Fall 2016 Astro 250: Stellar Populations

Instructor: Dan Weisz (dan.weisz@berkeley.edu)

http://dweisz.github.io/ay250_fall2016/

Final Project Description

Pull Requests should be resolved by 12/12/16

The final project for this class is to create a 'unit test' for `python-fsps`. `python-fsps` is undergoing constant development, and the creation of unit tests will enable code sanity checking. For example, for a fixed set of input parameters, one might expect the optical color evolution of an SSP to remain constant as the code evolves with development. Deviations in excess of numerical precision, etc. may indicate an underlying problem in a new version of the code. Unit tests will help to ensure proper development of the code and ease identification of potential code issues.

Through pair-coding, your job is to develop a unit test and use the `git` interface to merge your changes into the master branch of `python-fsps`.

In no particular order, some possible unit tests are:

- Generate Figures 3-7 from Conroy, Gunn, & White (2009)
- Generate Figure 9 from Conroy, Gunn, & White (2009)
- Generate Figure 2 from Conroy & Gunn (2010), excluding M05, BC03, but adding FSPS/MIST
- Generate Figure 3 from Conroy & Gunn (2010), for various TB-AGB prescriptions
- Generate Figures 6, 10, 11 from Conroy & Gunn (2010), excluding M05, BC03, but adding FSPS/MIST
- Few unit tests exist for nebular emission. Develop a unit test for nebular emission. One example would be to looks at difference in R-band magnitudes and U-B colors at select ages/metallicities assuming no nebular emission at all, only nebular lines, only nebular continuum, lines + continuum.
- Generate plots showing the evolution of stellar mass, GALEX FUV, SDSS r,and WISE 1 magnitudes for sfh=1,4 with sparse grids (e.g., 2 values for each) in sfstart, sftrunc, tage, tau, tburst, fburst, and constant. No need to include nebular emission.
- Propose a unit test.

Timeline: Each pair of coders should let me know what unit test they propose for their final project no later than 11/23/16. Keep in mind that your changes should be completely merged into `python-fsps` by 12/12/16. Recall that this

project depends on people maintaining the `python-fsps` repository, so please be considerate of their time and do not try to rush through this project at the last minute.

Using Git: You can start by forking the `python-fsps` repository, creating a "feature branch" (e.g., named 'unit-test-x'), and modifying your version of the repository on this branch. When you are ready to incorporate changes into the `python-fsps` main branch, you can issue a pull request. Those unfamiliar with pull requests should take time to do necessary background reading. You should not hesitate to use the git collaboration interface (e.g., raising issues) in the event that problems arise.

Important: do no touch anything outside of your unit test code without a very good reason. This may cause problems with `python-fsps` and could result in your pull request being denied.

Code requirements: Your code should:

- Produce clear plots (i.e., clearly labeled, reasonable color-schemes)
- Be well documented, this includes comments in the source code, an explanation of what the code is designed to do, and how to use it. This includes specifying inputs/outputs, etc, using the designated API format (a template should soon be available in the `python-fsps` repository). Recall that your code will be used by the broader astronomical community, so clarity is important.
- Should be able to write your data to optionally output your data in HDF5 format (http://www.h5py.org).
- Should be able to optionally read in an HDF5 file and compare it to new data generated by `python-fsps`. Your code should report the level of disagreement between input and generated data.

It will be important for all codes to have the same API. An example API for your code might look like:

```
'''
test_x(make_plots=True, h5_in=None, h5_out=None)

:returns figure:
A matplotlib.pyplot.Figure object with plots showing the
quantities being tested (e.g. color vs age).  If make_plots is False this is None.

:returns delta:
If h5_in is not None, this will return a blob with
1) the maximum (or typical) delta between the reference data (h5_in)
```

```
and the current calculated quantities, and
2) some indication of where this difference occurs
'''
```