MagE's Python-Based Pipeline Beta CookBook

Penélope Longa-Peña

Abstract

I've been trying to understand the Carnegy's Python based pipeline by Dan Kelson for several months now. This is a gathering of the things I've figure out about it so I don't forget them / somebody else can use it after I'm gone! (Without the painful process of trial and error that leads to work with this undocumented pipeline)

Chapter 1

Setup

To setup the carnegie pipeline from an astronomy machine run:

```
source /storage/astro2/phsgan/Software/CarPy/Setup.csh
‘carpy’
```

A list of the carnegie commands can be found in:
`/storage/astro2/phsgan/Software/CarPy/dist/bin_oldnumeric/`

The pipeline is not designed to make time resolved spectroscopy, so it average all the spectra with the same object name together. To prevent this, the easier way is to change the “OBJECT” parameter in the .fits files by adding a number. This numbers shoud be 2 figures numbers. For example: Having 3 observations for the star Vega, change the parameter “OBJECT” by: vega_00, vega_01, vega_02. Why using 2 figures? it doesn’t matter if there are less than 10 observations, but if there is more than 10, then the pipeline will add to the ones with one figure all the others that start with that number (It will add to vega_1 outcome vega_10, vega_11, vega_12, etc...). So if the first couple of spectra have funny observation times, this step went wrong.

Using the recently installed version of the pipeline will crop some of the order away. The older version is already cracked, but if it is reinstalled there are some changes to be done inside the code for it to extract all the orders. This is under my own risk, I’m sure that Dan Kelson should have good reasons for not to want all the orders, but I do want them all. The change to do in the code is:

Inside
`/storage/astro2/phsgan/Software/CarPy/dist/bin_oldnumeric/mageDistortion`
change:
`‘Nw = rdarg(argv,"-Nw",int,13)’`
by
`‘Nw = rdarg(argv,"-Nw",int,15)’`

That will consider all the 15 orders for extraction.
Now, when is making the object appertures also crops some orders it doesn’t

like. It follows a logic that I still can't crack, but it could be any order that gets cropped away. To play safe and get sure all the orders you extract are OK, then leave things like they are. If you still want to force the pipeline to give you all the orders, it can be worked out as follows:

Inside
`/storage/astro2/phsgan/Software/CarPy/dist/bin_oldnumeric/mkobjectaps`
Look for "What to keep...", then change the loghical arguments for:

```
good = logical_or(between(relpos-relpos0,-5*relpos1,
+5*relpos1),less(abs(relpos-relpos0),0.2))
good2d = [True,True,True,True,True,True,True,True,True,True,True,True,True,True,True]
print good2d
good1d = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 ]
print good1d
really = equal(good1d,len(inputs))
print really
```

With this the pipeline will get all 15 orders no mater what.

Chapter 2

Reduction

There is some documentation about this in `http://code.obs.carnegiescience.edu/mage-pipeline` but I'm trying to put all togheter here:

2.1 Setting up the MagE Database File

The first command will create a text file named "your_dirMAGE.db" inside the directory in which you call it. My personal advice is to make a new folder for each reduction since if there is more than one night, there are going to be different calibration files, and the pipeline will call them all the same and it can get messy.

This is known as the "database". It could be helpfull to let you know what is in your reduction directory, since it extracts the information from the .fits headers and put it into one place.

To generate it just type:

```
magedb -d ‘‘your path’’
```

The path can be absolute or a relative path to the directory in which the .fits files are.

Example: If I keep my observation .fits files in a file call "night1" then I can make a new directory outside night1 for the reductions:

```
mkdir redu_night1
cd redu_night1
magedb -d ../night1
```

will generate a file called night1MAGE.db inside the redu_night1 directory, which will be your database file, That is what the pipeline is going to use to reduce.

This file other than the .fits headers information, also has the absolute path to all of your .fits files. I found this usefull when changing the headers names, just copy the absolute paths of the ones I wanted to change into a list and run it inside a script to number their "OBJECT" header.

ALWAYS look inside the database file before reducing. All the information for the reduction is saved there, so if there are some flats with different observation times or some targets that are not wanted, these should be removed out of the database file before the next step.

2.2 Generate pipelines

Before generating the pipelines it is necessary to change a bit the headers of the calibration objects and call them something remarkable. In principle one could call the red flats Fred and the blue ones Barney, but it is better to call them something more straightforward. Why changing their names? because lets say the blue flats and the redflats are both called flats, then the pipeline don't know which is which. My default way of calling all the non science targets are:

red flat	DOMEFLAT
blue flat	Xe flat
slit	trace
lamp	thar

Table 2.1: My default nomenclature for calibration headers.

This are names that need to be rememberd since are be going to be used in the next step.

Also a warning: "blue flats" and "slits" are the same thing, so the person who observe will ussually call them the same, making this renaming step really important.

Next, the command:

```
magesetup -db (and your .db file here)
```

with nothing else, should print out a list of the objects that are going to be reduced. This is very important to check, if you see here any of the names of the calibration targets, then something is wrong and should re-name them again. Alternatively, just carry on to the next step and deal with it later (I'll tell how in a minute).

Now, for `magesetup` do actually generate the pipelines, it is necesary to tell it to. There are several ways of telling it to, but the fastest is just to tell it to generate all the scienca pipelines at once:

```
magesetup -db night1MAGE.db -blueflatkey Xe -redflatkey DOMEFLAT -slitfnkey
trace -lampkey thar -all -mk makefile
```

Now the `magesetup` is being told to do "all" the pipelines, meaning, all the pipelines for the targets that has listed previously (and that it will list again while it's making them). Notice that the words after thw minus (-) sign are the program setups and the ones after those are the ones that were defined manually in the headers. This means that you could call your makefile anything you wanted as well. To make an specific target instead of `-all` write `-OBJECTkey ''name of the target here''`. This should generate a pipeline just for the

target that ou want. There will be new directories in the reduction directory now, one for each science target and also blueflat, redflat, flat, slit and lamp. Also one called Final products, but that will be empty by now. There will also be some .fits files, but are just images by now so no point trying to open them yet.

In this point, if there is a directory called "DOMEFLAT" because you saw it in the setup list and didn't dealt with it the directory should be removed (and you were lucky). If you're onlucky, then after the `magesetup` command the list of targets just passed really fast or got a message saying something like "problem with your red flats?". This means that when calling the red flats by `-redflatkey DOMEFLAT` you call it something wrong or slightly different to the one on the header. It says in the web page that this is not case sensitive, but in fact it is, if you don't call the setup command with the exact same key word that appears in the headers the pipeline generation fails. So basically if in the header says Domeflat and in the setup DOMEFLAT it will not be the same, so beware.

2.3 Running the pipelines

In theory, if you called your makefile "makefile"just typing "make" inside the reduction directory should magically do everything. I said in theory (or maybe also in Dan Kelson's computer, but not here). In reality that almost never occur. First if in the setup step the makefile was called something else, then to actually run it it is necesary to type: `make ''somethingelse''`.

While the pipeline is running from the reduction directory it shows in the shell a line with the name of the directory that is making and when it is done, writes next to it the name of the outfile for that directory. The "directory.out1" (or .out2, .out3 depending on how many times the reduction fails), will have all the outcomes of the pipeline written inside, every step and every command and of course, the errors.

By experience, the pipeline always breaks while doing the lamp, so if you just type make and trust everything is been done, probably can spend hours waiting and staring at the shell while the list of directories and .out files grows. Hint, when it is broken from the begining, the science targets come out very quickly.

What I do is to go inside the "lamp" directory first of all (`cd lamp`) and then type `make`. Doing "make" inside the directory will show you the output of the .out file in the shell. Handycap: there will not be a .out file. Still is always better ro do it inside the "lamp" directory because it can be seen inmediatly when it breakes and type "make" again. The average "lamp" reduction breaking is 3 times, so don't panic if it keeps crushing.

Then you could potentially go out to the reduction directory and make from there and everything should be alright. To be extra sure, I do go inside every calibration director and make inside. There is a proper order to do so: lamps, red/blueflat, flat, slit. Now finally is very difficult that the science targets go wrong after all the calibrations are OK, so making inside the reduction directory should work.

2.4 Outcome and How to Deal with It

The outcome from the pipeline will be in a directory named "Final-Products". These will be multi.fits files, multidimensional .fits files. The 5 dimensions are:

```
BANDID1 = 'sky spectrum'
BANDID2 = 'object spectrum'
BANDID3 = 'noise spectrum'
BANDID4 = 'signal-to-noise spectrum'
BANDID5 = 'lamp spectrum'
```

So obviously, we can't just easily open these whith anything... but luckly Danny Steeghs has made a python script that extract the dimension wanted. It' called "extract_multispec.py" and by default extracts the object spectrum into an asci file plus a .head file (with the headers to read in "molly"). If you want another dimension, you need to change the number in square braquets in line 66 inside Danny's script, but be aware that is done with python so instead of 1-2-3-4-5 are 0-1-2-3-4 BANDIDs. To make my life easier, I just made another script that calls Danny's extract_multispec and give me inmediatly the molly files. It's named "Mage2molly.py" and it's found in
`/storage/astro2/phrkaq/scripts/python_scripts`.
It needs the name of the object without the multi or the number and then to tell it from wich number it starts and how many of the object spectra are. It doesn't breake if one gives it wrong numbers, but it slows it down. I basically did it that way so if in the first step one just copy the complete list of objects and add it numbers or numbered object by object works anyhow. (Sounds quite confusing, but just give it a go and it's easy to see how it works.)