apogee_tools Documentation

Jessica Birky

Feb 08, 2023

User Guide:

`apogee_tools` is a forward modeling framework for fitting atmospheric models to stellar spectra. Following from Blake et al. 2010, we synthesize the model:

$$M(\lambda) = \left(\left[L \left(\lambda \times \left(1 + \frac{v}{c} \right) \right) \star K \right] \times T(\lambda) \right) \star LSF$$

where L is the high resolution model template (parameterized by `Teff`, `logg`, and `[Fe/H]`), K is the rotational broadening kernel (parameterized by `vsini`), T is telluric spectrum (with variable strength `alpha`), and LSF is the line spread function of the instrument. Optimal fits and uncertainties are sampled using Markov Chain Monte Carlo, implemented via `emcee` (Foreman-Mackey et al. 2012).

CHAPTER 1

Contributors

- Christian Aganze (UCSD)
- Jessica Birky (UCSD)
- Adam Burgasser, PI (UCSD)
- Dino Chih-Chun Hsu (UCSD)
- Elizabeth Moreno (Guanajuato)
- Chris Theissen (UCSD)

Code and documentation is maintained by Jessica Birky here, and is currently under construction. Feel free to contact jbirky@ucsd.edu with suggestions.

This code also borrows from several other sources, see:

- Starfish - Ian Czekala
- apogee - Jo Bovy
- TheCannon - Anna Ho

1.1 Installation

1.1.1 Dependencies

- astropy
- astroquery
- emcee
- numpy
- matplotlib
- pandas

- PyAstronomy
- scipy

1.1.2 APOGEE Data Setup

Create a new directory to store your data files:

```
$ mkdir apogee_data/
```

Then download the APOGEE data info file for DR14:

```
$ cd apogee_data/

$ wget https://data.sdss.org/sas/dr14/apogee/spectro/redux/r8/allStar-l31c.2.fits --
↪no-check-certificate
$ wget https://data.sdss.org/sas/dr14/apogee/spectro/redux/r8/allVisit-l31c.2.fits --
↪no-check-certificate
```

Download the apogee_tools code and then set up the following environmental variables in your `.bash_profile` or `.bashrc`:

```
export PATH=$PATH:'/Users/path_to/apogee_tools'
export APOGEE_DATA=/Users/path_to/apogee_data
```

1.2 Instrument Tools

Todo: Add description of Spectrum object class.

1.2.1 APOGEE Data

Downloading and reading APOGEE data files

To download APOGEE spectrum by 2MASS name and data type `aspcap`, or `apstar`:

```
import apogee_tools as ap
ap.download('2M03425325+2326495', type='aspcap')
ap.download('2M03425325+2326495', type='apstar')
```

For data type `apvisit` or `ap1d`:

```
ap.download('2M03425325+2326495', type='apvisit')
ap.download('2M03425325+2326495', type='ap1d', visit=1, frame=1)
```

Note: `type='apvisit'` will download the spectra for all visits observed, while `type='ap1d'` will download only the visit specified (and if not specified, will default to `visit=1`, `frame=1`).

For information on APOGEE data files, see the following:

- aspcap - combined, continuum normalized spectra
- apStar - combined spectra

- apVisit - individual raw visit spectra with telluric correction
- ap1D - individual raw visit spectra with NO telluric correction

Also for info about the allStar file (such as aspcap pipeline parameters and photometry for all of the sources), see: allStar.

Once the data for a source has been downloaded, read aspcap or apStar files by specifying the 2MASS name and data type:

```
data = ap.Apogee(id='2M03425325+2326495', type='aspcap')
```

Or for single visit spectrum, indicate the index of the visit number at the end:

```
data = ap.Apogee(id='2M03425325+2326495', type='apvisit', visit=1)
```

Search the APOGEE catalog

Example search–will search the `allStar-l30e.2.fits` you downloaded:

```
params = ['TEFF', 'LOGG', 'M_H']
ranges = [[-10000,4000], [0,5], [-2,2]]
source_table = ap.multiParamSearch(par=params, select=ranges, dir='/path_to/')
```

Look up aspcap parameters in `allStar-l30e.2.fits` for specific list of 2MASS IDs:

```
tm_ids = ['2M01195227+8409327']
ap_dict = ap.returnAspcapTable(tm_ids, params=['TEFF', 'LOGG', 'M_H', 'SNR'],␣
↪save=False)
```

Plot data

Some plotting examples:

```
data = ap.Apogee(id='2M03290406+3117075', type='aspcap')

# plot spectrum
data.plot()

# plot aspcap model and noise:
data.plot(items=['spec', 'model', 'noise'], save=True)

# plot indentified lines (from Souto 2016):
data.plot(items=['spec', 'lines'], xrange=[15200,15500], yrange=[.6,1.2])
```

Mask outlying flux

Specify number of standard deviations above and below the mean of the flux to cut (`sigma = [lower cuttoff, upper cutoff]`), and the number pixels to buffer each side of the cut (`pixel_buffer = [lower mask pixel buffer, upper mask pixel buffer]`):

```
data.mask(sigma=[3,2], pixel_buffer=[0,3])
```

Chi-squared comparison

Compare two spectra; return `chi` (chi-squared value between data and mdl), `norm_data` (`data` spectrum normalized), and `scaled_mdl` (`mdl` which has been scaled to `data`):

```
chi, norm_data, scaled_mdl = ap.compareSpectra(data, mdl)
```

1.2.2 NIRSPEC Data

Todo: More info coming soon.

1.2.3 Adding New Instruments

Todo: More info coming soon.

1.3 Modelling Tools

1.3.1 Reading in Model Grids

Read in a model, specifying the parameters `[Teff, logg, [Fe/H]]`, grid type (listed below), and wavelength range `xrange`. Models sampled to APOGEE resolution are contained in the libraries folder of this package, and span the following parameter ranges: `PHOENIX: [[2500, 5500], [0.0, 5.5], [-1.0, 1.0]]`, `BTSETTL (CIFIST 2011b & 2015): [[2200, 3200], [2.5, 5.5], [-0.5, 0.0]]`. To use grids outside of these ranges, download the libraries from the links below, create an `.hdf5` file using Starfish, and add it to the `libraries` folder.

```
mdl = ap.getModel(params=[3200, 5.0, 0.0], grid='BTSETTL', xrange=[15200,16940])
```

Grid types:

- PHOENIX (Husser et. al. 2013)
- BTSETTL (Allard et. al. 2010) - CIFIST 2011b

1.3.2 Synthesize a model

First specify a dictionary of stellar parameters:

```
params = {'teff': 3051, 'logg': 5.2, 'z': -0.25, 'vsini': 10., 'rv': -12, 'alpha': 0.
↪2}
```

Read in some data you are creating a model for:

```
ap.download('2M01195227+8409327', type='ap1d', visit=1, frame=1)
data = ap.Apogee(id='2M01195227+8409327', type='ap1d', visit=1)
```

Look up the spectrum's fiber number:

```
ap_id, plates, mjds, fibers = ap.searchVisits(id_name='2M01195227+8409327')
```

Synthesize a model: (with resolution options: 23k, 50k, and 300k)

```
mdl = ap.makeModel(params=params, fiber=fibers[0], plot=True, xrange=[15678,15694],␣
↪res='300k')
```

1.4 Analysis Tools

1.4.1 Plotting Features

Todo: More info coming soon.

1.4.2 Atomic and Molecular Lines

Search atomic and molecular lines from the following databases:

NIST (download1): atomic lines

HITEMP (download2, paper2): molecular lines H2O, CO2, CO, NO, and OH

APOGEE (download3, paper3): atomic and molecular lines

Functions

Search what line libraries are available, and what lists of elements/molecules are stored in each:

```
>> import apogee_tools as ap

>> ap.listLibraries()
   ['APOGEE_MOLEC', 'SOUTO', 'APOGEE_ATOMS', 'HITEMP', 'NIST']

>> ap.listSpecies('APOGEE_ATOMS')
   ['CC', 'CN', 'CO', 'HH', 'OH', 'SIH']

>> ap.listSpecies('SOUTO'))
   ['Al I', 'Ca I', 'Cr I', 'Fe I', 'FeH', 'K I', 'Mg I', 'Mn I',
'Na I', 'OH', 'Si I', 'Ti I', 'TiO', 'V I'], dtype='<U4')

>> ap.listSpecies('APOGEE_ATOMS')
   ['AL I', 'AL II', 'AR I', 'AR II', 'AR III', 'AU I', 'B I', 'B II',
   'C I', 'C II', 'C III', 'CA I', 'CA II', 'CA III', 'CE III',
   'CL I', 'CL II', 'CL III', 'CO I', 'CO II', 'CR I', 'CR II',
   'CR III', 'CS I', 'CU I', 'CU II', 'F I', 'F II', 'F III', 'FE I',
   'FE II', 'FE III', 'GE I', 'HE I', 'K I', 'K III', 'LI I', 'MG I',
   'MG II', 'MN I', 'MN II', 'MN III', 'N I', 'N II', 'N III', 'NA I',
   'NE I', 'NE II', 'NI I', 'NI II', 'NI III', 'O I', 'O II', 'O III',
   'P I', 'P II', 'P III', 'RB I', 'S I', 'S II', 'SC I', 'SC II',
   'SC III', 'SI I', 'SI II', 'SI III', 'SR II', 'TI I', 'TI II',
   'TI III', 'V I', 'V II', 'V III', 'Y I', 'Y II', 'ZN III']
```

(continues on next page)

(continued from previous page)

```
>> ap.listSpecies('HITEMP')
   ['CO', 'CO2', 'H2O', 'NO', 'OH']
```

Search a wavelegth region for certain species of atoms/molecules (returns a dictionary):

```
>> lines = ap.searchLines(species=['OH', 'Fe I'], range=[15200,15210], \
           libraries=['NIST', 'APOGEE_ATOMS', 'APOGEE_MOLEC', 'HITEMP'])

   {'Fe I': array([15201.822, 15202.952,     0.   , 15207.106, 15208.251]),
    'OH': array([15200.214     , 15200.332     , 15201.556     , 15201.774     ,
    15202.037     , 15202.215     , 15202.296     , 15202.366     ,
    15202.93      , 15203.768     , 15203.908     , 15203.98      ,
    15204.371     , 15204.548     , 15205.168     , 15206.303     ,
    15207.019     , 15207.416     , 15207.546     , 15207.659     ,
    15208.254     , 15208.613     , 15209.474     , 15200.68827257,
    15202.3097307 , 15202.31102492, 15202.53883371, 15204.0112159 ,
    15204.21089622, 15204.31956908, 15205.52045337, 15205.71449241,
    15206.01852145, 15206.27113677, 15206.33572209, 15206.51148174,
    15207.09644387, 15207.93042108, 15208.05808934, 15208.06268267,
    15208.125648  , 15208.34296383, 15208.8060811 ])}
```

Example

How to identify lines in an *APOGEE* spectrum:

1. Read in a the spectrum of a source, and interpolate the spectrum using a spline function to determine where the max/min points are:

```
>> spec = ap.Apogee(id='2M19213157+4317347', type='aspcap')
>> spec = ap.rvShiftSpec(spec, rv=-80)
>> spec.name = '2M19213157+4317347'

>> interp, local_min, local_max = ap.splineInterpolate(spec)
>> min_lines = {'min':np.array(local_min)}
```

2. Create a list of species that you want to search for.

For example to search for `Fe I`, `Ca I`, `Mg I` and `K I`:

```
>> fe = ['Fe I', 'FE I']
>> mg = ['Mg I', 'MG I']
>> ca = ['Ca I', 'CA I']
>> k  = ['K I', 'K I']

>> species = fe + mg + ca + k
```

or to search for all of the species in all of the libraries:

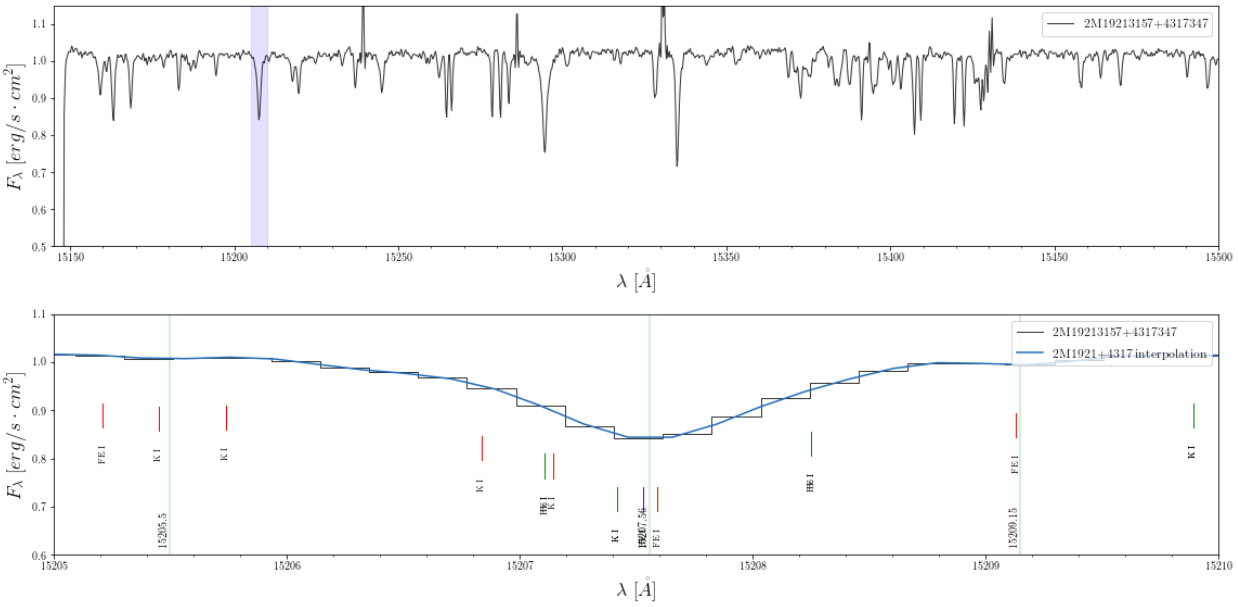
```
>> species = sum([ap.listSpecies(lib) for lib in ap.listLibraries()], [])
```

3. Now choose the line lists you want to search and plot the spectrum. Pick a wavelength region with a feature to `zoom` onto. Here the green lines mark the minimum points from the interpolated spectrum. For example searching for species `Fe I`, `Ca I`, `Mg I` and `K I` returns:

```
>> broad = [15145,15500]
>> zoom  = [15205,15210]

>> lines1 = ap.searchLines(species=species, libraries=['APOGEE_ATOMS', 'APOGEE_MOLEC
↪'], range=zoom)
>> lines2 = ap.searchLines(species=species, libraries=['NIST'], range=zoom)
>> lines3 = ap.searchLines(species=species, libraries=['SOUTO'], range=zoom)

>> spec.plot(xrange=broad, yrange=[.5,1.15], highlight=[zoom])
>> spec.plot(items=['spec'], xrange=zoom, yrange=[.6,1.1], line_lists=[lines1, lines2,
↪ lines3], \
          line_style='short', style='step', objects=[interp], vert_lines=[min_lines])
```



4. Now check what lines were found in the `zoom` range for each list:

```
>> lines1
>> lines2
>> lines3
```

Then play around with the `zoom` range to get a better view of the feature.

1.5 MCMC Fitting

Warning: These functions are under construction.

1.5.1 Setup

1. Copy the `config.yaml` and `run.py` from the main directory to an external folder.
2. Edit your configuration script `config.yaml`, which should look something like below.
3. In your new directory run `python run.py` in terminal.

1.5.2 Configuration

```
# Instrument specifications
data:
  instrument: "APOGEE"
  data_path: "default" # defaults to $APOGEE_DATA path (see setup documentation),␣
↪unless otherwise specified
  ID: "2M01195227+8409327"
  orders: [[15200,15800],[15860,16425],[16475,16935]] # wave ranges, and orders
  dtype: "ap1d"
  visit: 1
  sigma_clip: [.3,.05]
  pixel_buffer: [0,2]

# Make sure this config.yaml and run.py files are placed in your input directory
# I recommend copying config.yaml and run.py to a path external to apogee_tools
workdir:
  input: "/home/jess/Desktop/Research/FAST/fit_models"
  output: "/home/jess/Desktop/Research/FAST/fit_models/output"

out:
  mcmc_sampler: False
  corner: False
  walkers: False
  print_report: True

# Specify which parameters will be sampled by MCMC
# otherwise parameters will be fixed at 'init' values
model:
  grid_name: "PHOENIX" #directory: phoenix/apogee/order
  theta: ['teff', 'logg', 'fe_h', 'rv', 'vsini', 'alpha']

fix_param: # specify fixed parameters (not sampled by MCMC)
  airmass: 1.0  # airmass of telluric model, either 1.0 or 1.5
  cont_deg: 5   # continuum polynomial degree
  interp_method: "splat" # or "cannon"
  resample_method: "fast" # or "splat"

# MCMC tuning
mcmc:
  nwalkers: 12
  nsteps: 3

# Initial parameters for MCMC
init:
  teff: 3500
  logg: 4.50
  fe_h: 0.0
  rv: -4.77
  vsini: 5.79
  alpha: 1.0

# Step parameters for MCMC
step:
  teff: 1
  logg: .01
  fe_h: .01
```

(continues on next page)

(continued from previous page)

```
  rv: .1
  vsini: .1
  alpha: .01

# Prior ranges for MCMC (for flat prior)
prior:
  teff: [2500, 5500]
  logg: [0.0, 5.5]
  fe_h: [-1.0, 1.0]
  rv: [-200, 200]
  vsini: [0, 200]
  alpha: [0, 5]
```

1.5.3 Pre-MCMC Testing

To test to make sure all of the modeling modules are working, run the following command in terminal:

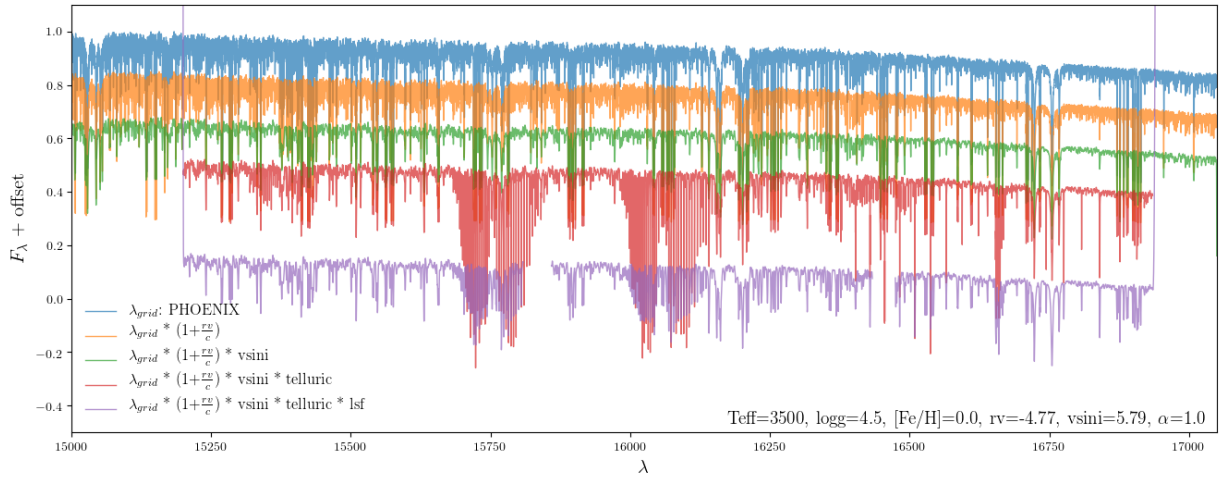
```
python run.py make_model
```

which should return something like:

```
[25.732014894485474s] MCMC initialization step complete.

##################################################
Making model: teff=3500 logg=4.5 fe_h=0.0 rv=-4.77 vsini=5.79 alpha=1.0

[0.07615256309509277s] Interpolated model
[0.0025053024291992188s] Shifted radial velocity
[0.0032796859741210938s] Applied vsini broadening
[0.05470013618469238s] Convolved telluric model
[0.08379793167114258s] Applied LSF broadening
```



To test by eye, that your initial MCMC parameters are some close to the data:

```
python run.py test_fit
```

1.5.4 Running the MCMC

Run the MCMC:

```
python run.py mcmc
```

Plot the outputs:

```
python run.py walkers
python run.py corner
```

CHAPTER 2

Search

- genindex
- modindex
- search