Design and Implementation of a Spectral Fitting Engine in Python

Retrieval and Analysis of Photometry Data from the SDSS Catalog to Spectroscopically Infer Stellar Distances

Caden Gobat[1] for Sylvain Guiriec[1,2]

[1] Department of Physics, George Washington Univ., Washington, DC 20052
e-mail: cgobat@gwu.edu
[2] NASA Goddard Space Flight Center, Greenbelt, MD 20771
e-mail: sguiriec@gwu.edu

Submitted in partial fulfillment of the requirements of the ASTR 3141 course at the George Washington University.

ABSTRACT

Context. Many characteristics of a star can be inferred through spectroscopy, including temperature and luminosity. These values, when cross-referenced with the star's spectral and luminosity classes, can be used to infer the distance to the star in question.
Aims. This project serves as an exercise in astrophysical data analysis using Python and in the process, aims to assess the viability of using spectroscopic analysis as an alternative to geometric computations for determining stellar distances.
Methods. A fitting engine is developed in Python that uses a non-linear least squares regression to reconstruct and fit stellar spectra based on photometric observations in the SDSS catalog. An analysis of the fitted spectrum is performed, and the star's position on the H-R diagram is used to infer its temperature and luminosity. From there, distance from Earth is calculated using integrated flux and the inverse square law.
Results. Only a small portion of the data produced satisfactory results. The engine converged on a good fit ($R^2 \geq 0.96$) for only 1168 of 13219 (~8.8%) of stars in this catalog excerpt, and predicted the temperature to within 1000K of that expected for the star's type for 7,804 of them. In general, these good fits produced results consistent with expectations based on the spectral class of the star. The distance to most objects was on the order of ~100 ly.
Conclusions. The combination of spectroscopic and statistical analysis is a powerful one, allowing for better insights and results across the board. The next step would be to cross-reference SDSS data with information from another catalog such as *Gaia*, at which point a more accurate distance measurement could be used to refine other derived properties of the stars.

Key words. Methods: data analysis – Surveys – Techniques: spectroscopic – Parallaxes – Python

1. Introduction

A vast quantity of information about astronomical targets can be unlocked through analysis of an object's spectrum. From temperature to chemical composition and other physical properties, spectral analysis is one of the only ways to unlock information about objects that are so far away so as to be otherwise inscrutable.

Described here are the methodology and results of a data analysis project to interpret stellar spectra, retrieved from the Sloan Digital Sky Survey, (overview in Blanton et al. 2017, technical summary in York et al. 2000) and infer information about the stars' physical properties. There are two main critical pieces of information that can be used to do this: the parameters of a star's spectrum, and its spectral and luminosity classes—essentially, its position on the H-R diagram. The shape of the blackbody spectrum reveals its surface temperature, while knowledge of its actual power output based on its theoretical luminosity allows for observed and modeled flux to be compared. The scaling factor by which the two differ can then be used to approximate the distance to the star.

First, the data must be retrieved and parsed into a readable format for analysis. Then, photometric magnitudes are converted into units of physical flux and the star's classification used to deduce its approximate luminosity. Finally, the observed flux is integrated to determine the total power received on Earth, which is compared to the idealized power output of the star. The flux-luminosity relationship can be employed to determine the distance associated with the apparent reduction in power.

The software developed for this project accomplishes all of the above in a highly streamlined manner, automatically reading and fitting stars in the SDSS catalog. The code written for this project, as well as supporting materials, is available at `https://github.com/cgobat/stellar-spectra`. Readers are encouraged to click this link and follow along in the notebook in order to fully appreciate its capabilities.

2. Data Collection

The Sloan Digital Sky Survey's data is collected by two facilities, one in each hemisphere. Both are 2.5-meter Ritchey-Chrétien optical telescopes. The full technical and optical specifications of the Sloan Foundation's telescope at Apache Point Observatory in Arizona are described by Gunn et al. (2006). The Irénée DuPont Telescope at Las Campañas Observatory in Chile is described by Bowen & Vaughan (1973).

The photometric data itself that is analyzed in this project is collected by an array of 2048 by 2048 pixel CCDs arranged in

six columns of five sensors. Each column corresponds to one of the five SDSS optical filter bands: u, g, r, i, and z. Per Gunn et al. (1998), the camera is capable of "carry[ing] out photometry essentially simultaneously in five color bands spanning the range accessible to silicon detectors on the ground." Relative magnitude observations in these five bands are the key to this project, as they will be used to interpolate a full spectrum for each star.

Using VizieR,[1] a web-based tool that provides open access to nearly 20,000 astronomical catalogs, an excerpt of data from the twelfth SDSS-IV release was queried and retrieved using ADQL, a derivative of SQL built for astronomy-related purposes. The query for the data used in this project is provided here:

```
-- output format : csv
SELECT TOP 99999 "V/147/sdss12".SDSS12,
"V/147/sdss12".objID, "V/147/sdss12"."Sp-ID",
"V/147/sdss12".RA_ICRS, "V/147/sdss12".DE_ICRS,
"V/147/sdss12".ObsDate, "V/147/sdss12".spType,
"V/147/sdss12".subCl, "V/147/sdss12".umag,
"V/147/sdss12".e_umag, "V/147/sdss12".gmag,
"V/147/sdss12".e_gmag, "V/147/sdss12".rmag,
"V/147/sdss12".e_rmag, "V/147/sdss12".imag,
"V/147/sdss12".e_imag, "V/147/sdss12".zmag,
"V/147/sdss12".e_zmag, "V/147/sdss12".pmRA,
"V/147/sdss12".pmDE, "V/147/sdss12".zsp,
"V/147/sdss12".e_zsp, "V/147/sdss12".chi2
FROM "V/147/sdss12"
WHERE "SpObjID"!=0 and "class"=6 and "Q"=3
```

This is essentially a list of the desired columns, with the addition of a filter that ensures that all the objects are stars (`"class"=6`) with measured spectrum (`"SpObjID"!=0`) of the highest quality observations (`"Q"=3`). In the end, the most important variables queried here are the plate and fiber ID (`Sp-ID`), the star's spectral class (`subCl`), and the apparent magnitude in each of the five bands, as well as the error associated these observations. All of this is pulled from the `V/147/sdss12` catalog, which contains SDSS' twelfth data release. This query returns the requested data as a .csv file, which can then be loaded into Python as a dataframe by way of the pandas module (McKinney 2010).

3. Data Processing

The first task in analyzing the data of a given star is the conversion of the five photometric magnitudes in the u, g, r, i, and z bands into physical fluxes. Luckily, Bessell et al. (1998) provides formulas to convert magnitudes in various bands into flux densities, which are conveniently already implemented in the PyAstronomy (Czesla et al. 2019) package for Python. However, the first obstacle is that this function is coded in such a way so as to take inputs in the much more common U, B, V, R_C, and I_C bands, so the SDSS *ugriz* magntiudes must first be converted. A number of publications have laid out formulas for conversion, but here the transformations provided by Jordi et al. (2006) will be used, as they apply generally to stars and are derived based partially on SDSS photometry itself, so are highly relevant to this application.

The idea is that by applying these transformations to the *ugriz* magnitudes, they are converted to what they would have been had they been observed in the $UBVR_CI_C$ bands, which are similar, but shifted slightly and with different overlaps. The following equations detail the relationships between the two systems. Solving for each of U, B, V, R, and I will provide the transformed magnitudes.

$$
\begin{aligned}
U-B &= (0.79\pm0.02)*(u-g)-(0.93\pm0.02)\\
U-B &= (0.52\pm0.06)*(u-g)\\
&+ (0.53\pm0.09)*(g-r)-(0.82\pm0.04)\\
B-g &= (0.175\pm0.002)*(u-g)+(0.150\pm0.003)\\
B-g &= (0.313\pm0.003)*(g-r)+(0.219\pm0.002)\\
V-g &= (-0.565\pm0.001)*(g-r)-(0.016\pm0.001)\\
V-I &= (0.675\pm0.002)*(g-i)+(0.364\pm0.002)\\
&\text{if } g-i\leq 2.1\\
V-I &= (1.11\pm0.02)*(g-i)-(0.52\pm0.05)\\
&\text{if } g-i>2.1\\
R-r &= (-0.153\pm0.003)*(r-i)-(0.117\pm0.003)\\
R-I &= (0.930\pm0.005)*(r-i)+(0.259\pm0.002)\\
I-i &= (-0.386\pm0.004)*(i-z)-(0.397\pm0.001)
\end{aligned}
$$

It is a fairly simple matter to implement these transformations as a Python function that receives the five *ugriz* observations and returns five corresponding $UBVR_CI_C$ magnitudes. With the magnitudes converted, the five data points can be passed into the flux conversion function. This function returns flux densities in CGS units of ergs per second per square centimeter per angstrom. However, for ease of use with respect to Planck's Law, MKS units are more desirable. To this end, the unit conversion shown in Eq. (1) is applied.

$$
\begin{aligned}
1\frac{\text{erg}}{\text{s}\cdot\text{cm}^2\cdot\text{Å}} = \frac{10^{-7}\ \text{J}}{\text{s}\cdot 10^{-4}\ \text{m}^2\cdot 0.1\ \text{nm}} &= \frac{10^{-2}\ \text{W}}{\text{m}^2\cdot\text{nm}}\\
100\times\frac{\text{erg}}{\text{s}\cdot\text{cm}^2\cdot\text{Å}} &= 1\frac{\text{W}}{\text{m}^2\cdot\text{nm}}
\end{aligned}
\tag{1}
$$

Thus it is shown that a simple scaling factor of 100 must be applied to the outputs of the function in order to move to MKS units.

4. Fitting Procedure

With the data now in flux units, the five points essentially simulate a small sample of a complete blackbody spectrum for the wavelengths that correspond to each of the five filters. Using a non-linear least squares regression—implemented through the `optimize` package in SciPy (Virtanen et al. 2019)—these data were used to fit a modified Planck function (Eq. 2) where the temperature and an additional scaling factor (`amp`) were left free to vary.

$$
B_\lambda(\lambda,T,\text{amp}) = \frac{2hc^2}{(\lambda\times10^{-9})^5}\times\frac{\text{amp}}{e^{\frac{hc}{(\lambda\times10^{-9})\cdot k_B\cdot T}}-1}
\tag{2}
$$

The data and model are passed to the `scipy.optimize.curve_fit` function, as well as some initial "ballpark" estimates that initialize the parameters close to the data. For these purposes, a temperature of 7000K and an amplitude modifier of 1×10^{-30} are fairly universally good starting places. The optimizer minimizes the sum of the squares of the differences between the model and data point by varying

[1] https://vizier.u-strasbg.fr/

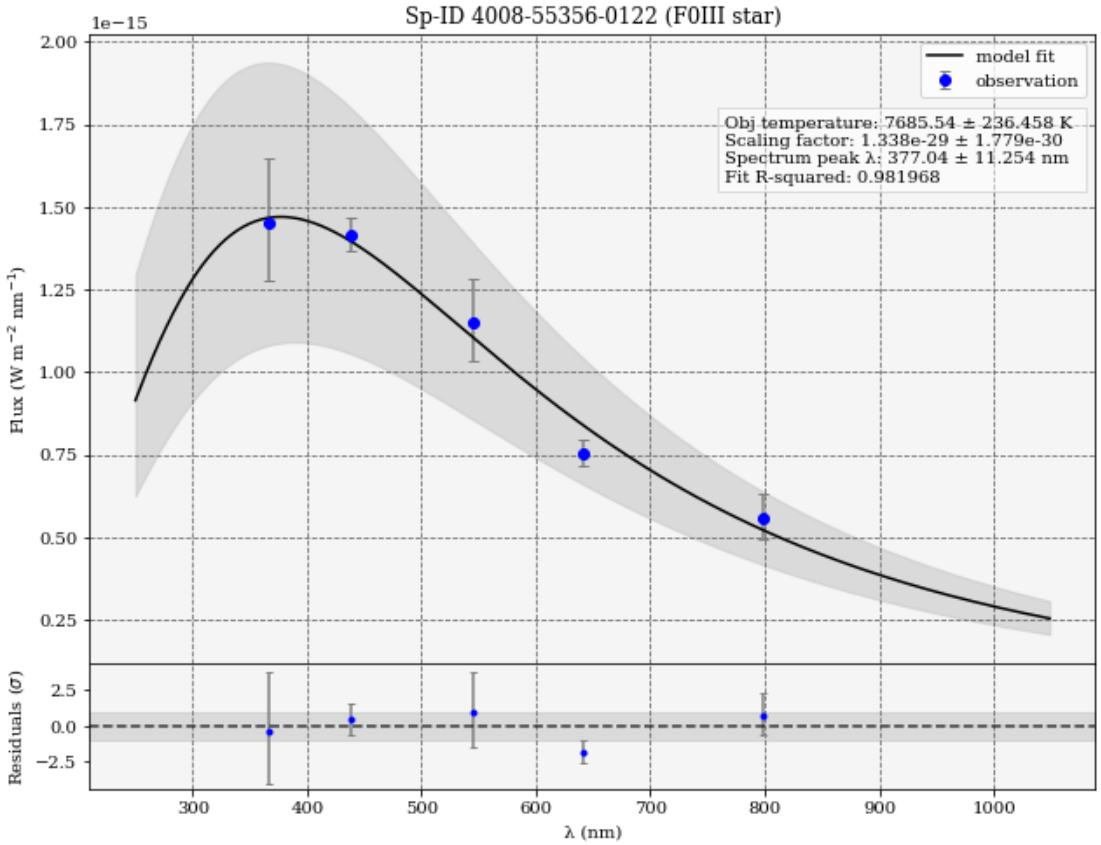

Fig. 1. Spectral curve generated using the five-band magnitudes recorded in the SDSS main catalog. The error bars on the data are propagated all the way from the original errors on the *ugriz* magnitude observations. The shaded confidence region on the main plot is computed using the uncertainties in the fit parameters, and the residual plot on the bottom is created by taking the differences between the data and fitted model and dividing them by σ, the standard deviation of the differences. There, the shaded bar is a visualization of the area within $\pm 1\sigma$ of the 0-line. The fit parameters and other properties of the object and model are displayed in the box towards the upper right corner of the graph. The software generates these plots automatically; as such, a similar plot could be created for any star in the catalog.

the amplitude and temperature variables, returning the values for these parameters that lead to the best fit (highest R^2 value) for the data. Knowing the temperature gives the peak wavelength of maximum flux, per Wien's Law (Eq. 3), and allows the modeled spectrum to be plotted.

$$\lambda_{\max} = \frac{b}{T} \text{ meters} \tag{3}$$

The curve fitting function also returns the covariance matrix of the fit, which is a matrix whose entries along the main diagonal correspond to the squares of the uncertainties in each of the varying parameters. Because there are two parameters free to vary in this application (temperature and amplitude), the covariance matrix is 2×2. Therefore, the square root of the first (upper left) entry is the uncertainty in the temperature optimization, while the square root of the last (lower right) entry is the uncertainty in the amplitude fit. The R-squared value to assess goodness of fit can be calculated by subtracting the ratio of the variance of the residuals to the variance of the data from one, where the residuals are the difference between the model and actual data, i.e. $R^2 = 1 - \frac{\text{Var}[data-model]}{\text{Var}[data]}$. The closer this indicator is to 1, the better the fit.

With a fitted model, known errors on the data, uncertainties for each optimized parameter, and residual errors, a plot of the results can be generated. An example of this is shown in Fig. (1). A similar plot can be created for any star in the catalog.

5. Analysis

In order to find the distance to a given star, now with a fitted model of its spectrum, its absolute luminosity must be determined. Given that the SDSS catalog does not provide this information directly, it must be inferred based on its spectral and luminosity classes—essentially, where it lies on the H-R diagram. To this end, calibrated formulas provided by de Jager & Nieuwenhuijzen (1987) and implemented in PyAstronomy will allow for the effective temperature and absolute luminosity to be calculated for any combination of spectral (O, B, A, F, G, K, M) and luminosity (I, II, III, IV, V) class. More information on these relationships is provided in Appendix A.

The software is written to automatically parse the spectral and luminosity classes from the dataframe, separate the two, then pass them to the `pyasl.SpecTypeDeJager().lumAndTeff()` function to obtain a luminosity and effective temperature for the star based on its classification. The fitting engine then compares this temperature value (T_{eff}) with that calculated through the fitting of the blackbody spectrum. As long as they are close enough to one another (the cutoff here was arbitrarily chosen to be within 1000K), the fit is considered valid and the program will proceed.

Distance is calculated by integrating the modeled blackbody spectrum to obtain a value for the total flux received from the star here on Earth, which is some fraction of the total power output of the star, related by the inverse square law flux-luminosity

relationship (Eq. 4). Knowing L and F, D is easily solved for (in Eq. 5), providing the distance from the star to Earth.

$$F = \frac{L}{4\pi D^2} \tag{4}$$

$$D = \sqrt{\frac{L}{4\pi F}} \tag{5}$$

6. Results & Generalizations

There is now a fully defined procedure to create and fit the spectrum of any star in the catalog. The next step is to scale the process into a mass data analysis exercise so as to calculate the properties of not just *any* star in the catalog, but *all* stars in the catalog. With the analytical code already written, it was relatively simple to wrap the constituent functions in a loop to iterate through the entire list of stars. For each object, the spectrum, temperature, luminosity, and distance were calculated as previously described. Then, if the R^2 value was higher than 0.96 (also somewhat arbitrarily chosen based on the visual aptness of the fits), the index of that object was added to a list (`goodfits`) as having a precise fit. Likewise, if the temperature derived from a star's spectrum was close to the temperature that a star of its classification should have, the object was logged in a separate list (`goodtemps`) that contained all the objects with fits whose results led to accurate parameters. Because the model took a significant amount of time[2] to run through the entire catalog excerpt of 99,999 stars, these two resultant lists of satisfactory object identifiers were saved externally to be more easily reloaded into the program for use in subsequent runs. The intersection of these two sets was taken as a master list (`all_good`) of all-around satisfactory objects; in other words, $all\_good = goodfits \cap goodtemps$.

This master list of successful fits contained just 630 entries—far less than the initial 99,999 objects pulled from VizieR. Between the culling of the raw data for only objects with all the necessary data present (13,219 stars met this criteria), only accepting temperatures within 1000K of the expected value (7,804 stars met this criteria), and limiting the R^2 values to ≥ 0.96 (1,168 stars met this criteria), the number of stars in use decreased at each step. Finally, intersecting the two lists led to there being only 630 elements present in both of them.

The overall distribution of temperature discrepancies (including those over 1000K) is shown in Fig. (2). Clearly, the distribution is not normal, as the histogram appears skewed to the right, meaning negative values are more prevalent than positive ones. Because ΔT was calculated for each star by subtracting its class-derived temperature from its fit-derived value, the skewness of the distribution indicates that the fitting model returned a temperature value that was less than the class-based calculation more often than not.

As for the distances to the stars, only those whose fits and temperatures were close can really be asserted with any confidence. For the 630 stars of which this is true, the distances ranged from 12.448 up to 52,224 light years, with a mean of 1543.9 ly and a median of 327.25 ly. This wide range and significant discrepancy between mean and median shows that the vast majority of the stars are located closer that ~1500 ly.

In the end, this project should be considered a success, as it accomplished its task of providing a data analysis challenge of appropriate complexity and turned out to inspire the creation of a fairly substantial code base capable of performing robust and rigorous computations and analyzing large datasets. To tie the results back to something else, and perhaps make them more accurate, a possible next step would be to cross-match the objects in this list with another catalog such as Bailer-Jones et al. (2018) to obtain a more accurate distance measurement, in turn refining the results for the other physical properties of the stars.

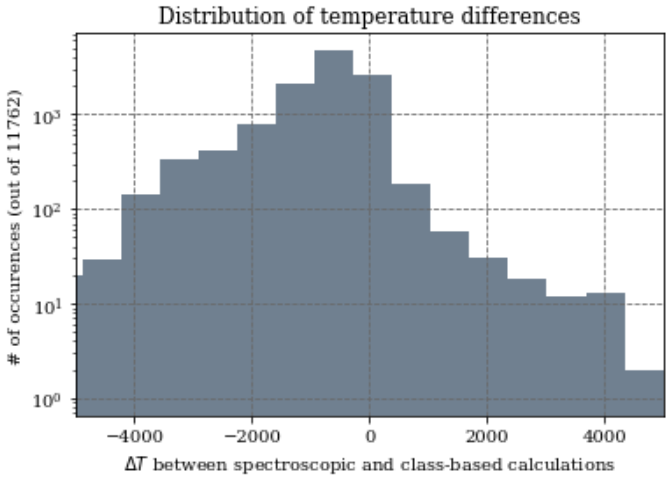

Fig. 2. Histogram of discrepancies between temperatures calculated using a fit of the star's spectrum and those calculated based on the star's spectral and luminosity classes. Here, ΔT = fit-derived temp − class-derived temp, showing that the spectral fitting method tended to underestimate the star's surface temperature relative to the class-based calculation.

Acknowledgements. Funding for the Sloan Digital Sky Survey IV has been provided by the Alfred P. Sloan Foundation, the U.S. Department of Energy Office of Science, and the Participating Institutions. SDSS-IV acknowledges support and resources from the Center for High-Performance Computing at the University of Utah. The SDSS web site is `www.sdss.org`. SDSS-IV is managed by the Astrophysical Research Consortium for the Participating Institutions of the SDSS Collaboration.

This research made use of Astropy,[3] a community-developed core Python package for Astronomy (Astropy Collaboration et al. 2013), as well as PyAstronomy,[4] a Python library of astronomy-related packages (Czesla et al. 2019).

This report was typeset using the A&A LaTeX macro package. The author acknowledges EDP Science and ESO for providing this resource.

References

Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33
Bailer-Jones, C., Rybizki, J., Fouesneau, M., Mantelet, G., & Andrae, R. 2018, AJ, 156, 58
Bessell, M. S., Castelli, F., & Plez, B. 1998, A&A, 333, 231
Blanton, M. R., Bershady, M. A., Abolfathi, B., et al. 2017, AJ, 154, 28
Bowen, I. S. & Vaughan, A. H. 1973, Appl. Opt., 12, 1430
Czesla, S., Schröter, S., Schneider, C. P., et al. 2019, PyA: Python astronomy-related packages
de Jager, C. & Nieuwenhuijzen, H. 1987, A&A, 177, 217
Gunn, J. E., Carr, M., Rockosi, C., et al. 1998, AJ, 116, 3040
Gunn, J. E., Siegmund, W. A., Mannery, E. J., et al. 2006, AJ, 131, 2332
Jordi, K., Grebel, E. K., & Ammon, K. 2006, A&A, 460, 339
McKinney, W. 2010, in Proceedings of the 9th Python in Science Conference, ed. S. van der Walt & J. Millman, 51 – 56
Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2019, arXiv e-prints, arXiv:1907.10121
York, D. G., Adelman, J., Anderson, John E., J., et al. 2000, AJ, 120, 1579

[2] To illustrate: computing fits and distances for the entire list of ~13,000 stars with usable data took over an hour (`01:04:19.22`) on a personal computer with an Intel i7 quad-core CPU running at 1.80 GHz.

[3] `http://www.astropy.org`
[4] `https://github.com/sczesla/PyAstronomy`

Appendix A: Class luminosity and temperature relationships

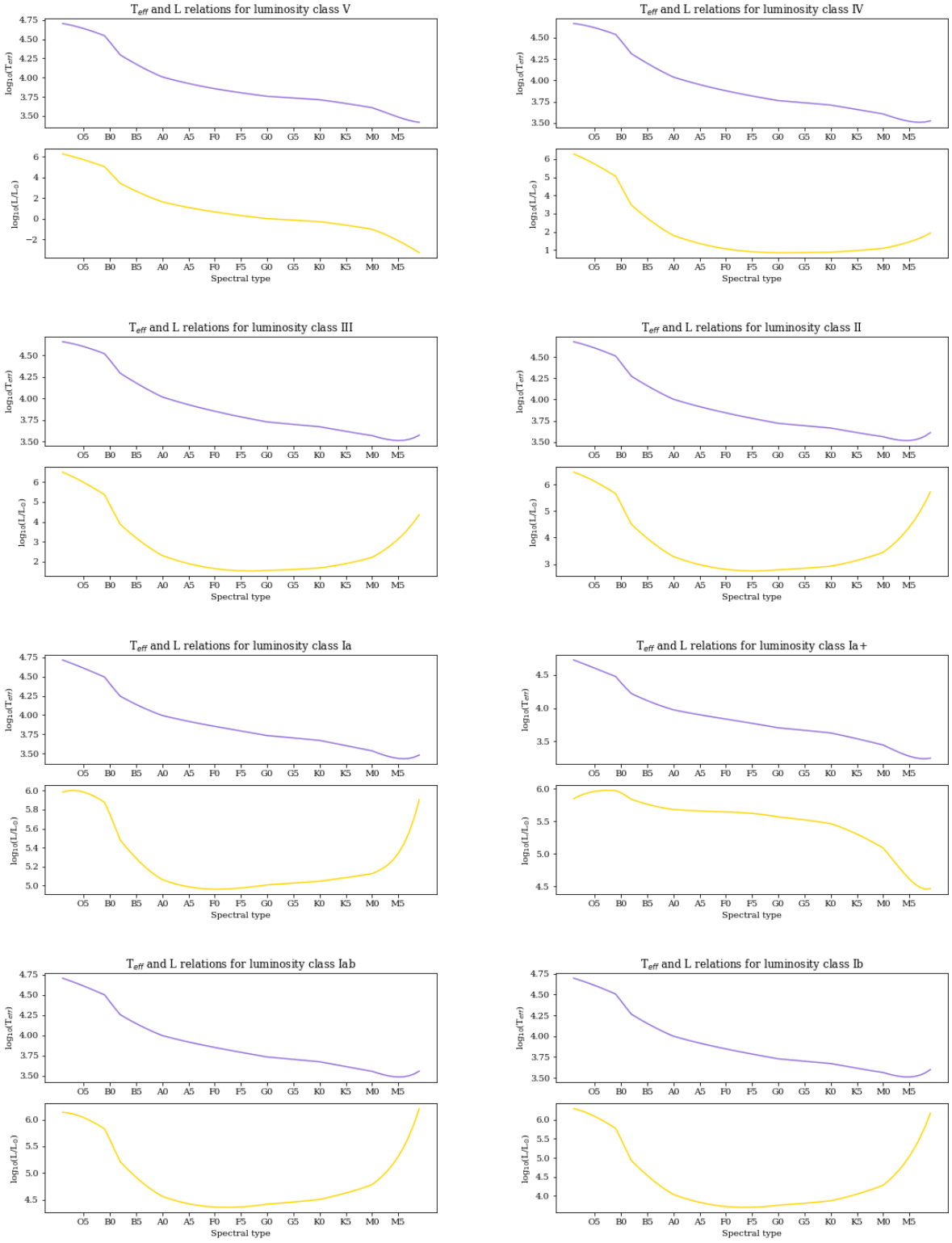

Fig. A.1. These graphs were generated using the formulas for the relations between spectral type and stellar luminosity and effective temperature (de Jager & Nieuwenhuijzen 1987) as an aid in visualizing the different luminosity classes. Class V are main sequence stars, IV are subgiants, III are normal giants, II are bright giants, and the various I types are various categories of supergiants.

Appendix B: Sky map of stars

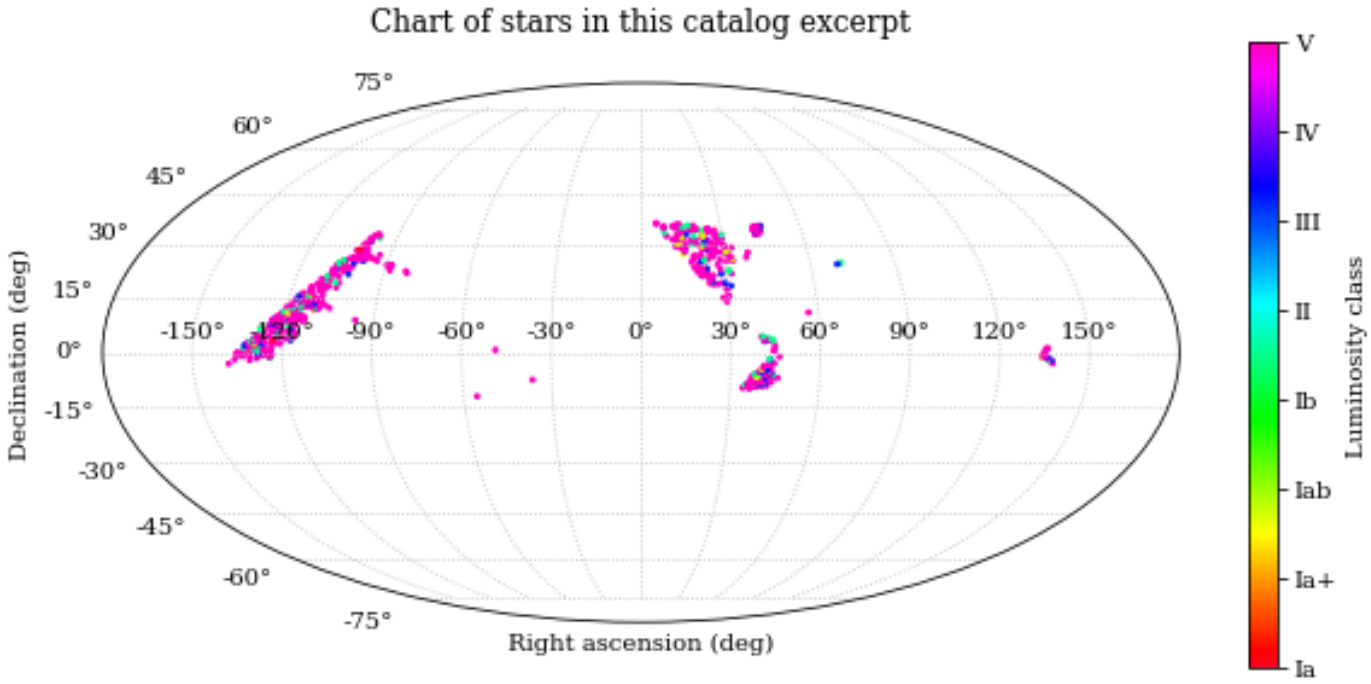

Fig. B.1. This chart displays the locations in the sky of each of the 630 stars with good and accurate models using their right ascension and declination coordinates provided in the SDSS catalog. The dominant color on the map is bright pink, which corresponds to main sequence stars (class V). This makes sense, as stars on the main sequence are the most common. Clearly, this excerpt of the catalog mostly contained stars in two or three main sections of the sky.