Description of fiber assignment code for Mocks

Robert Cahn and Louis Garrigue

March 9, 2015

1 Introduction

Martin White developed C++ code for fiber assignment over the full 14k sq. deg. footprint. We first modified Martin's code to incorporate features from Robert's python code, which ran on a restricted 480 sq. deg.. We included the improvement and redistribution steps, which switches fiber assignments to increase the number of galaxies observed. We then adapted the code to compute assignments not globally anymore "knowing all information on galaxies" but plate after plate, as in real experiment. The samples are taken from Martin's mocks in stored on NERSC at */project/projectdirs/desi/mocks/preliminary/*. From these files we create a single file containing the appropriate mix of ELG, LRG, QSO, SS (Standard Stars) and SF (Sky Fibers) using the python script in svn */code/survey/fiberassignment/FA_Bob/trunk/bin/make_catalog_rnc.py*. In the same place there is python code to produce a mixture of galaxies without any correlations, but with the correct dn/dz.

The code needs to know the locations of the positioners in the focal plane. They are given in :

$*DESIMODEL/data/focalplane/fiberpos.txt* It also needs to know the locations of the centers of the fields in the sky, i.e. the plates, and their order. The original code written by Martin White provided the option of having the plate centers given in a binary file or an ASCII file. We are now using the ASCII option by defining ASCIICENTERS at the outset. If one wants to use the previous binary format, he has to ask it to Robert and adapt it to the new structure. The format is that of $*DESIMODEL/data/footprint/desi − tiles.par*, but an alternative ASCII file can be provided. If the executable is `assign` then the calling sequence will look like :

```
./assign objects0.rdzipn desi-tiles.par fiberpos.txt assignment
```

Here the catalog of targets is in the NERSC directory : */projects/projectdirs/desi/mocks/preliminary/objects.rdzipn* the binary file created by `make_catalog`. The plate centers are provided here by $*DESIMODEL/data/footprint/desi− tiles.par* The locations of the positioners are given in $*DESIMODEL/data/footprint/desi − tiles.par* An option not now used is to write the actual assignments to a file here called `assignment`.

README.rst gives instructions for running on NERSC. The problem of colliding fiber positioners is addressed in a simplified way. Two galaxies observed on the same plate (tile) cannot be within "Collide" (2.1) mm of each other. This is enforced at the outset. A fiber assignment is not allowed if it uses a galaxy too close to one already assigned for this plate. Of course the assignments then depend on the order in which the plates are considered.

We summarize here the various components of the code to facilitate their modification later. It has been written so as it is as flexible as possible to modify.

Table 1: Characteristics of galaxy samples as set in make_catalog_rnc.py

type	id	priority	nobs	density/sq.deg.
Ly-A QSO	0	1	5	50
tracer QSO	1	1	1	120
LRG	2	3	2	300
ELG	3	5	1	2400
fake QSO	4	1	1	90
fake LRG	5	3	1	50
SS	6	2	1	?
SF	7	4	1	?

2 Source files

They all consist in a .h and a .cpp file and are in this inscreasing dependency order :

- macros : set as global some parameters of the program
- misc : a home-made library of structures (and functions on them) needed to manipulate concerning datas, but independent of them. There are pair (of int), List (of int), Table, Cube, and timing, printing, string conversion, error report items.
- structs : structures of the manipulated datas and their members
- global : main high-level functions and algorithms used in the program to collect information, assign fibers and print statistics. Important ones are described further
- main : neat and quickly understandable code that sum up all steps

3 Classes and structures

They are build in order to be independent from one to another, flexible, quickly understandable, in a logical way, and with no redundant information.

3.1 Features of galaxies (Feat)

Initialized in the main function, memories priorities, number of observations wished, and types of different galaxies.

3.2 Plate Parameters (PP)

Carries positions of fiber positions on the plate, spectrometer correspondence, and neighboring fibers information.

3.3 Plate

3.3.1 Plate coordinates (onplate)

Is used for coordinates in the focal plane in mm. The member `id` is used to give the identity of a galaxy. Onplates is vector of onplate.

3.3.2 A plate (plate)

Peculiarly describes both the location in the sky of the tile in terms of a unit vector derived from RA and DEC. Carries also the Id of the tile, its pass, and the available galaxies it is able to reach. Then Plates is vector of plate, and has all information on tiles.

3.4 Galaxy (galaxy)

Information on a galaxy : an `id` that corresponds to table 1, a position in the sky (in 2 differents ways), and the available tile-fibers that can watch at it. Gals is vector of galaxy and carries all information on galaxies.

3.5 Assignments (Assignment)

Carries maping of tile-fibers to galaxies, its inverse, galaxies to tile-fibers, and the cube (3d-matrix) of assigned fibers for a kind, a petal and a plate (useful to have fast computations).

4 Functions of global

`plate_dist(const double theta)` turns radians into mm on the focal plane, i.e. it is the plate scale as a function of angle. `change_coords` is a critical function that combines a galaxy and a particular plate to give the coordinates the galaxy will have in the focal plane when observed as with this tile. It's a rotation in angular coordinates. This ought to be rigorously checked. `int find_collision` returns the fiber number of a fiber that conflicts with tiblefiber (j,k). Conflict is defined by two observed galaxies being separated by less than `Collide`, current set to 2.1 mm. `bool ok_assign_g_to_jk_nobs` checks to see if we can assign g to the tile-fiber jk, according to assigning rules described further

4.1 Collecting

`collect_galaxies_for_all` is written for multithreaded, and for each fiber of each tile, collects reachable galaxies. It uses kdTree and htmTree libraries written by Martin White, that as absolutelly necessary to do computations in a reasonnable time with supercomputers.

`collect_available_tilefibers` computes, using the previous work, available tile-fibers for each galaxy (inverse map)

4.2 Assigning with all information

Here are algorithms that know all information and are able to compute the assignment crossing between plates (so unrealistic, but interesting to compare its "ideal" results to real assignment ones)

`assign_fibers` makes a first assignment of all fibers of all plates, in a shuffled order

`improve` improves the previously made first assignment. In a shuffled order, try to use unused fibers by reassigning some used one. Before : (jp,kp) - g ; (j,k) & gp free. After : (j,k) - g & (jp,kp) - gp.

`redistribute` redistrubte assignments trying to get at least 500 free fibers on each plate. We look first at plates with too few free fibers. Before : (j,k) - g, with Sp(k) too much used. After : (jreassign,kreassign) - g & (j,k) free, such that (jreassign,kreassign) comes from most unused (ji,ki). Reassign if this is improvement.

4.3 Assigning tile after tile

`assign_fibers_for_one` makes a first assignment of all fibers on one plate, with only information on previous watched galaxies

4.4 Displaying results

`void results_on_inputs` displays some statistics on treated input files (recall input features, print statistics on the number of fibers with 0 galaxies within reach, 1 galaxy within reach etc. out to 19 galaxies within reach ; number of available tile-fibers for a galaxy, ...)

`void display_results` writes some statistics and provides the tex-formatted results to make Table 2

`print_free_fibers` print histograms on free fibers towards/regarding petals, for everything, for on ly SS and for only SF

`conflicts` returns a complete list of all conflicts display them, a list of triplets (j,k,kp). There are no conflicts engendered by our algorithms.

5 Rules of assignment

- of course, a tile-fiber is only assigned once
- a galaxy can't be assigned more than once in a pass
- two observed galaxies can't be separated by less than `Collide` (set by find_collision)
- in last pass, only ELG, SS and SF are scrutinised
- there can't be more than 10 fibers assigned to SS on a petal
- there can't be more than 40 fibers assigned to SF on a petal

6 Main

The calling sequence is `assign <galaxy> <plate_centers> <fiberpos> <assignment>`. The last of these is the file to which we would write all the final galaxy assignments, but this is generally suppressed since we aren't using it yet.

`main` proceeds by taking important parameters, features of galaxies, reading in the galaxies to make G, the positions of the fibers on the plate to make pp, the plate centers and positioner locations to make P.

We then collect available galaxies for each fiber for each plate, and compute the inverse map.

We next, tile after tile, do the assignment of galaxies, by calling `assign_fibers_for_one`. The results are then written using `display_results` and `print_free_fibers`.

Table 2: Remaining observations (id on lines, nobs left on rows) with total

Id	0	1	2	3	4	5	total
0	0	443,259	240,890	72,694	24,033	7,767	788,643
1	1,867,438	27,182	0	0	0	0	1,894,620
2	3,873,418	602,268	240,657	0	0	0	4,716,343
3	26,925,090	11,147,890	0	0	0	0	38,072,980
4	1,400,629	20,383	0	0	0	0	1,421,012
5	753,676	35,774	0	0	0	0	789,450
6	1,066,600	1,143,862	0	0	0	0	2,210,462
7	4,266,400	17,838,234	0	0	0	0	22,104,634

Table 3: Id on lines. (Per square degrees) Total, Fibers Used, Available, Percent of observations

Id	0	1	2	3
0	49.457	168.796	49.948	99.015
1	118.275	118.275	119.996	98.565
2	283.469	528.792	298.711	94.897
3	1,705.31	1,705.31	2,411.36	70.719
4	88.709	88.709	90.000	98.565
5	47.734	47.734	50,	95.468
6	67.553	67.553	14,0	48.252
7	270.213	270.213	14,00	19.300