# TRY

November 23, 2023

[14]:
```
# Import Python libraries to work with SciServer
import SciServer.CasJobs as CasJobs # query with CasJobs
import SciServer.SciDrive as SciDrive   # read/write to/from SciDrive
import SciServer.SkyServer as SkyServer   # show individual objects and␣
 ↪generate thumbnail images through SkyServer
print('SciServer libraries imported')
```

```
SciServer libraries imported
```

[15]:
```
# Import other libraries for use in this notebook.
import numpy as np                    # standard Python lib for math ops
#from scipy.misc import imsave         # save images as files
import pandas                         # data manipulation package
import matplotlib.pyplot as plt       # another graphing package
import os                             # manage local files in your Compute␣
 ↪containers
print('Supporting libraries imported')
```

```
Supporting libraries imported
```

[17]:
```
#import astroML
#from astroML.datasets import fetch_sdss_spectrum
from astropy.io import ascii
# Apply some special settings to the imported libraries
# ensure columns get written completely in notebook
pandas.set_option('display.max_colwidth', -1)
# do *not* show python warnings
import warnings
warnings.filterwarnings('ignore')
print('Settings applied')
```

```
Settings applied
```

[18]:
```
# Find objects in the Sloan Digital Sky Survey's Data Release 14.
#
# Query the Sloan Digital Sky Serveys' Data Release 14.
# For the database schema and documentation see http://skyserver.sdss.org/dr14
#
```

```
# This query finds all galaxies with a size (petror90_r) greater than 10␣
 ↪arcseconds, within
# a region of sky with 100 < RA < 250, a redshift between 0.02 and 0.5, and a␣
 ↪g-band magnitude brighter than 17.
#
# First, store the query in an object called "query"
query="""
SELECT p.objId,p.ra,p.dec,p.petror90_r, p.expAB_r,
    p.dered_u as u, p.dered_g as g, p.dered_r as r, p.dered_i as i,
    s.z, s.plate, s.mjd, s.fiberid
FROM galaxy AS p
   JOIN SpecObj AS s ON s.bestobjid = p.objid
WHERE p.petror90_r > 10
  and p.ra between 100 and 250
  and s.z between 0.02 and 0.5
  and p.g < 17
"""
#Then, query the database. The answer is a table that is being returned to a␣
 ↪dataframe that we've named all_gals.
all_gals = CasJobs.executeQuery(query, "dr16")

print("SQL query finished.")
print("SQL query returned " + str(len(all_gals))+ " galaxies")
```

```
SQL query finished.
SQL query returned 40332 galaxies
```

[19]:
```
all_gals[0:30]
```

[19]:

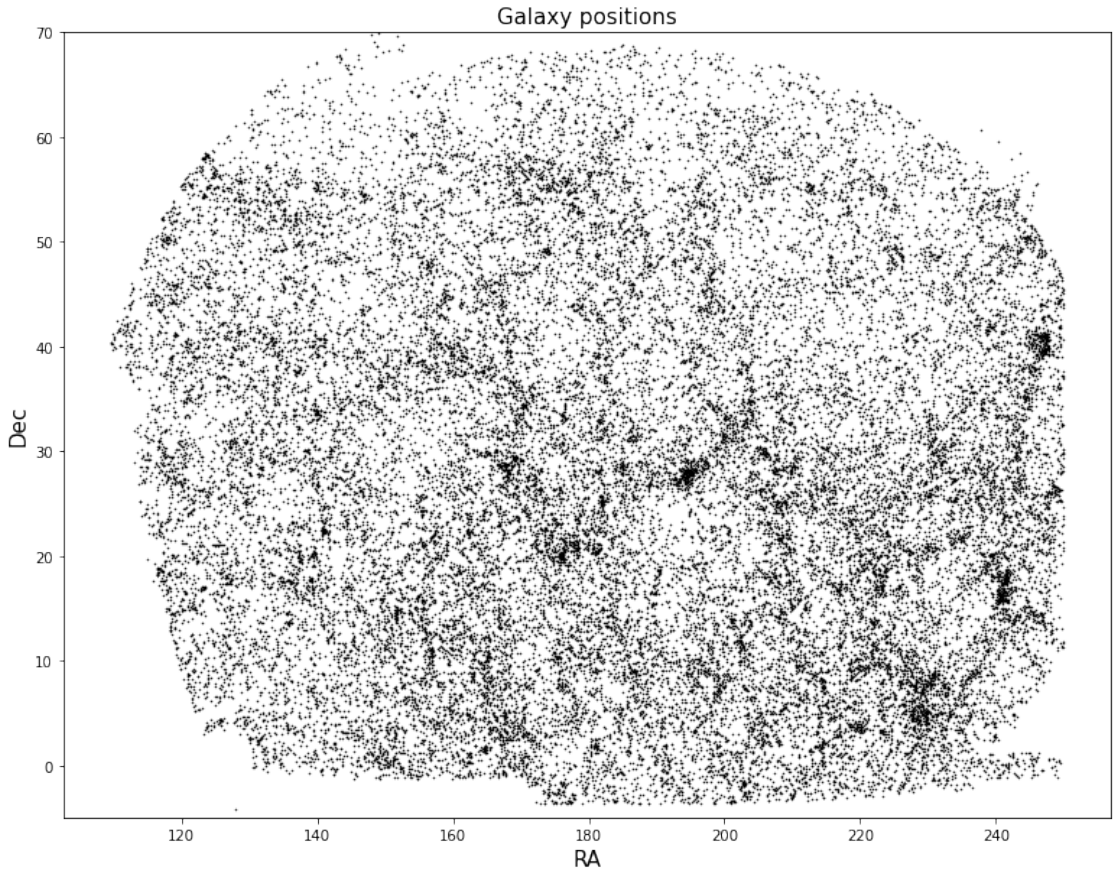
| | objId | ra | dec | petror90_r | expAB_r | \ |
|---|---|---|---|---|---|---|
| 0 | 1237651271892598827 | 145.338782 | 60.735032 | 19.27053 | 0.635783 | |
| 1 | 1237651271895285893 | 156.799361 | 63.968893 | 10.20452 | 0.732676 | |
| 2 | 1237651271895744566 | 158.787538 | 64.450701 | 11.36502 | 0.648912 | |
| 3 | 1237651271896072301 | 160.345644 | 64.653448 | 10.92565 | 0.610096 | |
| 4 | 1237651271897383030 | 167.043811 | 65.693755 | 19.93624 | 0.765442 | |
| 5 | 1237651271897514134 | 167.954399 | 65.815570 | 17.52676 | 0.614216 | |
| 6 | 1237651271897907310 | 169.917819 | 66.075611 | 11.37661 | 0.462223 | |
| 7 | 1237651271897907324 | 169.961092 | 66.070904 | 10.31823 | 0.664021 | |
| 8 | 1237654381446234322 | 155.809017 | 58.457353 | 11.87040 | 0.388531 | |
| 9 | 1237654381975437353 | 131.554084 | 48.348341 | 20.29563 | 0.710043 | |
| 10 | 1237654381975437593 | 131.507598 | 48.470443 | 12.89942 | 0.299922 | |
| 11 | 1237654381975437602 | 131.662793 | 48.374934 | 18.51303 | 0.724634 | |
| 12 | 1237654381975502873 | 131.638715 | 48.430073 | 19.63290 | 0.669211 | |
| 13 | 1237654381975503032 | 131.729455 | 48.422900 | 10.63851 | 0.994723 | |
| 14 | 1237654381975699577 | 132.211580 | 48.724168 | 11.17214 | 0.521521 | |
| 15 | 1237654381976027351 | 132.990141 | 49.306481 | 11.43833 | 0.673900 | |
| 16 | 1237654381977141338 | 135.863076 | 51.096415 | 11.20107 | 0.626453 | |

| | | | | | |
|---|---|---|---|---|---|
| 17 | 1237654381977469187 | 136.697117 | 51.608392 | 11.89828 | 0.488463 |
| 18 | 1237654381977469209 | 136.848635 | 51.574646 | 14.10025 | 0.391607 |
| 19 | 1237654381977534696 | 137.055191 | 51.651799 | 23.51641 | 0.447506 |
| 20 | 1237654381977731176 | 137.526620 | 51.911777 | 22.85088 | 0.868926 |
| 21 | 1237654381977796751 | 137.519461 | 52.067097 | 23.35419 | 0.832447 |
| 22 | 1237655108376789156 | 179.701215 | 60.056829 | 17.24273 | 0.435334 |
| 23 | 1237655108377247878 | 181.802712 | 60.030309 | 12.10586 | 0.518533 |
| 24 | 1237655108377772177 | 184.150247 | 60.006401 | 10.09675 | 0.360467 |
| 25 | 1237655108377837653 | 184.298593 | 60.103876 | 13.94069 | 0.867718 |
| 26 | 1237655108377903249 | 184.734979 | 60.168690 | 17.06170 | 0.562913 |
| 27 | 1237655108377968649 | 184.902627 | 60.057061 | 10.35223 | 0.558310 |
| 28 | 1237655108378165420 | 186.033916 | 60.182576 | 10.31417 | 0.565284 |
| 29 | 1237655108381573359 | 201.318760 | 59.079555 | 11.84369 | 0.704868 |

| | u | g | r | i | z | plate | mjd | fiberid |
|---|---|---|---|---|---|---|---|---|
| 0 | 18.01686 | 16.08480 | 15.21523 | 14.77336 | 0.074648 | 5716 | 56684 | 862 |
| 1 | 17.99719 | 16.89227 | 16.48965 | 16.33450 | 0.022344 | 488 | 51914 | 30 |
| 2 | 16.44495 | 15.47714 | 14.99789 | 14.74447 | 0.033555 | 489 | 51930 | 134 |
| 3 | 18.32820 | 16.90256 | 16.08510 | 15.65800 | 0.137548 | 489 | 51930 | 104 |
| 4 | 18.34957 | 16.35210 | 15.32754 | 14.89074 | 0.115611 | 490 | 51929 | 65 |
| 5 | 17.42230 | 15.89093 | 15.13130 | 14.74487 | 0.063745 | 490 | 51929 | 24 |
| 6 | 18.89230 | 16.94878 | 15.95506 | 15.42679 | 0.095305 | 491 | 51942 | 204 |
| 7 | 18.27424 | 16.30876 | 15.33956 | 14.89032 | 0.095846 | 491 | 51942 | 212 |
| 8 | 17.66907 | 15.96745 | 15.19155 | 14.75085 | 0.072617 | 559 | 52316 | 146 |
| 9 | 16.73816 | 14.90793 | 14.11314 | 13.69376 | 0.029618 | 550 | 51959 | 592 |
| 10 | 18.16921 | 16.45805 | 15.62297 | 15.20535 | 0.053055 | 550 | 51959 | 581 |
| 11 | 17.66162 | 15.74069 | 14.87057 | 14.50253 | 0.052678 | 550 | 51959 | 597 |
| 12 | 15.87545 | 14.47837 | 13.76806 | 13.40889 | 0.023865 | 550 | 51959 | 589 |
| 13 | 17.44487 | 15.81565 | 15.05068 | 14.66499 | 0.052921 | 550 | 51959 | 630 |
| 14 | 17.42587 | 16.03502 | 15.42764 | 15.07406 | 0.070412 | 5161 | 55968 | 805 |
| 15 | 17.71090 | 16.19894 | 15.49300 | 15.10967 | 0.050894 | 551 | 51993 | 306 |
| 16 | 18.72479 | 16.75804 | 15.78902 | 15.37666 | 0.102014 | 552 | 51992 | 162 |
| 17 | 17.79036 | 16.49780 | 16.02447 | 15.76369 | 0.029479 | 552 | 51992 | 564 |
| 18 | 17.16956 | 15.69981 | 14.95824 | 14.53885 | 0.055530 | 552 | 51992 | 617 |
| 19 | 16.29472 | 14.55208 | 13.78662 | 13.40837 | 0.029528 | 553 | 51999 | 264 |
| 20 | 16.60127 | 14.70435 | 13.87028 | 13.44532 | 0.038284 | 553 | 51999 | 231 |
| 21 | 17.21063 | 15.32413 | 14.48639 | 14.07425 | 0.037989 | 553 | 51999 | 135 |
| 22 | 17.15590 | 15.26730 | 14.46370 | 14.00029 | 0.044263 | 953 | 52411 | 633 |
| 23 | 17.02157 | 15.47792 | 14.80724 | 14.44958 | 0.042227 | 954 | 52405 | 520 |
| 24 | 18.40992 | 16.68489 | 15.87472 | 15.43631 | 0.059655 | 955 | 52409 | 147 |
| 25 | 16.97662 | 15.75756 | 15.23535 | 14.94297 | 0.044813 | 955 | 52409 | 491 |
| 26 | 17.16056 | 15.43388 | 14.63508 | 14.24345 | 0.043742 | 955 | 52409 | 518 |
| 27 | 17.40172 | 16.23417 | 15.77503 | 15.48377 | 0.043710 | 6969 | 56420 | 946 |
| 28 | 18.18910 | 16.46868 | 15.67857 | 15.31096 | 0.049709 | 6968 | 56443 | 612 |
| 29 | 17.89634 | 15.84634 | 14.95409 | 14.52477 | 0.074905 | 959 | 52411 | 431 |

[9]:
```
#Possible solution
plt.figure(figsize=(13,10))
plt.scatter(all_gals['ra'], all_gals['dec'], marker='.', color='black',s=1)
plt.xlabel('RA', fontsize=15); plt.ylabel('Dec', fontsize=15)
plt.title('Galaxy positions', fontsize=15)
plt.ylim(-5,70)
```

[9]:
```
(-5.0, 70.0)
```




### 0.0.1 Now, make de gif source.

[20]:
```
import imageio #Let us make the gif file.
print('Import successful')
```

```
Import successful
```

[22]:
```
#Possible solution

frames= []
```

```
num_frame =30

for i in range(50):
    plt.figure(figsize=(13,10))
    slice1 = np.where( (all_gals['z'] > i/100) & (all_gals['z'] < (i+2)/100))[0]
    a= i/100
    b= (i+2)/100

    plt.scatter(all_gals.loc[slice1]['ra'], all_gals.loc[slice1]['dec'],␣
 ↪color='green', marker='.', s=20, label= b)
    plt.ylim(-5,70)
    plt.xlabel('RA', fontsize=15); plt.ylabel('Dec', fontsize=15)
    plt.title('Galaxy positions, slice 1', fontsize=15)
    plt.legend()
    filename = f"frame_{i:03d}.png"
    plt.savefig(filename)
    plt.close()

    frames.append(filename)

with imageio.get_writer('grafico_animado.gif', duration=0.5) as writer:
    for filename in frames:
        image = imageio.imread(filename)
        writer.append_data(image)




print('Gif is ready !')
```

```
Gif is ready !
```

## 0.1 Galaxy morphology

Galaxy morphology studies the shapes of galaxies. You will already have some understanding of how local galaxies look like, from your exploration of SDSS imaging in the first Lab session using the SDSS SkyServer Navigate Tool.

Here, we will do a more systematic exploration of how galaxy shapes are related to other properties.

The next cell provides a bit of code that selects 16 **random** galaxies from your dataframe, and shows you their optical images.

[23]:
```
def show_galaxy_images(my_galaxies):
    #plot a random subset of 16 galaxies
    # set thumbnail parameters
    width=200           # image width
    height=200          # height
    pixelsize=0.396     # image scale
```

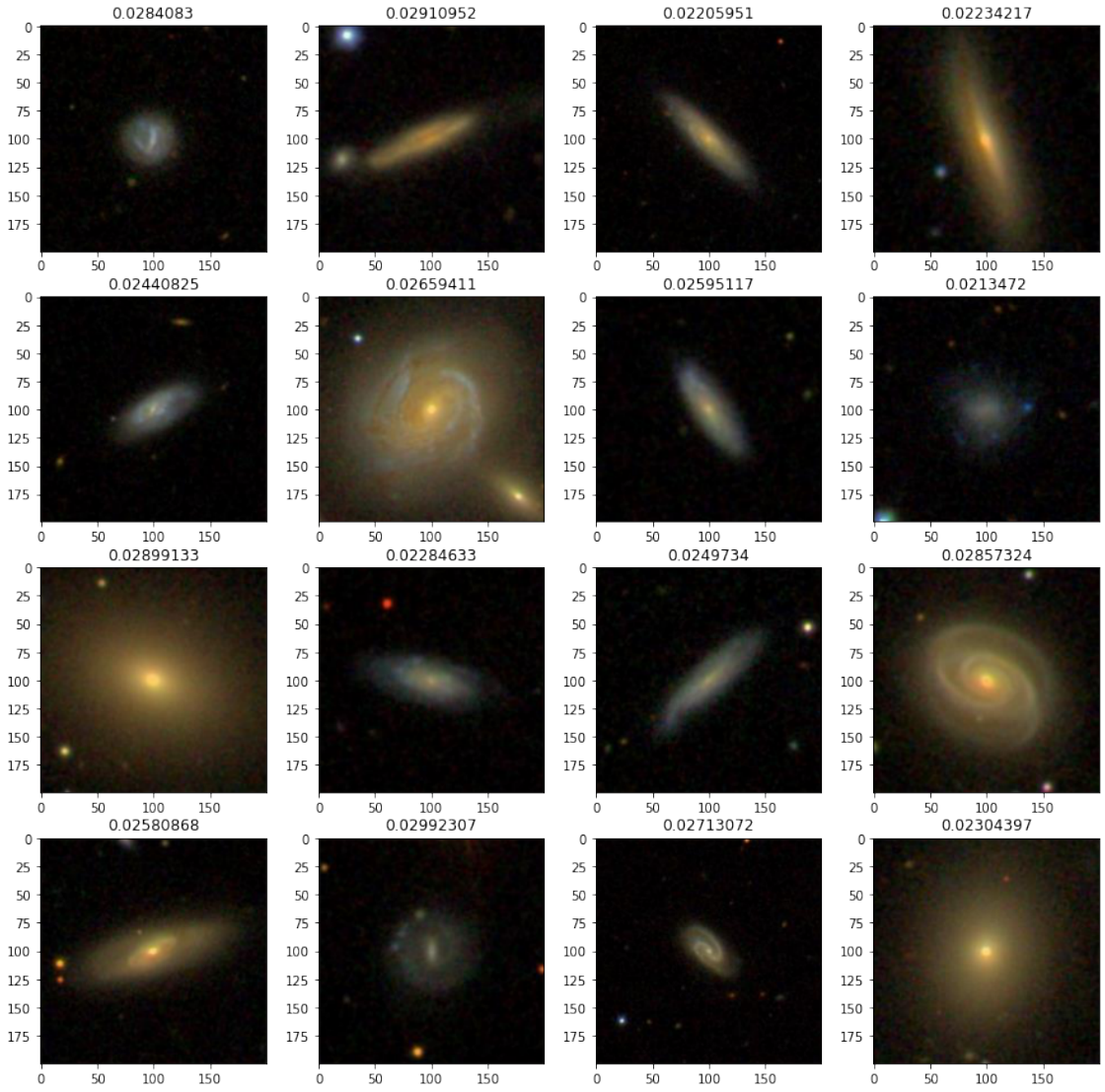
```
    plt.figure(figsize=(15, 15))    # display in a 4x4 grid
    subPlotNum = 1


    i = 0
    nGalaxies = 16 #Total number of galaxies to plot
    ind = np.random.randint(0,len(my_galaxies), nGalaxies) #randomly selected␣
↪rows
    count=0
    for i in ind:           # iterate through the randomly selected rows in the␣
↪DataFrame
        count=count+1
        print('Getting image '+str(count)+' of '+str(nGalaxies)+'...')
        if (count == nGalaxies):
            print('Plotting images...')
        scale=2*all_gals.loc[i]['petror90_r']/pixelsize/width
        img = SkyServer.getJpegImgCutout(ra=all_gals.loc[my_galaxies[i]]['ra'],␣
↪dec=all_gals.loc[my_galaxies[i]]['dec'], width=width, height=height,␣
↪scale=scale,dataRelease='DR14')
        plt.subplot(4,4,subPlotNum)
        subPlotNum += 1
        plt.imshow(img)                                    # show images in grid
        plt.title(all_gals.loc[my_galaxies[i]]['z'])
```

You can use the function defined above to plot 16 *random* galaxies from any dataframe. For example, to plot 16 galaxies randomly selected in a redshift slice $0.02 < z < 0.03$ you might do:

[24]:
```
my_galaxies = np.where( (all_gals['z'] > 0.02) & (all_gals['z'] < 0.03))[0]
show_galaxy_images(my_galaxies)
```

```
Getting image 1 of 16…
Getting image 2 of 16…
Getting image 3 of 16…
Getting image 4 of 16…
Getting image 5 of 16…
Getting image 6 of 16…
Getting image 7 of 16…
Getting image 8 of 16…
Getting image 9 of 16…
Getting image 10 of 16…
Getting image 11 of 16…
Getting image 12 of 16…
Getting image 13 of 16…
Getting image 14 of 16…
Getting image 15 of 16…
Getting image 16 of 16…
Plotting images…
```


0.0284083
0.02910952
0.02205951
0.02234217
0.02440825
0.02659411
0.02595117
0.0213472
0.02899133
0.02284633
0.0249734
0.02857324
0.02580868
0.02992307
0.02713072
0.02304397

[ ]: