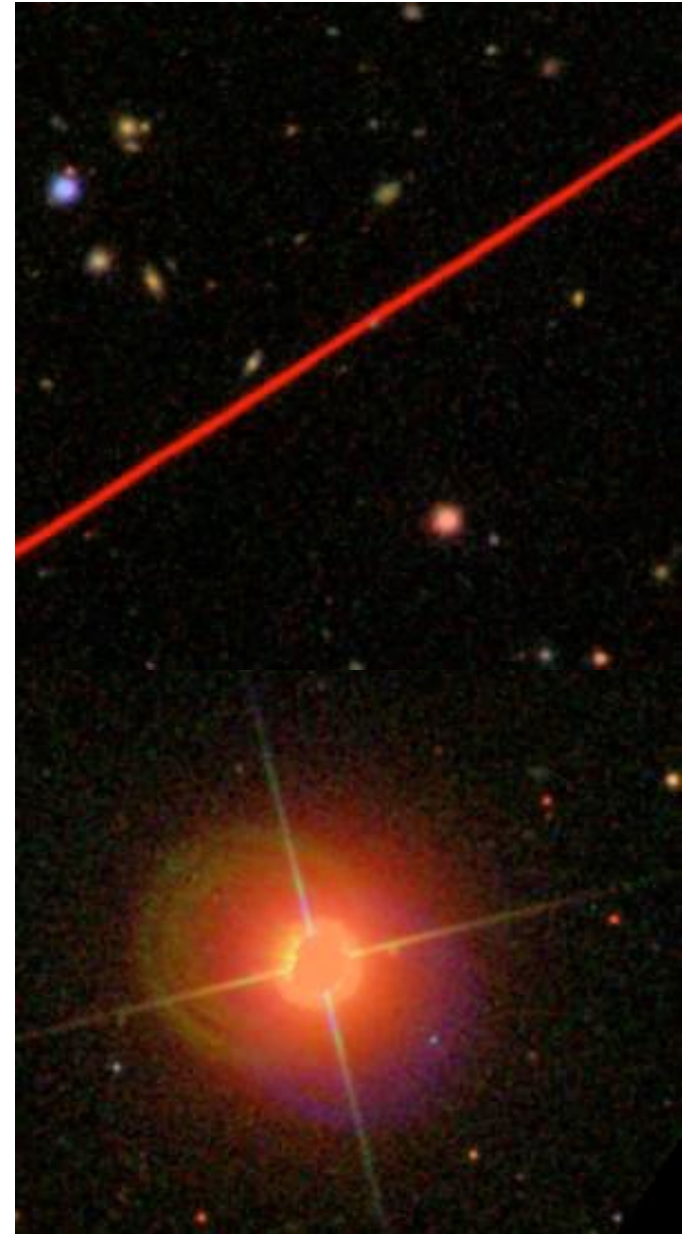


Flagging Bad Data

Flagging Bad Data in Imaging

- Observations are never perfect, due to observing conditions
 - e.g., bad seeing, moonlight, the solar wind, clouds, airplanes, cosmic rays, telescope malfunctions etc.
- Further, even good observations can be rendered unusable by astronomical sources
 - e.g., very bright objects can create diffraction spikes, double or “ghost” images saturated CCD pixels etc.



Flagging Bad Data in Imaging

- As many as half of the observations in the SDSS may be spurious because of these effects
- To attempt to allow users to correct for “bad” imaging data, most surveys have sets of “flags” that can be employed to discard spurious objects
- The common way to express flags is as a *bitmask*
- Say you have a collection of 4 flags that represent, respectively, a saturated pixel (0), a diffraction spike (1), terrible seeing (2) and a ghost image (3). For an object that contains a saturated pixel and ghosting:

$$- \text{flag} = (1 \times 2^0) + (0 \times 2^1) + (0 \times 2^2) + (1 \times 2^3) = 9$$

Flagging Bad Data in Imaging

- In the SDSS sweeps files the flags are recorded in the columns “*OBJC_FLAGS*” and “*OBJC_FLAGS2*”
 - A description of the meanings of the flags is in the file */ASTR8080/runnoe/week11/sdssMaskbits.par* under *maskbits OBJECT1* and *maskbits OBJECT2*
 - Each flag is also described in the *SDSS Schema* (see the syllabus link and search for *photoflags*)
 - note that in the online SDSS catalog archive server, the two sets of flags are combined into one
 - The sweeps files contain flag information both for the imaging combined across *all* filters (as *OBJC_FLAGS* and *OBJC_FLAGS2*) and for the imaging in each *individual* (*ugriz*) band (as *FLAGS* and *FLAGS2*)
-

Flagging Bad Data in Imaging

- To use the flags in the sweeps files, the following sequence of commands will be common:
 - $flag = 2^{**}9$
 - $w = np.where((objs[“OBJC_FLAGS”] \& flag) == 0)$
 - $objs = objs[w]$
 - Where, here, I’ve used *OBJC_FLAGS* but for the second set of flags I’d use *OBJC_FLAGS2*
 - Here, I’ve assumed that we’ve read the sweeps file into a structure called *objs* and that we want to recover the objects that do *not* have the flag set ($== 0$)
 - The value of “*flag*” would be looked up for the flag of interest (e.g. “*CR*” for a cosmic ray would be $flag = 2^{**}12$ for the column *OBJC_FLAGS*)
-

Other useful bitmasks

- Beyond flags, the SDSS (and other current surveys) use bitmasks to capture a range of information
 - Because the SDSS scans the sky multiple times, objects can be detected multiple times (or can be flagged as spurious due to only appearing in one scan)
 - The *best* observation of each *real* object is stored as *SURVEY_PRIMARY* ($2^{**}8$) in the *RESOLVE_STATUS* column of the sweeps. To recover *PRIMARY* objects, e.g.
 - $flag = 2^{**}8$
 - $w = np.where((objs[“RESOLVE_STATUS”] \& flag) \neq 0)$
 - $obj = objs[w]$
 - Often, it takes a great deal of trial-and-error to determine which flags should be applied, but *you will almost always want to restrict to SURVEY_PRIMARY*
-

Python tasks

1. Find the closest *galaxy* in the SDSS sweep files to the point $(\alpha, \delta) = (336.4388^\circ, -0.8343^\circ)$
 - When looking up only a few objects (and not matching to *WISE* forced photometry), it will be quicker to use the *sdss_sweep_circle.py* code in my week 10 directory rather than using *sdss_sweep_data_index.py*
 - Galaxy-like images can be retrieved by passing (*objtype*='gal' to *sdss_sweep_circle.py*)
 2. Is this galaxy a set of *blended* images? Does it contain any pixels that are *saturated*? Is it *blended* or *saturated* in every band or just in the overall combined image?
 - Find the image of this galaxy in the *SDSS Navigator Tool*. Does it look saturated? Is it even a galaxy?
-

Python tasks

3. Last lecture, you wrote code to separate spectroscopically confirmed quasars and stars using *ugriz* color cuts. Let's see how well that would work in a real imaging survey
- retrieve every point source (“*objtype=star*”) in the SDSS sweeps files imaging that lie within a 3° radius of the coordinate $(\alpha, \delta) = (180^\circ, 30^\circ)$. We'll call these *objs*
 - For a circular area, you can use *sdss_sweep_circle* to retrieve the *objs*...but in this case, also send *all=True* so we can illustrate the use of *SURVEY_PRIMARY*
 - restrict the *objs* in magnitude to $i < 20$
 - coordinate-match the *objs* to the *qsos-ra180-dec30-rad3.fits* file in my week 10 git directory, to find which $i < 20$ objects in SDSS imaging are quasars
-

Python tasks

4. Apply the color-cut code you wrote last lecture to the *objs* to determine which of them are likely to be quasars
- What is the *area* of the circle of radius 3° within which we are considering the *objs*?
 - Given that spectroscopy is the only way to know for sure if an object is a quasar or a star, how many spectra would we have to obtain per sq. deg. to determine the number of quasars per sq. deg. that your code recovers?
 - Find *flag* cuts on the *objs* that retain $> 90\%$ of known quasars (the quasars from my *qsos-ra180-dec30-rad3.fits* file) but that reduce the number of spectra per sq. deg. we'd have to obtain to confirm new quasars
 - (*SURVEY_PRIMARY* would be a good place to start...)
-