Lab1

April 10, 2024

[1]:
```
import warnings
import os
display(os.getcwd())
```

```
'/home/jovyan'
```

[2]:
```
import numpy as np
import pandas as pd
import matplotlib as plt

import astropy.units as u
import astropy.coordinates as coord
from astropy.coordinates import get_body
from astropy.time import Time

from astroplan.plots import plot_finder_image, plot_airmass
from astroplan import Observer, FixedTarget, time_grid_from_range

from astroquery.simbad import Simbad as simbad
```

0.1 Building Functions

(scroll down for lab portion)

1. Establishing location and observing time for Kitt Peak Observatory

[3]:
```
def calc_observation(date, location):
    loc_noon = location.noon(date, which = "nearest")
    loc_midnight = location.midnight(loc_noon, which='next')

    obs_night = location.tonight(loc_noon, horizon = -18*u.deg)
    obs_length = (obs_night[1] - obs_night[0]).to(u.h)
    obs_window = time_grid_from_range(obs_night, time_resolution = 0.1*u.h)

    print(f"Observation Window at {location.name}:",
        f"\n  {obs_length:.1f} of observation time between",
        f"\n  {obs_night[0].strftime('%b %d, %Y %H:%M')} and {obs_night[1].
 ↪strftime('%b %d, %Y %H:%M')} UTC or",
```

```
        f"\n  {obs_night[0].to_datetime(location.timezone).strftime('%b %d, %Y␣
↪%H:%M')} and {obs_night[1].to_datetime(location.timezone).strftime('%b %d,␣
↪%Y %H:%M')} local time ({kpo.timezone})")
    print('')
    print(f"Potential Impacts of the Moon:",
          f"\n  Illumination: {round(location.moon_illumination(loc_midnight) *␣
↪100, 2)}%",
          f"\n  Moon Phase: {(location.moon_phase(loc_noon).to(u.deg)):.2f}",
          f"\n  Moon Rise Time: {location.moon_rise_time(loc_midnight,␣
↪which='nearest').to_value('iso', 'date_hm')} {loc_midnight.scale}",
          f"\n  Moon Set Time: {location.moon_set_time(loc_midnight,␣
↪which='nearest').to_value('iso', 'date_hm')} {loc_midnight.scale}")
    print('')
    print("Now returning loc_noon, loc_midnight, obs_night, obs_length, and␣
↪obs_window.")

    return loc_noon, loc_midnight, obs_night, obs_length, obs_window
```

2. Filtering out objects to narrow down the observation list

[4]:
```
def filter_airmass(list, location, loc_midnight):
    targets_coord = []
    targets_name = []

    for object in list:
        try:
            object_coords = FixedTarget.from_name(object)
            airmass = location.altaz(loc_midnight, object_coords).secz

            if abs(airmass) < 1.5:
                targets_coord.append(object_coords)
                targets_name.append(object)

        except Exception as e:
            print(f"{object} was skipped: {e}. \nContinuing with next iteration.
↪")
            continue
    return targets_coord, targets_name
```

[5]:
```
def filter_visible(location, coordinate_list, window, name_list):
    count_index = 0
    cnt_true = 0
    time_seen = []
    names_visible = []
    names_limited = []

    for object in coordinate_list:
```

```
        for time in window: # goes through every time for one object at a time
            if location.target_is_up(time, coordinate_list[count_index]) ==␣
↪True:
                cnt_true = cnt_true + 1
                time_seen.append( Time(time, format = 'ymdhms') )

        if cnt_true == len(window):
            names_visible.append(name_list[count_index])
        elif len(time_seen) != 0:
            names_limited.append(name_list[count_index])
            print(f"{object.name} will only be visible between {min(time_seen).
↪to_datetime(location.timezone).strftime('%H:%M')} and",
                  f"{max(time_seen).to_datetime(location.timezone).strftime('%H:
↪%M')} {window.scale}.")

        # preparing for next object
        count_index = count_index + 1
        cnt_true = 0
        time_seen = []

    return names_visible, names_limited
```

3. Generating images for remaining interacting galaxies Commenting out a test with one object

[6]:
```
# object = coord.SkyCoord.from_name(data[0]['MAIN_ID'])

# fig, ax = plt.pyplot.subplots(
#     figsize = (2, 2),
#     constrained_layout = True
# )

# ax, hdu = plot_finder_image(
#     object,
#     fov_radius= 0.055 * u.deg,
#     ax = ax,
#     survey = 'DSS2 Blue'
# )

# ax.imshow(hdu.data,
#           cmap = plt.cm.magma,
#           origin = "lower", # sets origin to lower left corner
#           vmin = hdu.data.mean(),
#           vmax = np.max(hdu.data)
#          )
```

[7]:
```
def show_objects(nrows, ncols, figsize, names_list, *, cmap = plt.cm.magma):
    counter = 0

    fig, axs = plt.pyplot.subplots(nrows=nrows, ncols=ncols, figsize=figsize,
                           subplot_kw={'xticks': [], 'yticks': []})

    for ax, name in zip(axs.flat, names_list):
        object = coord.SkyCoord.from_name(name)

        ax, hdu = plot_finder_image(
            object,
            fov_radius= 0.04 * u.deg,
            ax = ax,
            survey = 'DSS2 Blue'
        )

        ax.imshow(hdu.data,
                  cmap = cmap,
                  vmin = hdu.data.mean(),
                  vmax = np.max(hdu.data),
                  aspect='auto'
                 )

        # x-axis is RA, y-axis is Dec -- just setting a bottom label
        ax.set_xlabel(f"RA: {np.ma.getdata(data[[counter]]['RA'])[0]}      Dec:␣
↪{np.ma.getdata(data[[counter]]['DEC'])[0]}")
        ax.set_ylabel('')
        ax.set_title(f"#{counter+1}: {name}")
        counter = counter + 1

    #display(data['MAIN_ID', 'RA', 'DEC'].to_pandas().query('MAIN_ID in␣
↪@names_list'))
    plt.pyplot.tight_layout(pad=0.03, h_pad=1.2, w_pad=0.3, rect=(0, 0, 0, 0))
    plt.pyplot.show()
```

0.2 Lab 1 - Planning Observations

Select a particular kind of astronomical object (some possibilities, not exhaustive - quasars, galaxies, binary stars, supernovae, globular clusters, planetary nebulae). Find
catalog that contains a fair number of these objects. When you have found it, use t e rubric on Canvas to identify your observatory window.

0.2.1 Write the object type and the catalog you'll be using

I will be using interacting galaxy objects from the Simbad database. Gathering the first 100 objects from Simbad query:

[8]:
```
data = simbad.query_criteria(maintypes='IG')[0:100]
data[:5]
```

[8]:
```
<Table length=5>
 MAIN_ID         RA           DEC       …     COO_BIBCODE     SCRIPT_NUMBER_ID
              "h:m:s"       "d:m:s"     …
  object       str13         str13      …        object            int32
---------- ------------ ------------ … ------------------- ----------------
 ESO 193-7    00 00 24.1     -49 04 23 … 1989ESOLV.C…0L                 0
 ESO 118-4 04 08 19.0495 -61 16 05.539 … 2020yCat.1350…0G                 0
ESO 113-51 01 31 07.5396 -60 29 37.357 … 2020yCat.1350…0G                 0
ESO 337-13    19 11 07.9     -38 39 42 … 1982ESO…C…0L                   0
 ESO 360-6    04 19 41.0     -36 26 51 … 1982ESO…C…0L                   0
```

Write your observatory assignment & observing window. What will the moon phase be? Will the moon interfere with your observations? Explain your reasoning

Selecting Kitt Peak Observatory for an observation run on May 5th, 2024

[9]:
```
date1 = Time("2024-05-05", format='iso', out_subfmt='date_hm')

kpo = Observer(latitude = (31 * u.deg) + (57 * u.arcmin) + (30.40 * u.arcsec),
                       longitude = (-111 * u.deg) + (35 * u.arcmin) + (40.90 * u.
 ↪arcsec),
                       timezone = 'America/Los_Angeles',
                       name = "Kitt Peak Observatory"
                      )
```

[10]:
```
run1 = calc_observation(date1, kpo)
```

```
Observation Window at Kitt Peak Observatory:
  7.4 h of observation time between
  May 05, 2024 03:35 and May 05, 2024 11:00 UTC or
  May 04, 2024 20:35 and May 05, 2024 04:00 local time (America/Los_Angeles)

Potential Impacts of the Moon:
  Illumination: 11.09%
  Moon Phase: 134.24 deg
  Moon Rise Time: 2024-05-05 10:56 utc
  Moon Set Time: 2024-05-04 22:27 utc

Now returning loc_noon, loc_midnight, obs_night, obs_length, and obs_window.
```

The moon should not have much impact on the observing window - it will only be about 11% illuminated, as it is almost close to being in the new moon phase. Astroplan indicates phases by degrees between 0° and 180°(or), where 0° represents a full moon and 180° represents a new moon. The moon phase is calculated to be 134.24 deg and will have less impacts on the observation window.

Using more than 20 objects (and less than 100) - determine which of your objects will be visible during your observing window.

Upload your code & list to Canvas, and write an explanation of your process (If you have more than 10 objects that are visible, pick ten for the rest of the assignment). Your object list should have at least name, RA, and DEC for each object.

Filtering through objects from the Simbad query:

[11]:
```
pass_one = filter_airmass(list(data['MAIN_ID']), kpo, run1[1])

print(len(pass_one[0]), len(pass_one[1])) # verifying filter worked and that␣
 ↪the lists of names & coordinates are lined up
```

```
[RSG99b] 06009-7716 E was skipped: Unable to find coordinates for name '[RSG99b]
06009-7716 E' using https://cds.unistra.fr/cgi-bin/nph-
sesame/A?%5BRSG99b%5D%2006009-7716%20E.
Continuing with next iteration.
47 47
```

[12]:
```
pass_two = filter_visible(kpo, pass_one[0], run1[4], pass_one[1])
pass_two[0]
```

[12]:
```
['APG 171',
 'VV  747',
 'VV  774',
 'APG  32',
 '[EAD2001] HDFN J123652.88+621404.8',
 'TKRS  8710',
 'APG 277',
 'MCG+03-40-057']
```

Double checking for moon interference:

[13]:
```
moon_kpo = get_body('moon', date1)

for name in pass_two[0]:
    display(name, coord.SkyCoord.from_name(name).separation(moon_kpo))
```

```
'APG 171'
```

$19^\circ47'38.13775828''$

```
'VV  747'
```

$75^\circ36'33.13169724''$

```
'VV  774'
```

$48^\circ03'26.79874856''$

```
'APG  32'
```

81°03′20.43117047″

'[EAD2001] HDFN J123652.88+621404.8'

82°37′23.89716751″

'TKRS 8710'

82°35′29.54478834″

'APG 277'

33°51′03.02422525″

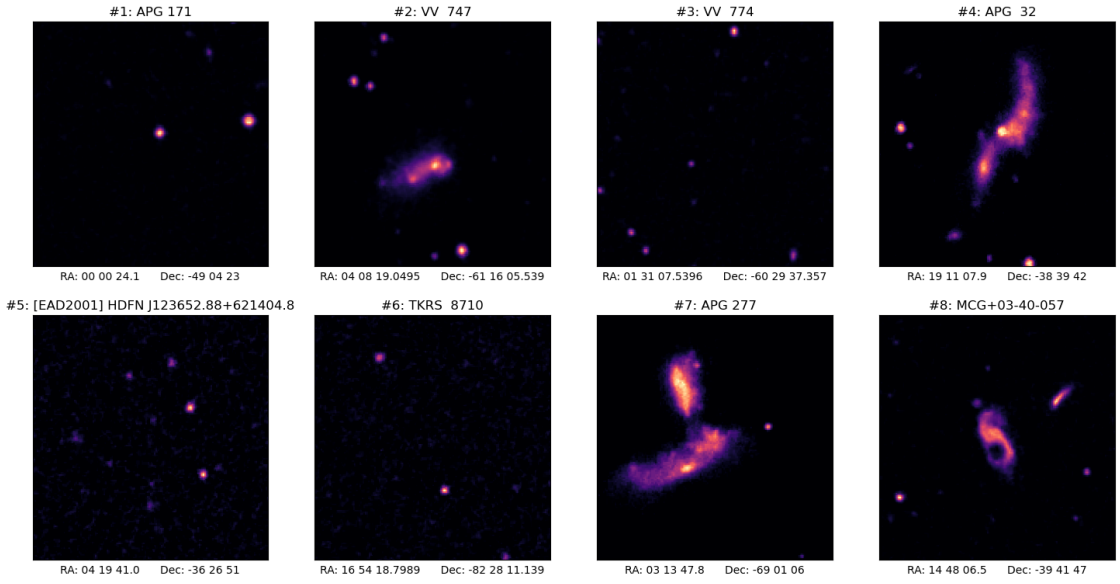
'MCG+03-40-057'

38°19′04.84678895″

Rendering images of visible objects during this observing window:

[14]:
```
show_objects(
    nrows=2, ncols=4,
    figsize = (18, 9),
    names_list = pass_two[0])
```

```
/tmp/ipykernel_285/3533517207.py:31: UserWarning: Tight layout not applied. The
left and right margins cannot be made large enough to accommodate all axes
decorations.
  plt.pyplot.tight_layout(pad=0.03, h_pad=1.2, w_pad=0.3, rect=(0, 0, 0, 0))
```



0.2.2 Observing 7 weeks later

Your observing run gets clouded out (a tragically common occurance) but you're able to get director's discretionary time 7 weeks later. Using the same objects you planned to observe initially - will it be better or worse for your same ten objects? Explain

[15]:
```
date2 = Time("2024-06-23", format='iso', out_subfmt='date_hm')
run2 = calc_observation(date2, kpo)
```

```
Observation Window at Kitt Peak Observatory:
  6.4 h of observation time between
  Jun 23, 2024 04:11 and Jun 23, 2024 10:36 UTC or
  Jun 22, 2024 21:11 and Jun 23, 2024 03:36 local time (America/Los_Angeles)

Potential Impacts of the Moon:
  Illumination: 97.86%
  Moon Phase: 10.87 deg
  Moon Rise Time: 2024-06-23 03:49 utc
  Moon Set Time: 2024-06-23 13:33 utc

Now returning loc_noon, loc_midnight, obs_night, obs_length, and obs_window.
```

[16]:
```
pass_one_later = filter_airmass(list(data['MAIN_ID']), kpo, run2[1])
len(pass_one_later[0]), len(pass_one_later[1])
```

```
[RSG99b] 06009-7716 E was skipped: Unable to find coordinates for name '[RSG99b]
06009-7716 E' using https://cds.unistra.fr/cgi-bin/nph-
sesame/A?%5BRSG99b%5D%2006009-7716%20E.
Continuing with next iteration.
```

[16]: (33, 33)

[17]:
```
pass_two_later = filter_visible(kpo, pass_one_later[0], run2[4],␣
 ↪pass_one_later[1])
pass_two_later[0]
```

[17]: ['VV 774', 'APG 32', 'MCG+03-40-057']

[18]:
```
# double checking for moon interference:
moon_kpo_later = get_body('moon', date2)

for name in pass_two_later[0]:
    display(name, coord.SkyCoord.from_name(name).separation(moon_kpo_later))
```

```
'VV  774'
```

$57°38'03.43656947''$

```
'APG  32'
```

$83^\circ 28' 06.18885006''$

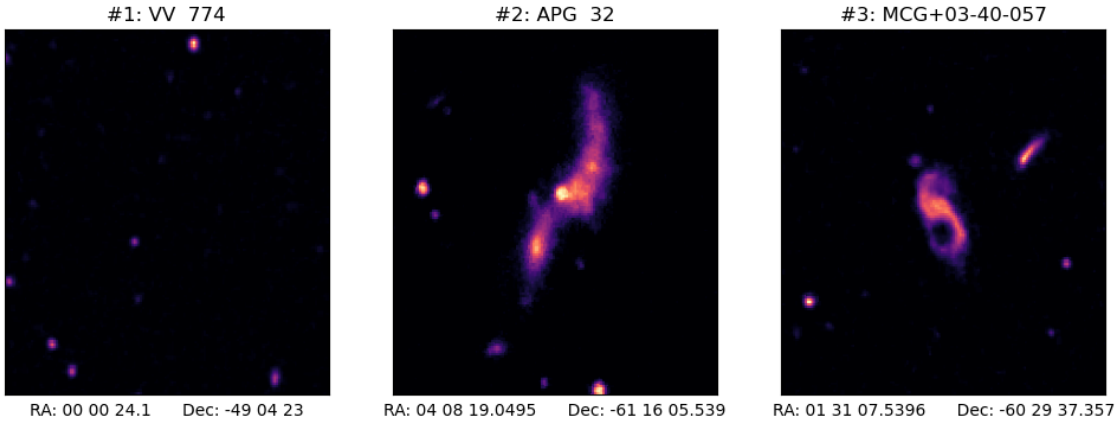
'MCG+03-40-057'

$52^\circ 59' 08.94603295''$

Objects that will be seen during the whole observing window:

[29]:
```
show_objects(
    nrows=1, ncols=3,
    figsize=(12, 4),
    names_list=pass_two_later[0])
```

```
/tmp/ipykernel_285/3533517207.py:31: UserWarning: Tight layout not applied. The
left and right margins cannot be made large enough to accommodate all axes
decorations.
  plt.pyplot.tight_layout(pad=0.03, h_pad=1.2, w_pad=0.3, rect=(0, 0, 0, 0))
```



Comparing objects from May 5th, 2024 and June 23rd, 2024

[22]:
```
set(pass_two[0]) & set(pass_two_later[0])
```

[22]:
```
{'APG  32', 'MCG+03-40-057', 'VV  774'}
```

For the 8 objects that are visible during the observation window at Kitt Peak Observatory on May 5th, 2024, APG 32, MCG+03-40-057, and VV 774 are the only objects that will also be visible during the observing window on June 23rd, 2024. These overlapping objects from May 5th, 2024 will also be visible during the entire observation window on June 23rd, 2024. So, for the interactiing galaxies that are visible from KPO on May 5th, 2024, it's a little worse since only 3/8 objects will be still be visible 7 weeks later.

[ ]:
```

```