ls[0].data[2], origin='lower', cmap=plt.get_cmap('gray_r'), vmin=im_vmin, vmax=im_vmax  )

sel_line = (grp["det_type"] == "line")
if np.sum(sel_line) >= 1:
    plt.scatter(
        grp["RA_det"][sel_line],
        grp["DEC_det"][sel_line],
        transform=ax1.get_transform("world"),
        marker="x",
        color="orange",
        linewidth=2,
        s=50,
#        zorder=100,
        label="line emission",
    )

sel_cont = grp["det_type"] == "cont"
if np.sum(sel_cont) >= 1:
    plt.scatter(
        grp["RA_det"][sel_cont],
        grp["DEC_det"][sel_cont],
        transform=ax1.get_transform("world"),
        marker="x",
        color="green",
        linewidth=2,
        s=50,
        label="continuum",
    )
lon = ax1.coords[0]
lat = ax1.coords[1]
lon.set_axislabel('RA', minpad=0.5)
lat.set_axislabel('Dec', minpad=-0.6)
lon.set_ticklabel(exclude_overlapping=True)

ax2 = plt.subplot(gs[1])

plt.plot(wave_rect, spectra, linewidth=1.2, color='tab:blue')#, yerr=spec_table['specId_err'])
plt.xlabel(r'$\lambda$ ($\AA$)')
plt.ylabel('f$_\lambda$ (10$^{-17}$ ergs/s/cm$^2$/$\AA$)')
plt.xlim(3540, 5450)

selw = (wave_rect > 3540) & (wave_rect < 5450)
y2 = np.max(spectra[selw])
y1 = np.min(spectra[selw])

if y1 > 0:
    y1 = 0

# plot all emission lines detected in det_table related to the source_id

for line in np.array( np.unique(grp['line_id'])):
    if line == 'null':
        continue

    sel_line = grp['line_id'] == line

    plt.bar(grp['wave'][sel_line][0], height=2*y2, width=30, bottom=y1, color='orange', alpha=0.3)

    label = '{}  {{}}'.format( grp['detectid'][sel_line][0], line.decode())
    if np.isfinite( grp['wave'][sel_line][0]):
        plt.text(grp['wave'][sel_line][0]-70, 0.1*y2, label, rotation=90, fontsize=10)

plt.axhline(0, color='tab:grey', linestyle='dashed')
plt.text(0.05, 0.7, 'z=${:6.4f}'.format(redshift), transform=ax2.transAxes, color='tab:red', fontsize=20)
plt.text(0.05, 0.85, 'source_id={}'.format(sid), transform=ax2.transAxes, fontsize=14, color='black')
plt.ylim(y1,1.1*y2)
plt.subplots_adjust(wspace=0.3, hspace=0)
plt.tight_layout()
```

References

- Dey, A., Schlegel, D.J., Lang, D., et al. 2019, Astrophysical Journal, 157, 168. doi:10.3847/1538-3881/ab089d LEGACY SURVEY: we use the LS API to obtain a sky cutout at the HETDEX source https://www.legacysurvey.org.
- Gebhardt, K., Mentuch Cooper, E., Ciardullo, R., et al.2021, Astrophysical Journal, 923, 217. doi:10.3847/1538-4357/ac2e03