Proposed data model for a general archetype redshift template file format

Adam S. Bolton

August 5, 2014

1 Introduction

This note describes a proposed file format for storing general "archetype" redshift template grids for ingestion into the `redmonster` classification, redshift, and parameter measurement Python code.

The proposed standard is motivated by the desire to provide a general form of template file that can be written by end users and ingested into the redshift software to allow highly configurable spectroscopic template classes without any re-coding of the low-level analysis routines.

"Archetypes" refers to sets of single spectral template vectors that are not fit in linear combination with any other templates (except possibly low-order nuisance vectors).

This file standard is oriented towards the familiar units and conventions of optical spectroscopic redshift measurement, which may not be convenient for other experimental regimes.

2 File Format

2.1 File naming convention

Files conforming to this data model will follow the naming scheme

```
ndArch-CLASS-VERSION.fits
```

The string "`ndArch`" signifies "N-dimensional archetype", reflecting the accommodation of multiple physical parameter dimensions in the file standard.

`CLASS` and `VERSION` are arbitrary strings that may contain any legal filename characters other than the dash (`-`). Both the `CLASS` and `VERSION` strings are required to be present in some non-trivial form. The `CLASS` string will be extracted and used by the `redmonster` code as the primary qualitative identifier of the class of models contained in the file, while the `VERSION` string is intended to distinguish between multiple versions of the same type of model file. Users may adopt further specific conventions for `CLASS` and `VERSION` as appropriate to their needs.

The exact extension string "`.fits`" is formally required at present, but may in the future be relaxed to permit alternate and/or compressed versions.

2.2 File type

Valid `ndArch` files will be uncompressed FITS files with all relevant information contained in the filename and the primary HDU.

Additional FITS HDU extensions may be present, but are not defined by this data model.

2.3 Data structure and requirements

The data contained in the `ndArch` file will consist of a single multi-dimensional array containing flux densities or luminosity densities in units of F_λ (power per unit area per unit wavelength) or L_λ (power per unit wavelength). The absolute normalization may be physically meaningful, but is not required to

be meaningful. If the normalization is meaningful, the units shall be specified via the `BUNIT` keyword (see below), and must be the same for all data values in the file.

The first axis of the data array should correspond to **vacuum** wavelength (see below for the air-wavelength alternative case), and shall be gridded in positive increments of constant log-wavelength. There may be zero or more axes in addition to the first axis, up to the maximum number allowed by the FITS standard. Each axis beyond the first will generally correspond to a monotonically ordered physical model-parameter dimension (age, metallicity, emission-line strength, etc.), but may also correspond to an arbitrary labeled or unlabeled collection. Conventions for specifying the parameter dimensions are described in the following section.

The archetype template vectors should generally be assumed to have a uniform resolution characterized by a Gaussian line-spread function with a dispersion parameter σ equal to one sampling pixel, and should therefore be prepared as such to the extent possible.

2.4 Header structure and requirements

The following primary header keywords are **required** to be present and defined as specified, in addition to header keywords required by the FITS standard itself:

`CRPIX1`: Shall be set to the value 1, referencing the first sample point ("pixel") along the wavelength axis (one-based indexing).

`CRVAL1`: Shall specify the **base-10 logarithm** of the central wavelength **in vacuum Angstroms** of first pixel along the wavelength axis.

`CDELT1`: Shall specify the pixel-to-pixel increment in **base-10 logarithm** of vacuum wavelength from one pixel to the next along the wavelength axis.

The following primary header keyword is **required** in the case that the physical normalization of the templates is meaningful:

`BUNIT`: String giving the units of the template spectra.

The following primary header keyword is **required** in the case that the templates are given with respect to air wavelengths rather than vacuum:

`AIRORVAC`: String that is either `'air'` or `'vac'` depending upon wavelength convention. (The only significant value is `'air'`, since anything else, including the absence of this keyword, will be interpreted as `'vac'`.)

The following primary header keywords are supported as optional but **recommended** in some combination as appropriate:

`CNAMEn`: For $n > 1$, a string giving the name of the physical parameter coordinate along the n^{th} axis.

`CUNITn`: For $n > 1$, a string giving the units of the physical parameter coordinate along the n^{th} axis.

`CRPIXn, CRVALn, CDELTn`: For $n > 1$, specifying the physical parameter baselines for axes corresponding to regularly gridded numerical physical parameters.

`PVn_j`: For $n > 1$ and for the case of irregularly gridded numerical physical parameters along axis n , specifies the parameter value of the j^{th} point along the n^{th} axis. The index j shall begin with 1 and increase in integer steps up to the size of the n^{th} axis, with one keyword per grid step. Note that the FITS standard currently limits the maximum size of axes that can be represented in this manner to 99, and that leading zero-padding is not allowed.

`PSn_j`: For $n > 1$ and for the case of non-numerical physical parameters along axis n , specifies the parameter string value of the j^{th} point along the n^{th} axis. The index j shall begin with 1 and increase in integer steps up to the size of the n^{th} axis, with one keyword per grid step. Note that the FITS standard currently limits the maximum size of axes that can be represented in this manner to 99, and that leading zero-padding is not allowed.

`Nn_j`: For $n > 1$ and for the case of arbitrary labels along axis n, specifies the label string for the j^{th} point along the n^{th} axis. The index j shall begin with 1 and increase in integer steps up to the size of the n^{th} axis, with one keyword per grid step. These keywords do not belong to the FITS standard, and can accommodate dimensionality up to 999.

For any given axis beyond the first (wavelength) axis, the `redmonster` code

will use information from the above keywords to construct parameter-grid baselines. The precedence for establishing the baseline for each axis is first for `CRPIXn, CRVALn, CDELTn`, then for `PVn_j`, then for `PSn_j`, then for `Nn_j`, then for a one-based integer baseline in the absence of a valid set of keywords of any of the preceding specified types.

3 Implementation

Reader and writer routines for `ndArch` files conforming to this proposed standard are implemented in the `redmonster.datamgr.io` module as `read_ndArch` and `write_ndArch`. (The location of these routines is subject to change with future package reorganizations.) Detailed documentation of these file-handling routines can be found in their embedded docstrings.

A script called `test_ndArch.py` is also provided, which generates a file named `ndArch-TEST-v00.fits`, which in turn provides a unit test of the functioning of the `read_ndArch` routine.