Galactic Astronomy: Problem Set 2

Jonas Powell, *Wesleyan University*

February 20, 2019

Due: Thursday, Feb. 21 by midnight. Late papers are not accepted. If you cannot complete the assignment, hand in what you have completed before the deadline. Consider the deadline to be like the boarding time for an airplane, or the deadline for a grant submission to NASA or NSF. If you miss the deadline, you do not get on the airplane, no matter how good your excuse is. If you miss an NSF or NASA deadline, you do not get the grant, no matter how good your project is. The best advice is ... finish early. You can submit multiple times, right up to the deadline. Whatever your latest submission is, when the deadline occurs, is what will be graded.

Problem 1. There is a small error in the book in the first paragraph of Section 2.3. What is the correction required?

Answer 1. In that first paragraph, the author gives the wrong units for f_ν, saying that it comes in units of W m^{-2} s^{-1} Hz^{-1}. This is wrong; the units should be power per area per frequency, meaning that the inverse time term that they have is unnecessary. The correct units are W m^{-2} Hz^{-1}.

Problem 2. Estimate the effective temperature of a star with the following properties, by fitting a black body curve to its flux density distribution:

$$B = 9.31,\ V = 8.94,\ J = 8.11,\ H = 7.93,\ K = 7.84$$

Plot the data and the black body that you chose as your best fit to the data and attach the plot to your answer. You can make this fit simply by eye, or you can use a more sophisticated fitting process – it is up to you. Based on the color of the star, estimate its spectral type, neglecting interstellar reddening. Compare the effective temperature based on the black body fit to the one based on the spectral type of the star and comment on the difference. Assuming that the star has luminosity class V, what is its distance, again neglecting interstellar reddening?

Answer 2. To fit the data, we first convert them from magnitudes to flux densities by recalling:

$$f_\nu = \text{zpf } 10^{-0.4\ M},$$

where ZPF is the spectral band's zero-point flux (given on the course Moodle page), and M is the star's magnitude in the spectral band. As a model, we will use a blackbody curve, given by Planck's Law:

$$B(\nu, T) = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{k_B T}} - 1}$$

However, Planck's law returns brightness (or specific intensity), but we would like to have it in units of flux density, to match our data. This is something of a problem, since the relationship between the two is given by

$$\begin{aligned} f_\nu &\approx \int_{\text{source}} B(\nu, T) d\Omega \\ &\approx \Omega_0 B(\nu, T), \end{aligned}$$

where Ω is the solid angle subtended by the source. Since astronomical sources are very far away, this solid angle is bound to be extremely small, meaning that the resulting scaling factor, Ω_0, between $B(\nu)$ and f_ν will be an extremely small number. Further complicating this is the fact that, since we don't know the star's distance, we don't know the solid angle it subtends anyways. Therefore, we must now fit both for temperature, T, as well as for Ω_0.

Since I find manually fitting two dimensions by hand to be a bit tedious, I decided to fit the two dimensions using a Markov Chain Monte Carlo (MCMC) routine. MCMC is a valuable tool for modeling in astronomy, and the field has benefited greatly from the Python package `emcee` (Foreman-Mackey et al. 2012), which makes this algorithm accessible fairly easily. MCMC works by sampling, in a pseudo-random walk, the probability distribution for a given parameter space, which means that not only is a best-fit value accessible (at the high-point of that distribution), but so too is an approximation of the distribution itself, which is an extremely informative feature to have.

To run my MCMC, I developed a probability function that sampled a blackbody curve of a given temperature at the frequencies of the given data, then calculated a χ^2 value between the data and model ($\chi^2 = \sum_i (\text{data} - \text{model})^2$), which was then turned into a log-probability ($\ln \text{prob} = -0.5\chi^2$) value and fed back into the MCMC run, informing the run's next step. The log-probabilities are shown in Fig in $\Omega_0 - T$ parameter space.

As we see, there is some degree of degeneracy between the two parameters. This is somewhat reasonable, since increases in temperature and solid angle-subtended will both result in increased flux density for a source. However, the resulting fit (Fig) looks pretty splendid, returning a best-fit temperature of 6745 K, which is characteristic of an F3 star. This is quite close to the F2 stellar type that is predicted by the star's 0.37 B-V color, indicating that the star is either truly an F3 that has been a bit reddened, or it is an F2 star that is slightly too cool.

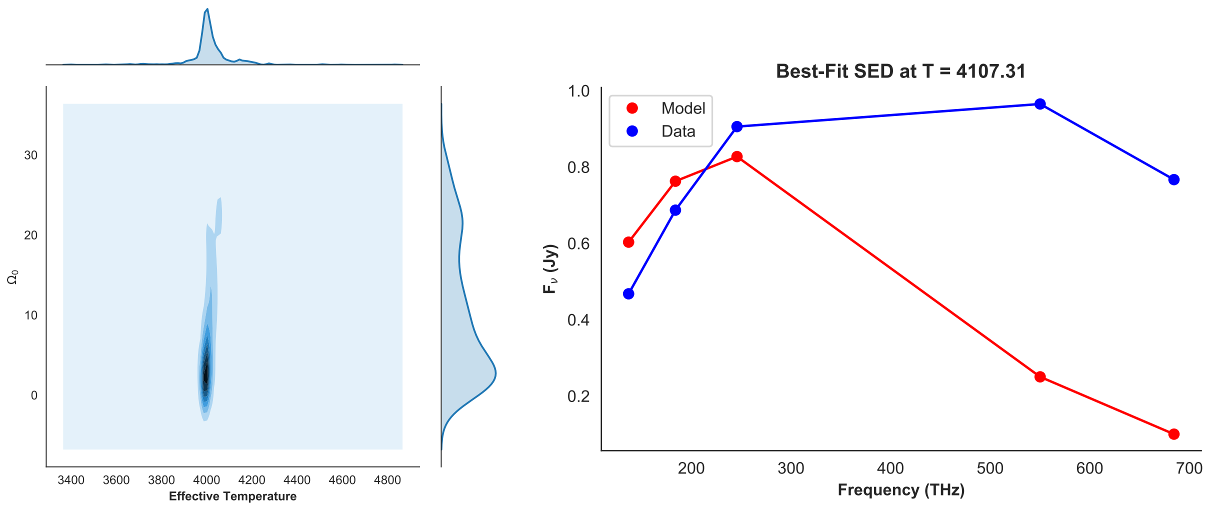

Figure 1: Results from an MCMC fitting run. The kernel-density estimate (KDE) on the left characterizes the probability distribution of points in $\Omega_0 - T$ parameter space, where darker regions correspond to better fits. On the right, data and best-fit model are overlaid, demonstrating that a good solution was found.

Problem 3. Given the information on a fictitious star, as listed in the table below, determine its other properties. Present your results as a table and explain each calculation in comments below the table.

Table of data on a fictious star

α, δ	13:42:25.6, -7:13:42.1 (J2000.00)
μ	13.7 mas y^{-1}
PA	122
v_r	-13 km s^{-1}
V	10.86
B-V	1.63
SpT	K2V

Properties for you to determine and tabulate: (l,b), M_V, E(B-V), A_V, d, the magnitude of heliocentric space velocity (v_{space}), M_{bol}, $L/L_\odot$, T_e, $R/R_\odot$, and $M/M_\odot$. Explain how you obtained each of the requested quantities in comments below the table. In addition, comment on the likely age and chemical composition of this star and whether it is likely to have a planetary system. As always, justify your answers.

Results

l, b	(324.52^o, 53.49^o)
M_V	6.19
E(B-V)	0.75
A_V	2.25
d	30.48 pc
v_{space}	2628 km s^{-1}
M_{bol}	5.9
T_e	5040
$L/L_\odot$	0.27
$M/M_\odot$	0.72
$R/R_\odot$	0.68

Answer 3. How I found these values: Many of these questions required looking up values in a table. For any question where this was the case (i.e. where I say "I looked up this value"), the table I am referring to can be found here: `http://www.pas.rochester.edu/~emamajek/EEM_dwarf_UBVIJHK_colors_Teff.txt`

l, b: Using the calculator I made for the last problem set, I found that the (l, b) coordinates of this source are (324.52^o, 53.49^o).

E(B-V): Recalling the definition E(B-V) = (B-V) - $(B\text{-}V)_0$, where the first term is the observed color and the second term is the color we would expect from a star of this spectral type, we just have to look up this expected color and subtract it from the given B-V color. I looked up a value of $(B\text{-}V)_0 = 0.884$ for a K2V star which yields a value of E(B-V) = 0.75.

A_V : Recalling the definition A_V = 3.1 E(B-V), we may just multiply our existing answer by 3.1. Doing so, we find A_V = 2.25.

M_V: Looking up in our table, we find that for a K2V star, M_V = 6.19.

M_{bol}: We recall that $M_{\rm bol} = M_V + BC_V$, where BC_V is the Bolometric Correction in the V-band, and can be looked up to be -0.29. This yields a result of M_{bol} = M_V + BC_V = 6.19 - 0.29 = 5.9.

d: We may rearrange the magnitude equation for distance:

$$
\begin{aligned}
V - M_V &= 5 \log d - 5 + A_V \\
\rightarrow \log d &= \frac{V - M_V - A_V + 5}{5} = 1.48 \\
\rightarrow d &= 10^{1.48} \\
&= 30.48 \text{ pc}
\end{aligned}
$$

v_{space}: The space velocity is the triangulation of the proper motion, μ, and the radial velocity, i.e. $v_{space} = \sqrt{v_{rad}^2 + v_{trans}^2}$. To solve this, we must turn proper motion into a velocity, using the familiar equation, $v_{trans} = 4.74\ \mu\ d = 4.74\ (13.7)(30.48) = 2628$ km s^{-1}. Therefore, $v_{space} = \sqrt{(13)^2 + (2628)^2} \approx 2628$ km s^{-1}. This feels way too high, but I don't really have any justification to back up that intuition and the math seems tight, so I guess that's what I'll submit.

T$_{eff}$: Looking it up, we find that for a K2V star, T$_{eff}$ = 5040.

L/L$_\odot$: Since luminosity and magnitudes are related as $M - M_\odot = -2.5 \log \frac{L}{L_\odot}$, then we may look up values for M and $M_\odot$ and find $\frac{L}{L_\odot} = 10^{-0.4\ (M-M_\odot)} = 10^{-0.4\ (6.19-4.79)} = 0.27$

M/M$_\odot$: From the Mass/Luminosity relation, we know that $\frac{M}{M_\odot} = \frac{L}{L_\odot}^{1/4} = 0.72$

R/R$_\odot$: We may find the radius by first recalling:

$$T_{eff}^4 = \frac{L}{4\pi\ R^2 \sigma}$$

Since we know T$_{eff}$ for our fictional star and the Sun (another lookup reveals that a G2V's effective temperature is 5770), and we know the two's luminosity ratio, we may construct the following ratio by rearranging the above equation and solving for the radius relationship:

$$\begin{aligned} \frac{L}{L_\odot} &= \frac{4\pi\ R^2 T_{eff}^4}{4\pi\ R_\odot^2 T_{eff,\odot}^4} \\ \rightarrow \frac{R}{R_\odot} &= \sqrt{\frac{L}{L_\odot}\left(\frac{T_{eff,\odot}}{T_{eff}}\right)^4} \\ &= \sqrt{0.27 \times \left(\frac{5770}{5040}\right)^4} \\ &= 0.68 \end{aligned}$$

Thus, $R/R_\odot \approx 0.55$.

Given that this is a K-star, we expect it to have a high metallicity. Lower mass stars tend to live much longer, so it is more likely that this star is an older star than it's higher-mass companions.

To determine the star's planet forming potential, I decided to see what the real data say about planets around this type of star. To do so, I used the Python package `kplr`, an API wrapper of the Kepler database's data portal, to download information about ~2300 planets. Each planet's information contained a field with information about it's host star. This allowed me to develop my own little database with features like metallicity, effective temperature, colors, and so on for each confirmed planet's host.

Unfortunately, one feature that my database did not yet contain was spectral type. This is obviously critical, since the whole point of this little project was to determine the relative frequency of planets around stars by spectral type. I decided to use effective temperature as a spectral type proxy, since it goes linearly with spectral type and I had access to good data on the spectral class/T$_{eff}$ relationship. To do this, I copied, processed, and tabulated the table that I used above for other reference values into a robust dataframe in Python. I was then able to iterate through each star in my database, cross-check its reported T$_{eff}$ against the list of effective temperatures in the linked table, and label the star with the spectral type attached to the nearest T$_{eff}$ value in the linked table. This is a somewhat sketchy way of spectral-typing stars and, were this a more serious project, is definitely something I would give more thought to, but for the time being, it gave me a good, quick way to get approximate spectral types.

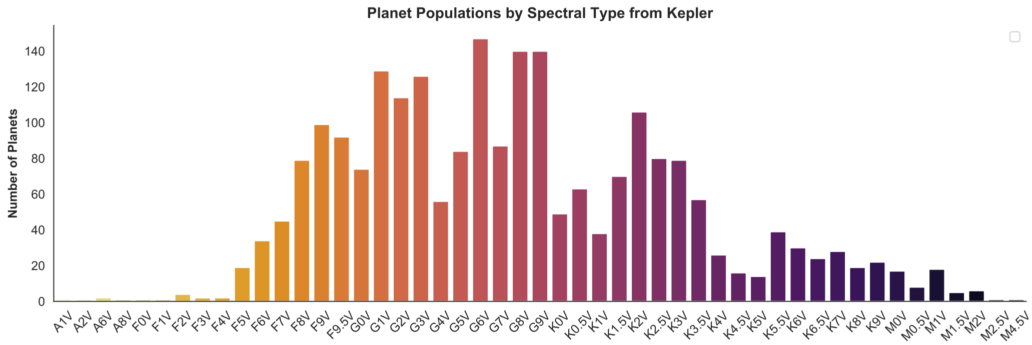

Figure 2: Planetary populations around stars of different spectral types, drawn from Kepler data. Unfortunately, it's hard to set photos in a two-column layout, so the inclusion of this plot forced me back into a one-column submission. Still, I'm really excited about how the plot turned out, so I think it's worth it.

Once I had these types, it was simply a matter of making an histogram of the results.

Planet Counts by Spectral Class

A	F	G	K	M
5	378	1097	760	56

We can see from Table that K stars have the second most detections of planets orbiting them, trailing only our beloved G stars. This indicates that planet-forming potential is likely high. Zooming in on this particular star's subtype, a K2V star, we see from Fig 2 that this specific class actually has the highest planetary detection rate of all K-stars, and the seventh highest rate overall. Therefore, from this data, I would say its planet-forming potential is high.

Note: I have entirely ignored, of course, the very significant issue of detection bias in this dataset. For example, we know that planets are extremely common around M-dwarfs, but since they are faint and hard to detect, they are underrepresented in this distribution. Still, since this is nothing more than a first-order diagnostic and the fictious star in question happens to be of a spectral type that has lots of observed planets, I think it's a reasonable sacrifice to make. Also, while it might not be enormously informative of the "true" planet-forming potential, it certainly is a useful way to predict how likely it is that we'd actually be able to find a planet around it. Finally, my data is all from the Kepler mission; a more robust implementation would, at the very least, also draw on K2 data and hopefully would reach out to other telescopes as well.

Problem 4. Fun with magnitudes and colors! If a star with a parallax of 25 mas that initially appears to be single and is measured to have V=6.50, later turns out to be a spectroscopic binary with equal mass components, what is the apparent magnitude of each component? Neglecting reddening, what is the absolute magnitude of each component? If that same star has a measured B-V = 1.25, what is the B-V of each component?

Answer 4. Since magnitude is given by

$$V = -2.5 \log \frac{f}{f_0},$$

then f, the flux of a star, is

$$f/f_0 = 10^{-0.4\ V}$$

Therefore, given that the V-magnitude of a binary is V=6.5, then we know that

$$\begin{aligned}(f_1 + f_2)/f_0 &= 10^{-0.4\ V}\\ &= 0.0025\end{aligned}$$

Given that the two sources have the same mass, we may draw on the Mass-Luminosity relationship (and recall that flux is just distance-scaled luminosity) to infer that they have the same fluxes. Therefore,the flux of one of the sources is

$$f_1/f_0 = \frac{0.0025}{2}$$

We can now plug this value back into our original equation for magnitude and find a single source's contribution.

$$\begin{aligned}V &= -2.5 \log \frac{f_1}{f_0}\\ &= -2.5 \log \left(1.25 \times 10^{-3}\right)\\ &= 7.26\end{aligned}$$

We find a final apparent V magnitude of 7.26 for each star. To find the B-V color of each component, we can play the same game as above, just in the B-band. Again,

$$\begin{aligned}B &= -2.5 \log \frac{f}{f_0}\\ \rightarrow f/f_0 &= 10^{-0.4\ B}.\end{aligned}$$

Since our apparent B-magnitude is B = 1.25 + (V = 6.5) = 7.75, then:

$$\begin{aligned}(f_1 + f_2)/f_0 &= 10^{-0.4\ B}\\ &= 7.94 \times 10^{-4}\end{aligned}$$

Again, we may recall that the two sources have the same mass and thus the same flux contributions, so

$$f_1/f_0 = \frac{0.000794}{2}$$

We can now plug this value back into our original equation for magnitude and find a single source's contribution.

$$\begin{aligned} B &= -2.5 \log \frac{f_1}{f_0} \\ &= -2.5 \log \left(3.97 \times 10^{-4}\right) \\ &= 8.51 \end{aligned}$$

We see that each source has an apparent B-magnitude of 8.51, resulting in B-V color of 8.51-7.26 = 1.25, the same as the binary's combined color. I guess this is fairly obvious; the combined color of two identical sources should be the same as the color of each source. Still, it's pretty neat to get to walk through the actual equations themselves and see that that relationship does, in fact, hold.

To find the star's absolute magnitude, we recall the relationship between apparent magnitude, absolute magnitude, and distance.

$$V - M = 5 \log d - 5 + A_V$$

Here $d = \frac{1}{25 \text{ mas}} = 400$ pc and A_V, the reddening, is taken to be zero. Therefore,

$$\begin{aligned} M &= V - 5 \log 400 + 5 + 0 \\ &= -0.75, \end{aligned}$$

so each star has an absolute magnitude in the V-band of -0.75. This is quite bright, perhaps a bit brighter than what I'd expect, but I guess it is quite far away, so maybe that does make sense.

Problem 5. More fun with magnitudes and colors! Suppose a binary system is composed of an A star, with V = 7.80 and B-V = 0.00, and a K star, with V = 8.20 and B-V = +1.50. If the stars are so close together on the sky that that they cannot be resolved as individual objects (i.e. an unresolved binary), what will be the measured V magnitude and B-V color of the "star" (that is actually the combined light of both components)?

Answer 5. We may use the same logic as before again to find the star's fluxes in each band and then work backward to single-source magnitudes.

$$
\begin{aligned}
f_{A,V}/f_0 &= 10^{-0.4\ V_{A,V}} \\
&= 7.59 \times 10^{-4} \\
f_{A,B}/f_0 &= 10^{-0.4\ V_{A,B}} \\
&= 7.59 \times 10^{-4} \\
f_{B,V}/f_0 &= 10^{-0.4\ V_{B,V}} \\
&= 5.25 \times 10^{-4} \\
f_{B,B}/f_0 &= 10^{-0.4\ V_{B,B}} \\
&= 1.32 \times 10^{-4}
\end{aligned}
$$

Combining sources A and B's V-band and B-band fluxes, we find:

$$
\begin{aligned}
V_{both} &= -2.5 \log \left[(7.59 + 5.25) \times 10^{-4}\right] \\
&= 7.23 \\
B_{both} &= -2.5 \log \left[(7.59 + 1.32) \times 10^{-4}\right] \\
&= 7.96
\end{aligned}
$$

Solving, we find that the binary would present as a star with V=7.23, B-V=0.73.

```python
"""
Calculations for Problem Set 2
Galactic Astronomy
Due Feb. 14
"""

import sys
import emcee
import cPickle
import numpy as np
import pandas as pd
import seaborn as sns
import astropy.units as u
import matplotlib.pyplot as plt
from astropy.coordinates import SkyCoord, Angle, get_constellation
from astropy.constants import c, h, k_B
from collections import Counter
from inspect import *
sns.set_style('white')

# Load the data for problem 2
mags = [9.31, 8.94, 8.11, 7.93, 7.84]




# Problem 2

def spectrum(t, coeff, show=False):

    temp = t * u.K
    mags = [9.31, 8.94, 8.11, 7.93, 7.84]
    wavelengths = [0.438 * u.um, 0.545 * u.um, 1.22 * u.um, 1.63 * u.um, 2.19 * u.um]
    freqs = [(c/(lam).decompose()).to('Hz') for lam in wavelengths] # Hz

    unit_nu = (1e-20 * u.erg / (u.cm**2 * u.s * u.Hz)).to('Jy')
    zpfs_nu = [4.063 * unit_nu, 3.636 * unit_nu, 1.589 * unit_nu, 1.021 * unit_nu, 0.64 * unit_nu]
    data_f_nu = [zpf * 10**(-0.4*mag) for mag, zpf in zip(mags, zpfs_nu)]

    fudge = (coeff * 1e18*u.Jy)**(-1)
    model_f_nu = [((2 * h * nu**3 * c**(-2)) / (-1 + np.exp(h * nu/(k_B * temp)))).to('Jy') * fudge
                  for nu in freqs]

    freqs_thz = [nu.to('THz').value for nu in freqs]
    model = [f.value for f in model_f_nu]
    data = [f.value for f in data_f_nu]


    if show:
        plt.show()
```

```python
    else:
        pass

    return (freqs_thz, model, data)



def lnprob(p, mags, priors, save=False):
    t, coeff = p

    # Check on priors:
    for param, prior in zip(p, priors):
        if not prior['min'] < param < prior['max']:
            return -np.inf

    temp = t * u.K
    wavelengths = [0.438 * u.um, 0.545 * u.um, 1.22 * u.um, 1.63 * u.um, 2.19 * u.um]
    freqs = [(c/(lam).decompose()).to('Hz') for lam in wavelengths] # Hz


    unit_nu = (1e-20 * u.erg / (u.cm**2 * u.s * u.Hz)).to('Jy')
    zpfs_nu = [4.063 * unit_nu, 3.636 * unit_nu, 1.589 * unit_nu, 1.021 * unit_nu, 0.64 * unit_nu]
    data_f_nu = [zpf * 10**(-0.4*mag) for mag, zpf in zip(mags, zpfs_nu)]

    fudge = (coeff * 1e18*u.Jy)**(-1)
    model_f_nu = [((2 * h * nu**3 * c**(-2)) / (-1 + np.exp(h * nu/(k_B * temp)))).to('Jy') * fudge
                  for nu in freqs]

    freqs4plotting = [nu.to('THz').value for nu in freqs]
    model = np.array([f.value for f in model_f_nu])
    data = np.array([f.value for f in data_f_nu])

    chisq = np.sum((model - data)**2)
    lnp = -0.5 * chisq
    return lnp



def run_mcmc(mags=mags, nsteps=1000):

    ndim, nwalkers = 2, 30
    priors_t = {'min': 0, 'max': 20000}
    priors_coeff = {'min': 0, 'max': 30}
    priors = [priors_t, priors_coeff]

    p0 = np.random.normal(loc=(4000, 4), size=(nwalkers, ndim))
    sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[mags, priors])
    pos, prob, state = sampler.run_mcmc(p0, 10)
    prob
```

```
    sampler.reset()
    print "Finished burn-in; starting full run now."
    # sampler.run_mcmc(pos, nsteps)

    run = sampler.sample(pos, iterations=nsteps, storechain=True)
    steps = []
    for i, result in enumerate(run):
        # Maybe do this logging out in the lnprob function itself?
        pos, lnprobs, blob = result

        new_step = [np.append(pos[k], lnprobs[k]) for k in range(nwalkers)]
        steps += new_step

        sys.stdout.write("Completed step {} of {}  \r".format(i, nsteps) )
        sys.stdout.flush()

    df = pd.DataFrame(steps)
    df.columns = ['temp', 'coeff', 'lnprob']

    print "Finished MCMC."
    print("Mean acceptance fraction: {0:.3f}".format(np.mean(sampler.acceptance_fraction)))

    # xlims, ylims = (priors_t['min'], priors_t['max']), (priors_coeff['min'], priors_coeff['max'])


    return (sampler, df)



def mcmc_full_driver(prev_run=None, nsteps=20, save=False):
    plt.close()

    # First, execute the MCMC run. If one is passed as an argument, don't.
    if not prev_run:
        sampler, df = run_mcmc(nsteps=nsteps)
    else:
        sampler, df = prev_run

    # Now make some plots.
    jp = sns.jointplot(x=sampler.flatchain[:,0], y=sampler.flatchain[:,1],
                       kind="kde")
    jp.set_axis_labels('Effective Temperature', r"$\Omega_0$", weight='bold')
    plt.tight_layout()
    if save:
        plt.savefig('prob2_kde.pdf')
        print "Saved plot to prob2_kde.pdf"
    else:
        plt.show()


    # Get the data for the SED.
```

```
    t = df[df['lnprob'] == max(df['lnprob'])]['temp'].values[0]
    coeff = df[df['lnprob'] == max(df['lnprob'])]['coeff'].values[0]
    freqs, model, data = spectrum(t, coeff, show=False)
    print "Got spectrum."

    plt.plot(freqs, model, 'or', label='Model')
    plt.plot(freqs, data, 'ob', label='Data')
    plt.plot(freqs, model, '-r')
    plt.plot(freqs, data, '-b')

    plt.xlabel('Frequency (THz)', weight='bold')
    plt.ylabel(r"F$_{\nu}$ (Jy)", weight='bold')

    plt.title("Best-Fit SED at T = {}".format(round(t, 2)), weight='bold')
    plt.legend()
    plt.tight_layout()
    sns.despine()

    if save:
        plt.savefig('prob2_spectrum.pdf')
        print "Saved plot to prob2_spectrum.pdf"
    else:
        plt.show()

mcmc_full_driver(save=True)




# Problem 3

test_coords = ['04h37m48s', '-0d21m25s']

pos_radec = ('13h42m25.6s', '-7d13m42.1s')
def radec_to_galactic_astropy(pos_radec):
    """
    Convert RA/dec coordinates to galactic (l, b) coordinates.

    Args: coords (tuple of strs): RA, dec values in a format understood
                                  by astropy.coordinates.Angle
    Returns: (l, b) tuple, in degrees.
    """
    ra_hms, dec_hms = Angle(pos_radec[0]), Angle(pos_radec[1])
    radec_coords_deg = SkyCoord(ra=ra_hms, dec=dec_hms, frame='icrs')
    galactic_coords_str = radec_coords_deg.transform_to('galactic').to_string
    ()
    galactic_coords_degs = [float(coord) for coord in galactic_coords_str.
    split(' ')]
    return galactic_coords_degs

def radec_to_galactic(coords):
    """
    Convert RA/dec coordinates to galactic (l, b) coordinates by hand.
```

```
    Sources:
        NGP coords taken from:
        https://en.wikipedia.org/wiki/Galactic_coordinate_system

        Conversion formula adapted from:
        http://www.atnf.csiro.au/people/Tobias.Westmeier/tools_coords.php
    Args:
        coords (tuple of strs): RA, dec values in a format understood
                                by astropy.coordinates.Angle
    Returns: (l, b) tuple, in degrees.
    """

    def gross_coords_to_rads(coords):
        ra, dec = coords
        coords = SkyCoord(ra=ra, dec=dec, frame='icrs')
        ra_rad, dec_rad = [float(a) * np.pi/180
                           for a in coords.to_string().split()]
        return (ra_rad, dec_rad)

    ra, dec = gross_coords_to_rads(coords)
    ra_NGP, dec_NGP = gross_coords_to_rads(['12h51m26.00s', '+27d 7m 42.0s'])
    l_NCP = 122.93 * np.pi/180

    b = np.arcsin(np.sin(dec_NGP) * np.sin(dec) \
                  + np.cos(dec_NGP) * np.cos(dec) \
                  * np.cos(ra - ra_NGP))

    x1 = np.cos(dec) * np.sin(ra - ra_NGP)
    x2 = np.cos(dec_NGP) * np.sin(dec) \
         - np.sin(dec_NGP) * np.cos(dec) * np.cos(ra - ra_NGP)

    # Arctan2 is basically a smart version of arctan(x1/x2)
    l = l_NCP - np.arctan2(x1, x2)

    # Convert to degrees and round out to 4 decs for prettiness.
    l, b = round(l * 180/np.pi, 4), round(b * 180/np.pi, 4)
    return [l, b]




# Kepler populations

def plot_planet_pops_by_stellar_type(cmap='inferno_r', spectral_grouping=False
    , save=True):

    # cmaps: viridis, Spectral, ocean, inferno

    host_stars_df = pd.read_csv('plotting_data.csv')

    if spectral_grouping:
        fig, ax = plt.subplots(figsize=(10, 6))
```

```python

        sns.countplot(x=host_stars_df['spectral_group'], palette=cmap, ax=ax)

        # sns.countplot(x=host_stars_df['spectral_group'], palette='ocean', ax=ax2)

        ax.legend().remove
        ax.set_ylabel('Number of Planets', weight='bold')
        ax.set_xlabel('Spectral Type', weight='bold')
        ax.set_title('Planet Populations by Spectral Type from Kepler', weight='bold')
        sns.despine()

        if save:
            plt.savefig('planet_dist_by_spectral_group.pdf')
        else:
            plt.show()
    else:
        labs = []
        for id in sorted(list(set(host_stars_df['spt_id']))):
            spt_name = host_stars_df[host_stars_df['spt_id'] == id]['spt'].values[0]
            labs.append(spt_name)

        fig, ax = plt.subplots(figsize=(14, 4))
        # fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 7))

        sns.countplot(x=host_stars_df['spt_id'], palette=cmap, ax=ax)

        ax.legend().remove
        ax.set_ylabel('Number of Planets', weight='bold')
        ax.set_xlabel('Spectral Type', weight='bold')
        ax.set_title('Planet Populations by Spectral Type from Kepler', weight='bold')
        ax.set_xticklabels(labs, rotation=45)

        sns.despine()

        if save:
            plt.savefig('planet_dist_by_spt.pdf')
        else:
            plt.show()


# Counter(host_stars_df['spt'])
# host_stars_df
# sorted(Counter(host_stars_df['spt']).values())
# host_stars_df['spt_id']/Counter(host_stars_df['spt_id']).values()


# stellar_info = pd.read_csv('spectraltype_data.csv')
# stellar_info[stellar_info['SpT'] == 'G2V']['Mv']

```

```
# The End
```

```
"""
Gather, clean, and organize data on stellar types. Drawn from:
http://www.pas.rochester.edu/~emamajek/EEM_dwarf_UBVIJHK_colors_Teff.txt
"""

import pandas as pd
import numpy as np
import cPickle
import kplr
import sys


def export_spectral_type_info():
    headers_list = filter(None, headers.split(' '))

    all_stars = o_stars + b_stars + a_stars + f_stars + g_stars + k_stars + m_stars
    df_unbound = []
    spt_id = 0
    for stellar_type in all_stars:
        d = {}
        # Get a list of feature values
        spt_features = filter(None, stellar_type.split(' '))
        for i in range(len(headers_list)):
            try:
                feature = float(spt_features[i])
            except ValueError:
                feature = spt_features[i]
                feature = np.nan if '..' in feature else feature


            d[headers_list[i]] = feature
        d['spt_id'] = spt_id
        spt_id += 1

        df_unbound.append(d)
    df = pd.DataFrame(df_unbound)

    df.to_csv('spectraltype_data.csv', index=False)
    print "Saved spectral type info to spectraltype_data.csv"

```

```python

def export_planets_to_pkl():
    client = kplr.API()
    planets = client.planets(koi_prad=">1.5") + client.planets(koi_prad="<=1.5")
    with open('planets.pkl', 'rb') as input_f:
        pickled_planets = cPickle.load(input_f)
    print "Saved planet info to planets.pkl"


def export_good_log(n_sources=-1):
    with open('planets.pkl', 'rb') as input_f:
        pickled_planets = cPickle.load(input_f)

    # Get SPT data:
    spt_info = pd.read_csv('spectraltype_data.csv')

    planets = pickled_planets if n_sources is -1 else pickled_planets[:n_sources]
    host_stars = []
    for i in range(len(planets)):
        p = planets[i]
        s = p.star
        j_min_k = s.kic_jmag - s.kic_hmag
        t_eff = s.kic_teff
        metallicity = s.kic_feh
        try:
            row = spt_info[abs(spt_info['Teff'] - s.kic_teff) == min(abs(spt_info['Teff'] - s.kic_teff))]
            spt = row['SpT'].values[0]
            spt_id = row['spt_id'].values[0]
            predicted_teff = row['Teff'].values[0]
            spectral_group = spt[0]
        except TypeError:
            pass

        d = {'t_eff': s.kic_teff,
             'metallicity': s.kic_feh,
             # 'j_min_k': s.kic_jmag - s.kic_hmag,
             # 'predicted_teff': predicted_teff,
             'spt': spt,
             'spectral_group': spectral_group,
             'spt_id': spt_id}
        host_stars.append(d)

        pct_complete = round(100 * i/float(len(planets)), 2)
        # sys.stdout.write("Gotten %i% stars  \r" % (i) )
        sys.stdout.write("Gathered info for {}% of the stars  \r".format(pct_complete) )
        sys.stdout.flush()
    host_stars_df = pd.DataFrame(host_stars)
    host_stars_df.to_csv('plotting_data.csv')
    print "Saved plotting data to plotting_data.csv"
```

```
### THE ACTUAL, RAW TABLE FROM THE LINKED WEBSITE.
# Real ugly, gets processed in the functions above.
headers = 'SpT    Teff   logT    BCv     Mv     logL   B-V     Bt-Vt   G-V     U-B   V-Rc    V-Ic     V-Ks      J-H       H-Ks      Ks-W1    W1-W2   W1-W3   W1-W4   Msun   logAge  b-y      M_J     M_Ks    Mbol  i-z   z-Y   R_Rsun   SpT'


o_stars = [
    'O3V     46000  4.663  -4.05   -5.7    5.80  -0.32    ...      ...    -1.22      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   .....    ....     ....   -9.75  ...    ...   12.5     O3V',
    'O4V     43000  4.633  -3.92   -5.5    5.67  -0.32    ...      ...    -1.20      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   .....    ....     ....   -9.42  ...    ...   12.3     O4V',
    'O5V     41500  4.618  -3.77   -5.4    5.58  -0.32    ...      ...    -1.19      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   -0.133   ....     ....   -9.20  ...    ...   11.9     O5V',
    'O5.5V   40000  4.602  -3.65   -5.2    5.46  -0.32    ...      ...    -1.18      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   -0.133   ....     ....   -8.90  ...    ...   11.2     O5.5V',
    'O6V     39000  4.591  -3.57   -5.1    5.37  -0.32    ...      ...    -1.17      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   -0.132   ....     ....   -8.67  ...    ...   10.6     O6V',
    'O6.5V   37300  4.573  -3.45   -4.9    5.24  -0.32    ...      ...    -1.16      ...   .....    .....    .....    .....    ...      ...     ...      ...      ....    ...   -0.131   ....     ....   -8.35  ...    ...   10.0     O6.5',
    'O7V     36500  4.562  -3.38   -4.8    5.17  -0.32    ...      ...    -1.15      ...   .....    .....    .....    .....    ...      ...     ...      ...      28      ...   -0.130   ....     ....   -8.18  ...    ...   9.62     O7V',
    'O7.5V   35000  4.544  -3.27   -4.6    5.05  -0.32    ...      ...    -1.14      ...   .....    .....    .....    .....    ...      ...     ...      ...      24      ...   -0.130   ....     ....   -7.87  ...    ...   9.11     O7.5V',
    'O8V     34500  4.538  -3.22   -4.5    4.99  -0.32    ...      ...    -1.13      ...   .....    .....    .....    .....    ...      ...     ...      ...      22.9    ...   -0.129   ....     ....   -7.72  ...    ...   8.75     O8V',
    'O8.5V   33000  4.519  -3.13   -4.3    4.87  -0.32    ...      ...    -1.12      ...   .....    .....    .....    .....    ...      ...     ...      ...      20.5    ...   -0.128   ....     ....   -7.43  ...    ...   8.33     O8.5V',
    'O9V     32500  4.512  -3.09   -4.2    4.82  -0.318   ...      ...    -1.114     ...   -0.369   -1.000   -0.164   -0.071   ...      ...     ...      ...      19.7    6.6   -0.127   -3.44    -3.20  -7.29  ...    ...   8.11     O9V',
    'O9.5V   32000  4.505  -3.06   -4.1    4.76  -0.312   ...      ...    -1.087     ...   -0.361   -0.977   -0.161   -0.069   ...      ...     ...      ...      18.5    6.7   -0.125   -3.35    -3.12  -7.15  ...    ...   7.80     O9.5V']


b_stars = [
    'B0V     31500  4.498  -3.02   -4.0    4.70  -0.307   ...      ...    -1.067     ...   -0.355   -0.958   -0.159   -0.067   ...      ...     ...      ...      17.5    6.8   -0.122   -3.27    -3.04  -7.02  ...    ...   7.53     B0V',
    'B0.5V   29000  4.462  -2.87   -3.6    4.47  -0.295   ...      ...    -1.026     ...
```

```
    -0.338   -0.913   -0.153   -0.063    ...      ...      ...      ...      15     6.9
   -0.120   -2.90    -2.69    -6.43   ...    ...   6.81     B0.5V  ',
    'B1V      26000  4.415  -2.61    -3.1     4.13  -0.278   ...      ...     -0.995   -0.115
    -0.325   -0.874   -0.148   -0.059    ...      ...      ...      ...      11     7.0
   -0.113   -2.43    -2.23    -5.68   ...    ...   5.72     B1V',
    'B1.5V   24500  4.389  -2.43    -2.8     3.89  -0.252  -0.274   ...     -0.910   -0.114
    -0.281   -0.752   -0.132   -0.047    0.035    ...      ...      ...      10     7.1
   -0.103   -2.23    -2.05    -5.24   ...    ...   4.89     B1.5V',
    'B2V      20600  4.314  -2.06    -1.7     3.38  -0.210  -0.219   ...     -0.790   -0.094
    -0.230   -0.602   -0.113   -0.032    0.036    ...      ...      ...      7.3    7.2
   -0.094   -1.24    -1.10    -3.71   ...    ...  3.85     B2V',
    'B2.5V   18500  4.267  -1.79    -1.4     3.10  -0.198  -0.206   ...     -0.732   -0.087
    -0.210   -0.544   -0.105   -0.026    0.036    ...      ...      ...      6.1    7.3
   -0.087   -0.99    -0.86    -3.15   ...    ...   3.45     B2.5V',
    'B3V      17000  4.230  -1.58    -1.1     2.96  -0.178  -0.184   ...     -0.673   -0.080
    -0.192   -0.492   -0.098   -0.021    0.036    ...      ...      ...      5.4    7.4
   -0.083   -0.73    -0.61    -2.64   ...    ...   3.48     B3V',
    'B4V      16700  4.223  -1.53    -1.0     2.91  -0.165  -0.170   ...     -0.619   -0.074
    -0.176   -0.447   -0.092   -0.016    0.036    ...      ...      ...      5.0    7.5
   -0.078   -0.66    -0.55    -2.52   ...    ...   3.41     B4V',
    'B5V      15700  4.196  -1.35    -0.9     2.80  -0.156  -0.160   ...     -0.581   -0.070
    -0.165   -0.417   -0.089   -0.013    0.036   -0.045  -0.117  -0.070   4.6    7.7
   -0.072   -0.59    -0.48    -2.25   ...    ...   3.40     B5V',
    'B6V      14500  4.161  -1.16    -0.5     2.54  -0.140  -0.142   ...     -0.504   -0.062
    -0.145   -0.358   -0.081   -0.007    0.035   -0.045  -0.117  -0.070   4.0    7.8
   -0.066   -0.23    -0.14    -1.66   ...    ...   2.95     B6V',
    'B7V      14000  4.146  -1.07    -0.4     2.49  -0.128  -0.129   ...     -0.459   -0.058
    -0.133   -0.325   -0.077   -0.004    0.035   -0.045  -0.117  -0.070   3.9    7.9
   -0.062   -0.16    -0.08    -1.47   ...    ...   2.99     B7V',
    'B8V      12500  4.097  -0.81    -0.2     2.27  -0.109  -0.107   ...     -0.364   -0.048
    -0.108   -0.254   -0.067    0.003    0.034   -0.046  -0.116  -0.067   3.4    8.1
   -0.045   -0.01     0.05    -0.91   ...    ...   2.91     B8V',
    'B9V      10700  4.029  -0.42     0.7     1.79  -0.070  -0.063   ...     -0.200   -0.028
    -0.061   -0.121   -0.050    0.016    0.032   -0.063  -0.104  -0.054   2.8    8.3
   -0.029    0.79     0.82     0.28   ...    ...   2.28     B9V',
    'B9.5V   10400  4.017  -0.38     0.8     1.73  -0.050  -0.040   ...     -0.130   -0.017
    -0.035   -0.048   -0.044    0.021    0.031   -0.044  -0.091  -0.043   2.5    8.4
   -0.021    0.83     0.85     0.44   ...    ...   2.26     B9.5V']

a_stars = [
    'A0V      9700   3.987  -0.24     1.11    1.54   0.000   0.013  -0.023  -0.005    0.001
     0.004    0.041   -0.032    0.028    0.030   -0.041  -0.074  -0.022   2.3    8.5
    0.000    1.07     1.07     0.87   ...    ...   2.09     A0V',
    'A1V      9200   3.965  -0.15     1.34    1.41   0.043   0.056  -0.024   0.033    0.019
     0.044    0.101   -0.024    0.031    0.030   -0.036  -0.068  -0.023   2.15   8.6
    0.017    1.25     1.24     1.19   ...    ...   2.00     A1V',
    'A2V      8840   3.946  -0.10     1.48    1.33   0.074   0.091  -0.025   0.063    0.042
     0.091    0.188   -0.010    0.034    0.029   -0.034  -0.067  -0.029   2.05   8.7
    0.038    1.32     1.29     1.38   ...    ...   1.97     A2V',
    'A3V      8550   3.932  -0.06     1.55    1.29   0.090   0.109  -0.026   0.077    0.050
     0.108    0.228   -0.002    0.034    0.029   -0.033  -0.066  -0.029   2.00   8.7
    0.055    1.35     1.32     1.49   ...    ...   2.01     A3V',
    'A4V      8270   3.917  -0.04     1.76    1.20   0.140   0.166  -0.029   0.097    0.078
```

```
      0.164    0.353    0.022    0.037    0.029   −0.031  −0.059  −0.021   1.90   8.8
     0.071    1.47    1.41    1.72  ...    ...  1.94     A4V',
     'A5V    8080   3.907  −0.03    1.84   1.16   0.160   0.185  −0.031   0.100    0.089
      0.186    0.403    0.031    0.038    0.029   −0.030  −0.058  −0.021   1.85   8.8
     0.090    1.51    1.44    1.81  ...    ...  1.94     A5V',
     'A6V    8000   3.903  −0.02    1.89   1.14   0.170   0.194  −0.032   0.098    0.094
      0.197    0.428    0.036    0.038    0.029   −0.030  −0.057  −0.018   1.83   8.9
     0.099    1.54    1.46    1.87  ...    ...  1.93     A6V',
     'A7V    7800   3.892   0.00    2.07   1.06   0.210   0.233  −0.037   0.091    0.117
      0.242    0.528    0.055    0.040    0.029   −0.030  −0.056  −0.017   1.76   8.9
     0.107    1.64    1.54    2.07  ...    ...  1.86     A7V',
     'A8V    7500   3.874   0.00    2.29   0.97   0.250   0.274  −0.042   0.082    0.140
      0.288    0.626    0.075    0.042    0.028   −0.028  −0.055  −0.009   1.67   9.0
     0.132    1.78    1.66    2.29  ...    ...  1.81     A8V',
     'A9V    7440   3.872   0.00    2.30   0.97   0.255   0.279  −0.043   0.080    0.143
      0.294    0.638    0.078    0.043    0.028   −0.028  −0.055  −0.007   1.67   9.0
     0.145    1.78    1.66    2.30  ...    ...  1.84     A9V']

f_stars = [
     'F0V    7220   3.857  −0.01    2.51   0.89   0.294   0.317  −0.049   0.053    0.166
      0.339    0.732    0.098    0.045    0.028   −0.026  −0.050   0.003   1.59   9.1
     0.158    1.90    1.76    2.50  ...    ...  1.79     F0V',
     'F1V    7030   3.847  −0.01    2.79   0.77   0.334   0.350  −0.057   0.021    0.190
      0.385    0.828    0.119    0.047    0.028   −0.026  −0.047   0.009   1.50   9.1
     0.204    2.13    1.96    2.78  ...    ...  1.64     F1V',
     'F2V    6810   3.833  −0.02    2.99   0.70   0.374   0.390  −0.066  −0.008    0.213
      0.432    0.925    0.140    0.050    0.028   −0.027  −0.046   0.011   1.44   9.2
     0.250    2.26    2.07    2.97  ...    ...  1.61     F2V',
     'F3V    6720   3.827  −0.03    3.08   0.67   0.389   0.405  −0.069  −0.016    0.222
      0.449    0.961    0.147    0.051    0.028   −0.028  −0.046   0.008   1.43   9.2
     0.263    2.32    2.12    3.05  ...    ...  1.60     F3V',
     'F4V    6640   3.822  −0.04    3.23   0.61   0.412   0.428  −0.075  −0.026    0.236
      0.476    1.017    0.159    0.052    0.028   −0.029  −0.046   0.000   1.39   9.2
     0.277    2.42    2.21    3.19  ...    ...  1.53     F4V',
     'F5V    6510   3.814  −0.04    3.40   0.54   0.438   0.455  −0.081  −0.029    0.252
      0.506    1.079    0.173    0.054    0.028   −0.030  −0.045  −0.004   1.33   9.3
     0.290    2.55    2.32    3.36  ...    ...  1.46     F5V',
     'F6V    6340   3.802  −0.05    3.70   0.43   0.484   0.504  −0.093  −0.021    0.276
      0.553    1.185    0.199    0.057    0.028   −0.033  −0.045  −0.012   1.25   9.4
     0.317    2.78    2.52    3.65  ...    ...  1.36     F6V',
     'F7V    6240   3.795  −0.06    3.87   0.36   0.510   0.534  −0.099  −0.012    0.290
      0.579    1.244    0.213    0.060    0.027   −0.036  −0.045  −0.013   1.21   9.5
     0.332    2.90    2.63    3.81  ...    ...  1.30     F7V',
     'F8V    6170   3.790  −0.07    4.01   0.31   0.530   0.558  −0.105   0.001    0.300
      0.599    1.290    0.225    0.061    0.027   −0.039  −0.044  −0.016   1.18   9.6
     0.350    3.01    2.72    3.96  ...    ...  1.25     F8V',
     'F9V    6060   3.782  −0.08    4.15   0.26   0.552   0.587  −0.111   0.014    0.312
      0.620    1.340    0.237    0.063    0.027   −0.041  −0.044  −0.014   1.14   9.7
     0.378    3.11    2.81    4.07  ...    ...  1.23     F9V',
     'F9.5V  6000   3.778  −0.08    4.29   0.21   0.572   0.615  −0.117   0.033    0.323
      0.640    1.385    0.249    0.065    0.027   −0.042  −0.043  −0.012   1.11   9.7
      ,..     3.22    2.91    4.22  ...    ...  1.18     F9.5V']
```

```
g_stars = [
    'G0V    5920  3.772  -0.09   4.45   0.14   0.596   0.650  -0.124   0.058    0.336
     0.664   1.440    0.262    0.067    0.027   -0.043  -0.043  -0.010   1.08   9.7
    ...      3.34    3.01    4.40   ...    ...   1.12    G0V',
    'G1V    5880  3.769  -0.10   4.50   0.13   0.604   0.661  -0.133   0.067    0.340
     0.672   1.458    0.267    0.068    0.027   -0.044  -0.042  -0.010   1.07   9.8
    ...      3.38    3.04    4.40   ...    ...   1.12    G1V',
    'G2V    5770  3.761  -0.11   4.79   0.01   0.650   0.724  -0.141   0.133    0.363
     0.713   1.564    0.293    0.073    0.028   -0.050  -0.040  -0.016   1.02   9.8
    ...      3.57    3.20    4.68   ...    ...   1.01    G2V',
    'G3V    5720  3.757  -0.12   4.86  -0.01   0.661   0.739  -0.144   0.152    0.368
     0.722   1.590    0.299    0.074    0.028   -0.050  -0.040  -0.014   1.00   ...
    ...      3.64    3.27    4.74   ...    ...   1.01    G3V',
    'G4V    5680  3.754  -0.13   4.94  -0.04   0.674   0.757  -0.148   0.175    0.374
     0.733   1.621    0.307    0.075    0.028   -0.052  -0.041  -0.014   0.99   ...
    ...      3.70    3.32    4.80   ...    ...   0.986   G4V',
    'G5V    5660  3.753  -0.13   4.98  -0.05   0.680   0.764  -0.150   0.185    0.377
     0.738   1.635    0.310    0.076    0.028   -0.052  -0.041  -0.014   0.98   ...
    ...      3.74    3.35    4.84   ...    ...   0.982   G5V',
    'G6V    5590  3.747  -0.15   5.13  -0.11   0.704   0.796  -0.158   0.227    0.388
     0.758   1.691    0.324    0.079    0.028   -0.053  -0.040  -0.012   0.97   ...
    ...      3.84    3.44    4.98   ...    ...   0.939   G6V',
    'G7V    5530  3.743  -0.16   5.18  -0.12   0.713   0.809  -0.161   0.243    0.393
     0.766   1.712    0.329    0.080    0.028   -0.054  -0.040  -0.013   0.96   ...
    ...      3.88    3.47    5.02   ...    ...   0.949   G7V',
    'G8V    5490  3.734  -0.17   5.32  -0.17   0.737   0.842  -0.169   0.284    0.404
     0.786   1.768    0.342    0.082    0.028   -0.057  -0.039  -0.018   0.94   ...
    ...      3.97    3.55    5.15   ...    ...   0.909   G8V',
    'G9V    5340  3.728  -0.21   5.55  -0.25   0.777   0.894  -0.182   0.358    0.423
     0.820   1.861    0.365    0.087    0.029   -0.060  -0.038  -0.018   0.90   ...
    ...      4.14    3.69    5.34   ...    ...   0.876   G9V']


k_stars = [
    'K0V    5280  3.723  -0.22   5.76  -0.33   0.816   0.944  -0.23    0.436    0.443
     0.853   1.953    0.387    0.091    0.030   -0.063  -0.037  -0.015   0.87   ...
    ...      4.29    3.81    5.54   ...    ...   0.817   K0V',
    'K0.5V  5240  3.719  -0.23   5.80  -0.33   0.828   0.955  -0.24    0.456    0.448
     0.862   1.977    0.393    0.092    0.030   -0.064  -0.038  -0.018   0.86   ...
    ...      4.31    3.82    5.57   ...    ...   0.828   K0.5V',
    'K1V    5170  3.713  -0.26   5.89  -0.36   0.847   0.976  -0.26    0.491    0.457
     0.879   2.021    0.402    0.094    0.030   -0.064  -0.038  -0.013   0.85   ...
    ...      4.37    3.87    5.65   ...    ...   0.814   K1V',
    'K1.5V  5140  3.711  -0.27   5.97  -0.39   0.859   0.999  -0.28    0.528    0.467
     0.895   2.066    0.412    0.096    0.030   -0.063  -0.037  -0.016   0.82   ...
    ...      4.41    3.90    5.70   ...    ...   0.809   K1.5V',
    'K2V    5040  3.702  -0.29   6.19  -0.47   0.884   1.035  -0.30    0.581    0.482
     0.920   2.132    0.427    0.098    0.031   -0.068  -0.037  -0.009   0.78   ...
    ...      4.57    4.04    5.88   ...    ...   0.763   K2V',
    'K2.5V  4990  3.698  -0.32   6.34  -0.51   0.938   1.101  -0.30    0.691    0.513
     0.974   2.274    0.459    0.104    0.032   -0.071  -0.032  -0.006   0.76   ...
    ...      4.63    4.07    6.02   ...    ...   0.742   K2.5V',
    'K3V    4830  3.684  -0.41   6.57  -0.58   0.980   1.150  -0.31    0.776    0.537
```

```
    1.013   2.380   0.483   0.109   0.033   -0.071  -0.031  -0.008   0.75   ...
    ...      4.76    4.16    6.16  ...   ...   0.729   K3V',
    'K3.5V  4700   3.672  -0.45    6.79  -0.64   1.050   1.239  -0.37    0.917    0.592
    1.108   2.582   0.520   0.118   0.036   -0.074  -0.031  -0.005   0.73   ...
    ...      4.85    4.21    6.34  ...   ...   0.720   K3.5V',
    'K4V    4600   3.663  -0.56    6.98  -0.67   1.100   1.306  -0.43    1.004    0.640
    1.190   2.733   0.544   0.125   0.039   -0.073  -0.031   0.009   0.72   ...
    ...      4.92    4.25    6.42  ...   ...   0.726   K4V',
    'K4.5V  4540   3.657  -0.60    7.04  -0.68   1.116   1.328  -0.42    1.028    0.654
    1.216   2.781   0.552   0.127   0.040   -0.073  -0.030   0.017   0.71   ...
    ...      4.94    4.26    6.44  ...   ...   0.737   K4.5V',
    'K5V    4410   3.644  -0.68    7.36  -0.78   1.150   1.373  -0.41    1.081    0.685
    1.272   2.883   0.568   0.132   0.042   -0.073  -0.029   0.019   0.68   ...
    ...      5.18    4.48    6.68  ...   ...   0.698   K5V',
    'K5.5V  4330   3.636  -0.75    7.60  -0.84   1.200   1.414  -0.44    1.144    0.728
    1.357   3.034   0.591   0.139   0.045    ...     ...     ...     0.66   ...
    ...      5.30    4.57    6.85  ...   ...   0.672   K5.5V',
    'K6V    4230   3.626  -0.81    7.80  -0.90   1.240   1.439  -0.48    1.184    0.759
    1.420   3.143   0.601   0.148   0.049    ...     ...     ...     0.65   ...
    ...      5.41    4.66    6.99  ...   ...   0.661   K6V',
    'K6.5V  4190   3.622  -0.92    8.01  -0.96   1.310   1.491  -0.52    1.213    0.796
    1.505   3.288   0.613   0.159   0.055    ...     ...     ...     0.64   ...
    ...      5.56    4.78    7.09  ...   ...   0.656   K6.5V',
    'K7V    4070   3.610  -0.95    8.15  -0.98   1.330   1.506  -0.57    1.221    0.806
    1.529   3.330   0.617   0.162   0.057    ...     ...     ...     0.63   ...
    ...      5.60    4.82    7.18  ...   ...   0.654   K7V',
    'K8V    4000   3.602  -1.03    8.47  -1.10   1.356   1.562  -0.63    1.216    0.843
    1.632   3.487   0.623   0.176   0.081    ...     ...     ...     0.59   ...
    ...      5.78    4.98    7.44  ...   ...   0.587   K8V',
    'K9V    3940   3.595  -1.10    8.69  -1.18   1.382   1.593  -0.69    1.210    0.866
    1.699   3.584   0.625   0.184   0.101    ...     ...     ...     0.56   ...
    ...      5.92    5.11    7.59  ...   ...   0.552   K9V']

m_stars = [
    'M0V    3870   3.588  -1.16    8.91  -1.20   1.408   1.623  -0.65    1.204    0.889
    1.766   3.680   0.626   0.193   0.122    ...     ...     ...     0.55   ...
    ...      6.04    5.22    7.75  0.33  ...   0.559   M0V',
    'M0.5V  3800   3.580  -1.38    9.20  -1.27   1.441    ...    -0.74    1.184    0.924
    1.886   3.84    0.620   0.208   0.130    ...     ...     ...     0.54   ...
    ...      6.19    5.36    7.90  0.37  ...   0.535   M0.5V',
    'M1V    3700   3.568  -1.44    9.69  -1.40   1.475    ...    -0.82    1.172    0.959
    2.019   4.02    0.613   0.225   0.137    ...     ...     ...     0.49   ...
    ...      6.51    5.67    8.25  0.41  ...   0.496   M1V',
    'M1.5V  3650   3.562  -1.57    9.97  -1.47   1.486    ...    -0.85    1.170    0.978
    2.089   4.12    0.607   0.228   0.105    ...     ...     ...     0.47   ...
    ...      6.69    5.85    8.40  0.47  ...   0.460   M1.5V',
    'M2V    3550   3.550  -1.65   10.30  -1.57   1.500    ...    -0.92    1.170    1.001
    2.173   4.24    0.600   0.234   0.110    ...     ...     ...     0.44   ...
    ...      6.89    6.06    8.65  0.53  ...   0.434   M2V',
    'M2.5V  3500   3.544  -1.76   10.70  -1.68   1.522    ...    -1.02    1.175    1.041
    2.306   4.43    0.589   0.244   0.117    ...     ...     ...     0.40   ...
    ...      7.01    6.27    8.94  0.57  ...   0.393   M2.5V',
    'M3V    3410   3.533  -1.97   11.14  -1.78   1.544    ...    -1.09    1.181    1.079
```

```
     2.420   4.60    0.579   0.252   0.122    ...     ...     ...     0.36   ...
    ...      7.40    6.54    9.17  0.61  ...   0.369    M3V',
    'M3.5V  3250  3.512  -2.27   12.19  -2.07   1.602   ...    -1.29   1.200   1.178
     2.680   5.00    0.558   0.269   0.132    ...     ...     ...     0.26   ...
    ...      8.02    7.19    9.92  0.66  ...   0.291    M3.5V',
    'M4V    3200  3.505  -2.59   12.80  -2.20   1.661   ...    -1.41   1.222   1.241
     2.831   5.25    0.557   0.282   0.139    ...     ...     ...     0.22   ...
    ...      8.39    7.55   10.21  0.71  ...   0.258    M4V',
    'M4.5V  3100  3.491  -3.05   13.57  -2.31   1.72    ...    -1.55   1.23    1.345
     3.073   5.64    0.564   0.301    ...     ...     ...     ...     0.18   ...
    ...      8.79    7.93   10.52  0.81  ...   0.243    M4.5V',
    'M5V    3030  3.481  -3.28   14.30  -2.52   1.874   ...    -1.74   1.24    1.446
     3.277   5.94    0.580   0.311    ...     0.17    ...     ...     0.16   ...
    ...      9.25    8.36   11.02  0.91  0.47  0.199    M5V',
    'M5.5V  3000  3.477  -3.80   15.51  -2.79   1.91    ...    -2.01   1.3     1.656
     3.664   6.50    0.588   0.329    ...     0.19    ...     ...     0.12   ...
    ...      9.93    9.01   11.71  1.13  0.52  0.149    M5.5V',
    'M6V    2850  3.455  -4.36   16.62  -3.02   2.00    ...    -2.14   1.3     1.950
     4.13    7.30    0.605   0.352    ...     0.21    ...     ...     0.10   ...
    ...     10.28    9.32   12.26  1.45  0.60  0.127    M6V',
    'M6.5V  2710  3.433  -4.60   17.07  -3.09   2.06    ...    -2.75   ...     2.003
     4.31    7.60    0.609   0.364    ...     0.22    ...     ...     0.097  ...
    ...     10.47    9.47   12.47  1.58  0.64  0.129    M6.5V',
    'M7V    2650  3.423  -5.06   17.81  -3.21   2.06    ...    -2.98   ...     2.180
     4.45    8.05    0.613   0.386    ...     0.24    ...     ...     0.090  ...
    ...     10.76    9.76   12.75  1.77  0.70  0.118    M7V',
    'M7.5V  2600  3.415  -5.46   18.42  -3.29   2.17    ...    -3.09   ...     2.160
     4.56    8.45    0.650   0.422    ...     0.25    ...     ...     0.089  ...
    ...     10.68    9.97   12.97  1.85  0.74  0.112    M7.5V',
    'M8V    2500  3.398  -5.70   18.84  -3.36   2.20    ...    -3.11   ...     2.150
     4.64    8.73    0.670   0.450    ...     0.26    ...     ...     0.082  ...
    ...     11.23   10.11   13.14  1.93  0.77  0.111    M8V',
    'M8.5V  2440  3.387  -5.80   19.14  -3.44    ...    ...    -3.09   ...     1.967
     4.71    8.92    0.685   0.470    ...     0.265   ...     ...     0.081  ...
    ...     11.45   10.22   13.34  1.96  0.80  0.107    M8.5V',
    'M9V    2400  3.380  -5.90   19.36  -3.57    ...    ...    -3.07   ...     1.890
     4.75    9.00    0.749   0.480    ...     0.27    ...     ...     0.079  ...
    ...     11.53   10.30   13.67  1.99  0.82  0.095    M9V',
    'M9.5V  2320  3.365  -6.13   19.75  -3.55    ...    ...    -3.10   ...     2.510
     4.79    9.30    0.826   0.505    ...     0.27    ...     ...     0.078  ...
    ...     11.78   10.45   13.62  2.00  0.84  0.104    M9.5V']


export_planets_to_pkl()
export_spectral_type_info()
export_good_log()



# The End
```