f311 Documentation

Release 23.07.27

Julio Trevisan

Jul 27, 2023

Contents

Welcome!

Most of the programming work in science is arguably related to loading, saving, visualizing, and manipulating data files somehow. For some, it would be nice a common interface to handle these several data file types.

Project F311 provides a common ground to create new Python packages (herein called *collaborator packages*) containing new file handling and visualization routines, allowing these file types to be accessed through the `explorer.py` (Figure 2.1) application in a plugin-like fashion.

In addition, F311 provide a `programs.py` scripts that work as an index to all scripts included in all collaborator packages.

By the end of 2017, the following projects were listed as *collaborators*:

- ariastro (https://github.com/aricorte/ariastro)
- PyFANT (source: https://github.com/trevisanj/pyfant; docs: https://trevisanj.github.io/pyfant)
- AOSSS (source: https://github.com/trevisanj/aosss; docs: https://trevisanj.github.io/aosss)
- F311 itself (source: https://github.com/trevisanj/F311; docs: https://trevisanj.github.io/f311)

Project F311 is a collaborator to itself, providing classes to handle 1D spectral files (such as FITS, "x-y"), and a few scripts as listed below.

CHAPTER 1

Importing the API

The F311 API (application programming interface) can be imported like this:

```
import f311
```

CHAPTER 2

Applications (scripts)

The following applications are included in package `f311`:

- *explorer.py*: F311 Explorer – file manager-like GUI (graphical user interface) (Figure 2.1)
- *programs.py*: Lists all programs available across all *collaborators packages*
- *cut-spectrum.py*: Cuts spectrum file to wavelength interval specified
- *plot-spectra.py*: Plots spectra on screen or creates PDF file

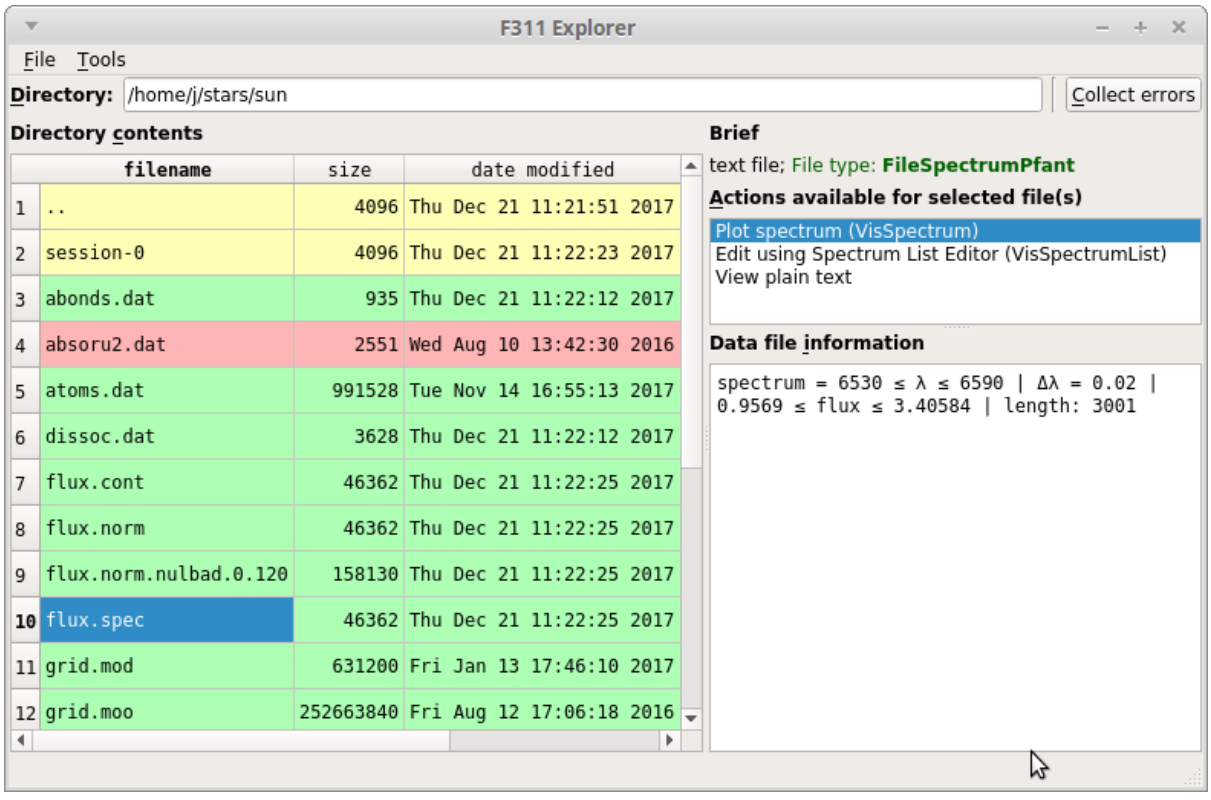

Figure 2.1: – screenshot of `explorer.py`.

CHAPTER 3

Acknowledgement

The project started in 2015 at IAG-USP (Institute of Astronomy, Geophysics and Atmospheric Sciences at University of São Paulo, Brazil).

Partially funded by FAPESP - Research Support Foundation of the State of São Paulo, Brazil (2015-2017).

CHAPTER 4

Contents

4.1 F311 Installation

Note: As of July 2023, we recommend using Anaconda/Miniconda and Python 3.11

4.1.1 Setting up Anaconda virtual environment

A virtual environment is an isolated Python installation, which can be activated/deactivated at will, and does not interfere in the rest of your system.

First you will need to have Anaconda or Miniconda installed. If you are in doubt about this, just install **Miniconda**.

Once Anaconda/Miniconda is installed, create a new virtual environment called **astroenv** (or any name you like):

```
conda create --name astroenv python=3.11
```

Activate this new virtual environment:

```
source activate astroenv
```

Now, install PyQt5 manually using the following:

```
conda install pyqt
```

Note: (about PyQt5) The correct installation of PyQt5 can be challenging depending on one's system. The above solution was found to work in July 2023. It is important to use `conda` (**not** `pip`) for the installation of PyQt5.

Now install package `f311`:

```
pip install f311
```

To activate the environment:

```
source activate astroenv
```

To deactivate the environment:

```
source deactivate
```

4.1.2 Installing F311 in Developer Mode

Note: Install F311 in developer mode only if you want to make changes to the source code.

First, set up Anaconda/Miniconda and install PyQt5 as described above.

Then, clone the F311 GitHub repository:

```
git clone ssh://git@github.com/trevisanj/f311.git
```

or

```
git clone http://github.com/trevisanj/f311
```

Finally, install F311 in **developer mode**:

```
cd f311
python setup.py develop
```

4.1.3 Upgrading package `f311`

If package `f311` is already installed, but you need to install a new version, please follow these instructions.

Upgrading in the majority of cases

Package `f311` can be upgraded to a new version by typing:

```
pip install f311 --upgrade
```

Upgrading F311 in developer mode

Enter the *f311* repository (directory) cloned from Github, then type:

```
git pull
python setup.py develop
```

4.1.4 Package dependencies

`f311` package dependencies are listed below. This list is provided only for reference, as **most of these packages should be automatically installed** during the installation process (except for `pyqt`).

- numpy
- scipy
- matplotlib

- astropy
- configobj
- bs4
- lxml
- robobrowser
- requests
- tabulate
- rows
- pyqt
- a99

4.1.5 Troubleshooting installation

This section shows some possible errors and their solutions.

PtQt5

If you have PyQt5-related errors such as the following:

```
ImportError: [...]/QtGui.abi3.so: undefined symbol: _ZdaPvm, version Qt_5
```

be sure to follow the above instructions to install PyQt5.

MatPlotLib and PyQt5

```
ValueError: Unrecognized backend string "qt5agg": valid strings are ['GTKAgg', 'template', 'pdf',
'GTK3Agg', 'cairo', 'TkAgg', 'pgf', 'MacOSX', 'GTK', 'WX', 'GTKCairo', 'Qt4Agg', 'svg', 'agg',
'ps', 'emf', 'WebAgg', 'gdk', 'WXAgg', 'CocoaAgg', 'GTK3Cairo']
```

Solution: upgrade Matplotlib to version 1.4 or later

Problems with package bs4

```
bs4.FeatureNotFound: Couldn't find a tree builder with the features you requested: lxml. Do you␣
↪need to install a parser library?
```

Solution: install package "lxml":

```
pip install lxml
```

Other problems

If you have other installation problems or have a suggestion, please open an issue at the project site on GitHub.

4.2 Index of applications (scripts)

4.2.1 Script `cut-spectrum.py`

```
usage: cut-spectrum.py [-h] llzero llfin fn_input fn_output

Cuts spectrum file to wavelength interval specified

Resulting spectrum Saved in 2-column ASCII format

The interval is [llzero, llfin]

positional arguments:
  llzero      lower wavelength boundary (angstrom)
  llfin       upper wavelength boundary (angstrom)
  fn_input    input file name
  fn_output   output file name

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311*

4.2.2 Script `plot-spectra.py`

```
usage: plot-spectra.py [-h] [--ovl | --pieces | --pages] [--aint [AINT]]
                       [--fn_output [FN_OUTPUT]] [--ymin [YMIN]]
                       [-r [NUM_ROWS]]
                       fn [fn ...]

Plots spectra on screen or creates PDF file

It can work in four different modes:

a) grid of sub-plots, one for each spectrum (default mode)
   Example:
   plot-spectra.py flux.norm.nulbad measured.fits

b) single plot with all spectra overlapped ("--ovl" option)
   Example:
   > plot-spectra.py --ovl flux.norm.nulbad measured.fits

c) PDF file with a small wavelength interval per page ("--pieces" option).
   This is useful to flick through a large wavelength range.
   Example:
   > plot-spectra.py --pieces --aint 7 flux.norm.nulbad measured.fits

d) PDF file with one spectrum per page ("--pages" option).
   Example:
   > plot-spectra.py --pages flux.*

Types of files supported:

  - pfant output, e.g., flux.norm;
  - nulbad output, e.g., flux.norm.nulbad;
  - 2-column "lambda-flux" generic text files;
  - FITS files.

positional arguments:
```

(continues on next page)

(continued from previous page)

```
  fn                    name of spectrum file(s) (many types supported)
                        (wildcards allowed, e.g., "flux.*")

optional arguments:
  -h, --help            show this help message and exit
  --ovl                 Overlapped graphics (default: False)
  --pieces              If set, will generate a PDF file with each page
                        containing one "piece" of the spectra of lengthgiven
                        by the --aint option. (default: False)
  --pages               If set, will generate a PDF file with one spectrum per
                        page (default: False)
  --aint [AINT]         length of each piece-plot in wavelength units (used
                        only if --pieces) (default: 10)
  --fn_output [FN_OUTPUT]
                        PDF output file name (used only if --pieces) (default:
                        (plot-spectra-<xxxx>.pdf))
  --ymin [YMIN]         Minimum value for y-axis (default: (automatic))
  -r [NUM_ROWS], --num_rows [NUM_ROWS]
                        Number of rows in subplot grid (default: (automatic))
```

This script belongs to package *f311*

4.2.3 Script `programs.py`

```
usage: programs.py [-h] [-p PKGNAME] [-l] [-k] [-_]
                   [{text,markdown-list,markdown-table,rest-list,rest-toctree}]

Lists all programs available

positional arguments:
  {text,markdown-list,markdown-table,rest-list,rest-toctree}
                        Print format (default: text)

optional arguments:
  -h, --help            show this help message and exit
  -p PKGNAME, --pkgname PKGNAME
                        List programs from this package only (default: (all))
  -l, --list-packages   Lists all packages (default: False)
  -k, --rest-links      If format=="rest-list", renders program names as links
                        to their respective documentation pages (default:
                        False)
  -_, --protected       Includes protected scripts (starting with '_')_
                        (default: False)
```

This script belongs to package *f311*

4.2.4 Script `explorer.py`

```
usage: explorer.py [-h] [dir]

f311 Explorer --  file manager-like application to list, visualize, and edit data files

positional arguments:
  dir         directory name (default: .)

optional arguments:
  -h, --help  show this help message and exit
```

This script belongs to package *f311*

See screenshot (Figure 2.1).

4.3 Collaboration

4.3.1 The *collaboration model*

Package `f311` provides a plugin-like model (*collaboration model*) that allows 3rd-party *collaborator packages* to contribute with:

- `f311.DataFile` subclasses
- `f311.Vis` subclasses
- Standalone scripts

4.3.2 Implications

- New file types are recognized in the F311 API, *e.g.* using methods `f311.load_any_file()`, `f311.load_spectrum()`, `f311.tabulate_filetypes_rest()`
- New file types and their visualizations are recognized in `explorer.py`
- Scripts will be indexed by `programs.py`

4.3.3 Creating a collaborator project

1. Start your new project. A template project skeleton is available with the source code as a directory named `template-project`
2. Create new resources as listed in the beginning of this section;
3. In order to make package *f311* "see" the new project, create a pull request for project f311 on GitHub (https://github.com/trevisanj/f311), and append your package name to the *f311.collaboration.EXTERNAL_COLLABORATORS* list.

4.4 Examples

4.4.1 List file types handled by all collaborator packages

The current list considers projects aosss, ariastro, f311, and PyFANT.

```
"""Lists different subsets of DataFile subclasses"""
import f311

titles = ("text", "binary", "1D spectrum")
allclasses = (f311.classes_txt(), f311.classes_bin(), f311.classes_sp())

for title, classes in zip(titles, allclasses):
    print("\n*** Classes that can handle {} files***".format(title))
    for cls in classes:
        print("{:25}: {}".format(cls.__name__, cls.__doc__.strip().split("\n")[0]))
```

```
*** Classes that can handle text files***
FileAbXFwhm              : `x.py` Differential Abundances X FWHMs (Python source)
FileAbonds               : PFANT Stellar Chemical Abundances
FileAbsoru2              : PFANT "Absoru2" file
FileAtoms                : PFANT Atomic Lines
FileConfigConvMol        : Configuration file for molecular lines conversion GUI (Python code)
FileDissoc               : PFANT Stellar Dissociation Equilibrium Information
FileHmap                 : PFANT Hydrogen Lines Map
FileKuruczMolecule       : Kurucz molecular lines file
FileKuruczMolecule1      : Kurucz molecular lines file following format of file "c2dabrookek.asc"
FileKuruczMoleculeBase   : Base class for the two types of Kurucz molecular lines file
FileKuruczMoleculeOld    : Kurucz molecular lines file, old format #0
FileKuruczMoleculeOld1   : Kurucz molecular lines file, old format #1
FileMain                 : PFANT Main Stellar Configuration
FileModTxt               : MARCS Atmospheric Model (text file)
FileMolConsts            : Molecular constants config file (Python code)
FileMolecules            : PFANT Molecular Lines
FileOpa                  : MARCS ".opa" (opacity model) file format.
FileOptions              : PFANT Command-line Options
FilePar                  : WebSim-COMPASS ".par" (parameters) file
FilePartit               : PFANT Partition Function
FilePlezTiO              : Plez molecular lines file, TiO format
FilePy                   : Configuration file saved as a .py Python source script
FilePyConfig             : Base class for config files. Inherit and set class variable
↪'modulevarname' besides usual
FileSpectrum             : Base class for all files representing a single 1D spectrum
FileSpectrumNulbad       : PFANT Spectrum (`nulbad` output)
FileSpectrumPfant        : PFANT Spectrum (`pfant` output)
FileSpectrumXY           : "Lambda-flux" Spectrum (2-column text file)
FileTRAPRBInput          : Input file for the TRAPRB Fortran code (which calculates Franck-Condon␣
↪factors)
FileTRAPRBOutput         : Output file for the TRAPRB Fortran code (which calculates Franck-Condon␣
↪factors)
FileToH                  : PFANT Hydrogen Line Profile
FileVald3                : VALD3 atomic or molecular lines file

*** Classes that can handle binary files***
FileFullCube             : FITS WebSim Compass Data Cube
FileHitranDB             : HITRAN Molecules Catalogue
FileModBin               : PFANT Atmospheric Model (binary file)
FileMolDB                : Database of Molecular Constants
FileMoo                  : Atmospheric model or grid of models (with opacities included)
FileSQLiteDB             : Represents a SQLite database file.
FileSparseCube           : FITS Sparse Data Cube (storage to take less disk space)
FileSpectrumFits         : FITS Spectrum
FileSpectrumList         : FITS Spectrum List
FileGalfit               : FITS file with frames named INPUT_*, MODEL_*, RESIDUAL_* (Galfit␣
↪software output)

*** Classes that can handle 1D spectrum files***
FileSpectrum             : Base class for all files representing a single 1D spectrum
FileSpectrumFits         : FITS Spectrum
FileSpectrumNulbad       : PFANT Spectrum (`nulbad` output)
FileSpectrumPfant        : PFANT Spectrum (`pfant` output)
FileSpectrumXY           : "Lambda-flux" Spectrum (2-column text file)


By the way, the table above was generated with the following code:
```

4.4.2 Another way to list the file types (as ReST table)

```
import f311.filetypes as ft
print("\n".join(ft.tabulate_filetypes_rest(description_width=55)))
```

Description	Default filename	Class r
“Lambda-flux” Spectrum (2-column text file)		FileSp
Atmospheric model or grid of models (with opacities included)	grid.moo	FileMo
Configuration file for molecular lines conversion GUI (Python code)	configconvmol.py	FileCo
Database of Molecular Constants	moldb.sqlite	FileMo
FITS Sparse Data Cube (storage to take less disk space)	default.sparsecube	FileSp
FITS Spectrum		FileSp
FITS Spectrum List	default.splist	FileSp
FITS WebSim Compass Data Cube	default.fullcube	FileFu
FITS file with frames named INPUT_*, MODEL_*, RESIDUAL_* (Galfit software output)		FileGa
File containing Franck-Condon Factors (FCFs)		FileFC
HITRAN Molecules Catalogue	hitrandb.sqlite	FileHit
Kurucz molecular lines file		FileKu
Kurucz molecular lines file, old format #0		FileKu
Kurucz molecular lines file, old format #1		FileKu
MARCS “.opa” (opacity model) file format.	modeles.opa	FileOp
MARCS Atmospheric Model (text file)		FileMo
Molecular constants config file (Python code)	configmolconsts.py	FileMo
PFANT “Absoru2” file	absoru2.dat	FileAb
PFANT Atmospheric Model (binary file)	modeles.mod	FileMo
PFANT Atomic Lines	atoms.dat	FileAto
PFANT Command-line Options	options.py	FileOp
PFANT Hydrogen Line Profile	thalpha	FileTol
PFANT Hygrogen Lines Map	hmap.dat	FileHm
PFANT Main Stellar Configuration	main.dat	FileMa
PFANT Molecular Lines	molecules.dat	FileMo
PFANT Partition Function	partit.dat	FilePar
PFANT Spectrum (*nulbad* output)		FileSp
PFANT Spectrum (*pfant* output)	flux.norm	FileSp
PFANT Stellar Chemical Abundances	abonds.dat	FileAb
PFANT Stellar Dissociation Equilibrium Information	dissoc.dat	FileDis
Plez molecular lines file, TiO format		FilePle
VALD3 atomic or molecular lines file		FileVal
WebSim-COMPASS “.par” (parameters) file		FilePar
x.py Differential Abundances X FWHMs (Python source)	abxfwhm.py	FileAb

4.4.3 Convert 1D spectral file to FITS format

```
#!/usr/bin/env python
"""Converts 1D spectral file of any supported type to FITS format.

The new file is saved with name "<original-filename>.fits".

TODO handle non-equally spaced wavelength values
"""

import f311
import sys
import logging
```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":
    if len(sys.argv) < 2 or any([x.startswith("-") for x in sys.argv[1:]]):
        print(__doc__+"\nUsage:\n\n    convert-to-fits.py filename0 [filename1 [filename2 [...]]]\n
↪")
        sys.exit()

    for filename in sys.argv[1:]:
        print("Converting file '{}'...".format(filename))

        try:
            spectrum = f311.load_spectrum(filename)

            if spectrum is None:
                print("File '{}' not recognized as a 1D spectral file".format(filename))
                continue

            filename_new = filename+".fits"

            fnew = f311.FileSpectrumFits()
            fnew.spectrum = spectrum
            fnew.save_as(filename_new)

            print("Successfully saved '{}'".format(filename_new))
        except:
            logging.exception("Error converting file '{}'".format(filename))
```

Todo: More examples

4.5 f311

4.5.1 f311 package

Subpackages

f311.explorer package

Subpackages

f311.explorer.gui package

Submodules

f311.explorer.gui.a_XExplorer module

f311.explorer.gui.a_XFileMainWindow module

f311.explorer.gui.a_XSelectDataFile module

f311.explorer.gui.a_XText module

Module contents