Final Project

March 20, 2021

[67]:
```
# Making a function that switches independent variables for plots on Juptyer␣
↪notebooks
```

[68]:
```
# Example with simple data

import matplotlib.pyplot as plt
import numpy as np
from ipywidgets import interact, fixed

my_x1 = np.array([1, 2, 3, 4])
my_y1 = np.array([9, 10, 11, 12])
my_y2 = np.array([8, 9, 10, 11])

y_lib = [my_y1, my_y2]

def make_plot(my_x, y_lib, index):

    fig,ax = plt.subplots(1,1)
    ax.set_xlabel('my_x')
    ax.set_ylabel('my_y')
    ax.plot(my_x, y_lib[index])

def interactive_plot():
    interact(make_plot, my_x=fixed(my_x1), y_lib=fixed(y_lib), index=(0,1))
```

[69]:
```
interactive_plot()
```

```
interactive(children=(IntSlider(value=0, description='index', max=1), Output()), _dom_classes=
```

[70]:
```
# Example with astronomical data below
```

[71]:
```
import pandas as pd
import sqlite3
```

[72]:
```
import os
try:
```

```
    os.remove("msdss.db")
except OSError:
    pass
```

[73]:
```
con = sqlite3.connect("msdss.db")
```

[74]:
```
con.execute("""
CREATE TABLE `sources` (
    `run`      INTEGER,
    `rerun`    INTEGER,
    `camcol`   INTEGER,
    `field`    INTEGER,
    `obj`      INTEGER,
    `type`     INTEGER,

    `ra`          REAL,
    `dec`         REAL,
    `psfMag_r`    REAL,
    `psfMag_g`    REAL,
    `psfMagErr_r` REAL,
    `psfMagErr_g` REAL
);
""")
```

[74]:
```
<sqlite3.Cursor at 0x7f5bfcb78a40>
```

[75]:
```
con.execute("""
CREATE TABLE `runs` (
    `run`         INTEGER,
    `ra`          REAL,
    `dec`         REAL,
    `mjdstart`    REAL,
    `mjdend`      REAL,
    `node`        REAL,
    `inclination` REAL,
    `mu0`         REAL,
    `nu0`         REAL
);
""")
```

[75]:
```
<sqlite3.Cursor at 0x7f5bfbed3c00>
```

[76]:
```
runs = pd.read_csv('runs.txt',
            sep=" ", header=None, skiprows=1,
            names=['run', 'ra', 'dec', 'mjdstart', 'mjdend', 'node',␣
 ↪'inclination', 'mu0', 'nu0'],
            index_col = 'run')
```

[77]:
```
sources = pd.read_csv('sample.csv',
                      dtype={
                              'run': np.int16,
                              'rerun': np.int16,
                              'camcol': np.int8,
                              'field': np.int16,
                              'obj':  np.int32,
                              'type': np.int16,
                              'psfMag_r': np.float32,
                              'psfMag_g': np.float32,
                              'psfMagErr_r': np.float32,
                              'psfMagErr_g': np.float32,
                            },
                      index_col=['run', 'rerun', 'camcol', 'field', 'obj'],
                      na_values={
                           'psfMagErr_g': ["-9999"],
                           'psfMagErr_r': ["-9999"],
                           'psfMag_g': ["-9999"],
                           'psfMag_r': ["-9999"],
                      },
                      verbose=True
                     )
```

```
Tokenization took: 31.23 ms
Type conversion took: 27.32 ms
Parser memory cleanup took: 0.01 ms
```

[78]:
```
runs.to_sql('runs', con, if_exists='replace')
```

[79]:
```
sources.to_sql('sources', con, if_exists='replace')
```

[80]:
```
result = pd.read_sql("""
    SELECT
        sources.ra, sources.dec, sources.run, mjdstart, psfMag_r, psfMag_g
    FROM
        sources JOIN runs ON sources.run = runs.run
""", con)
```

[81]:
```
result[:5]
```

[81]:

	ra	dec	run	mjdstart	psfMag_r	psfMag_g
0	8.129444	26.626617	7757	54764.323971	17.048889	18.165350
1	8.127839	26.627246	7757	54764.323971	17.374020	17.928749
2	8.127323	26.625120	7757	54764.323971	20.146601	21.352970
3	24.516117	-1.165794	4288	52971.187293	22.970320	24.325899
4	24.517941	-1.179207	4288	52971.187293	22.620520	25.091089

[82]:
```
# Picking out Andromeda's coordinates of approximately 10.6 ra and 41 dec

result_of_interest = result.query('ra < 11').query('ra > 10.5').query('dec >␣
↪40').query('dec < 42')

my_x1 = result_of_interest['ra']
my_y1 = result_of_interest['psfMag_r']
my_y2 = result_of_interest['psfMag_g']

y_lib = [my_y1, my_y2]

def make_plot(my_x, y_lib, index):

    fig,ax = plt.subplots(1,1)
    ax.set_xlabel('my_x')
    ax.set_ylabel('my_y')
    ax.plot(my_x, y_lib[index])

def interactive_plot():
    interact(make_plot, my_x=fixed(my_x1), y_lib=fixed(y_lib), index=(0,1))
```

[83]:
```
# This plots the r and g bands of Andromeda

interactive_plot()
```

```
interactive(children=(IntSlider(value=0, description='index', max=1), Output()), _dom_classes=
```

[ ]:
```

```