Goodman Focus Documentation

Release 0.3.5

Simon Torres

Jan 07, 2020

HOW TO INSTALL

CHAPTER

ONE

OVERVIEW

This is a tool to obtain the best focus from a series of images obtained at different focus values. It works for imaging and for spectroscopy.

This tool requires python *3.6* at least to work. It will not install with *3.5*.

We recommend using astroconda since it is easier.

Note: We recommend using a virtual environment management system such as astroconda

1.1 Install Using PYPI

Create a virtual environment using *conda* and specify python version *3.6*.

```
conda create -n goodman_focus python=3.6
```

Install using *pip*

```
pip install goodman-focus
```

1.2 Install Using GitHub

Clone the latest version using:

```
git clone https://github.com/soar-telescope/goodman_focus.git
```

Move into the new directory

```
cd goodman_focus
```

Create a virtual environment using the *environment.yml* file and activate it.

```
conda env create python=3.6 -f environment.yml
```

This will create a virtual environment named `goodman_focus` with all the important parts on it.

```
conda activate goodman_focus
```

Install using *pip*

```
pip install .
```

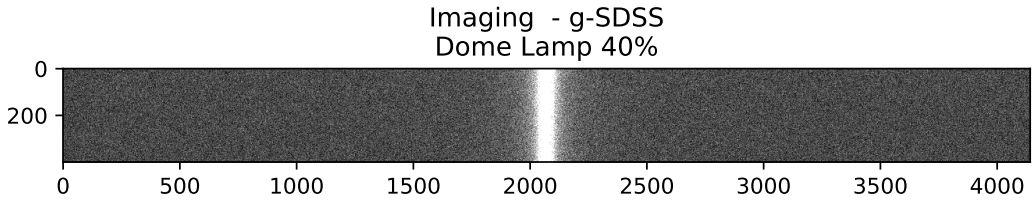
1.3 Getting Data

The goodman focus value ranges from -2000 to $+2000$ the best practice (from experience) is to sample every 200. If you want to refine you should not go below a 100 step. If you already know the approximate focus value for a given setup, you don't need to do the full range, just take enough samples so it is possible to make a fit or see the trends. At least five points.

Note: Always use the narrowest slit available

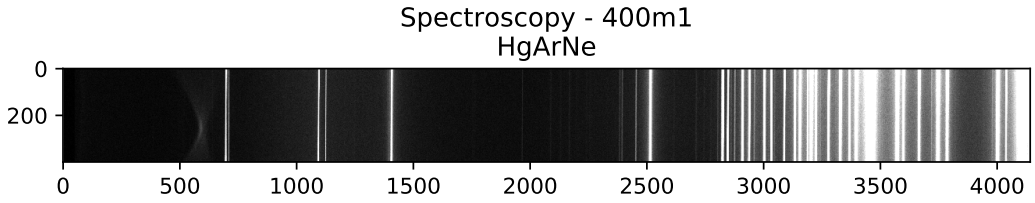
1.3.1 Imaging

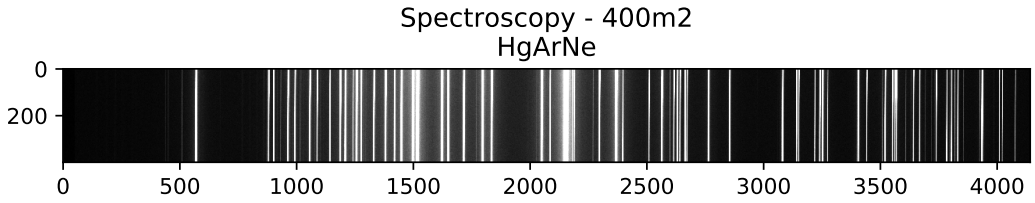
For imaging, an image of the slit is obtained, illuminated by an uniform light such as a quartz lamp. A special ROI is used for faster reading. This does not affect the focus of course.



1.3.2 Spectroscopy

In spectroscopy a comparison lamp is observed and after every line is measured and fitted, an average is obtained with sigma clip rejection.





1.4 Running from terminal

There is an automatic script that will obtain focus from a folder containing a focus sequence.

If you have *fits* files you can simply run.

```
goodman-focus
```

It will run with the following defaults:

Table 1: Default values for arguments

Argument	Default Value	Options
`--data-path <input>`	Current Working Directory	Any valid path
`--file-pattern <input>`	*.fits	Any
`--features-model <input>`	gaussian	moffat
`--plot-results`	False	True
`--debug`	False	True

Where `<input>` is what you type.

To get some help and a full list of options use:

```
goodman-focus -h
```

1.5 Using it as a library

After installing *Install Using PYPI* you can also import the class and instantiate it providing a set of arguments and values or using default ones.

```
from goodman_focus.goodman_focus import GoodmanFocus
```

If no argument is provided it will instantiate with the default values.

The list of arguments can be defined as follow:

```
import os
from goodman_focus.goodman_focus import GoodmanFocus

goodman_focus = GoodmanFocus(data_path=os.getcwd(),
                             file_pattern='*.fits',
                             obstype='FOCUS',
                             features_model='gaussian',
```

(continues on next page)

(continued from previous page)

```
                                plot_results=False,
                                debug=False)
```

Which is equivalent to:

```
from goodman_focus.goodman_focus import GoodmanFocus

goodman_focus = GoodmanFocus()
```

`features_model` is the function or model to fit to each detected line. `gaussian` will use a `Gaussian1D` which provide more consistent results. and `moffat` will use a `Moffat1D` model which fits the profile better but is harder to control and results are less consistent than when using a gaussian.

Finally you need to call the instance, here is a full example.

```
from goodman_focus.goodman_focus import GoodmanFocus

goodman_focus = GoodmanFocus()

results = goodman_focus()
```

However since version *0.3.0* you can pass a list of files and all will only check that all files exists

1.6 Interpreting Results

The terminal version will print a message like this

```
[17:16:06][INFO]: Best Focus for mode SP_Red_400m2_GG455 is -1032.
6413206603302
```

Using it as a library will return a dictionary with the following values. Combination of settings for which the code is the same is called a *mode*, so the keys of the dictionary are the *mode name*, how the name is constructed is explained in decoding-mode-name

```
{'IM_Red_g-SDSS': -571.4837418709354,
 'IM_Red_i-SDSS': -802.567783891946,
 'IM_Red_r-SDSS': -573.8694347173587,
 'IM_Red_z-SDSS': -1161.5072536268135,
 'SP_Red_400m1_NOFILTER': -492.0760380190095,
 'SP_Red_400m2_GG455': -1032.6413206603302}
```

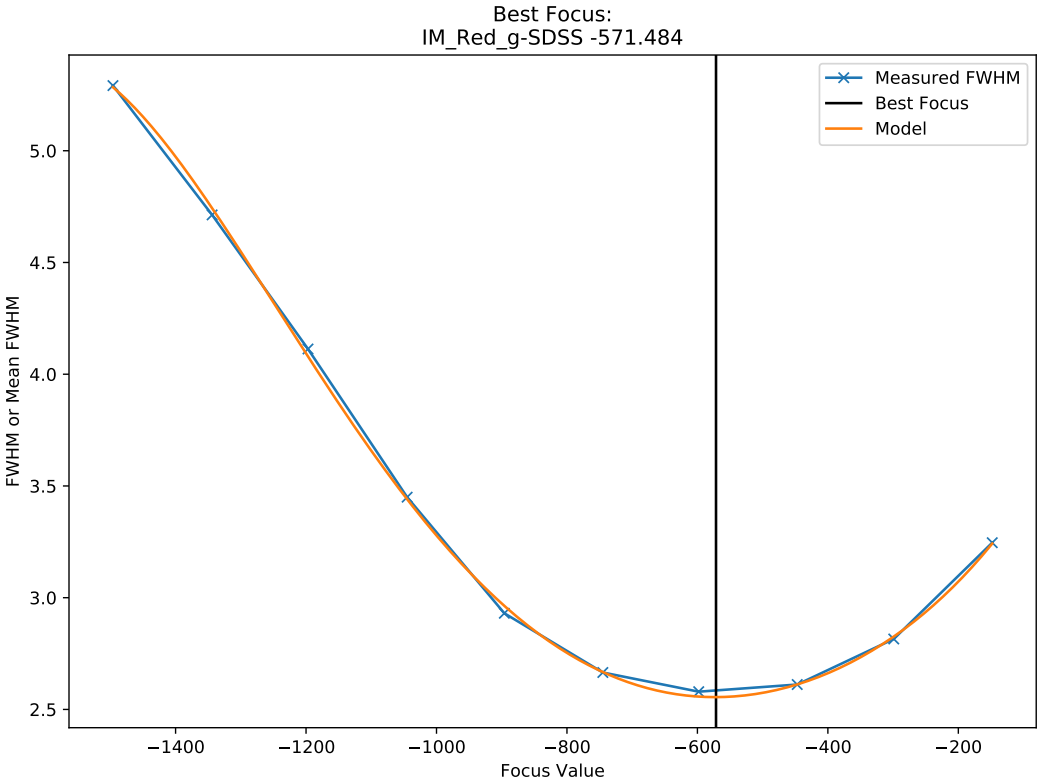
It is also possible to obtain a plot, from terminal, use `--plot-results`. Below is a repreduction of results obtained with test data.

1.6.1 Decoding de mode name

The mode name is constructed using two letters to define the observing technique (Imaging or Spectroscopy) and values obtained from the header. The characters <, > and *blanks* are removed.

The mode name is different for Imaging and Spectroscopy, since for imaging the important settings are the instrument and the filter and for spectroscopy the important values come from the instrument, the grating and observing mode and filter from second filter wheel. Below, the word inside the parenthesis represents a kewyword from the header.

For imaging:


Best Focus:
IM_Red_g-SDSS -571.484
Measured FWHM
Best Focus
Model
FWHM or Mean FWHM
5.0
4.5
4.0
3.5
3.0
2.5
−1400
−1200
−1000
−800
−600
−400
−200
Focus Value

```
IM_(INSTCONF)_(FILTER)
```

for example:

```
IM_Red_g-SDSS
```

For spectroscopy:

```
SP_(INSTCONF)_(WAVMODE)_(FILTER2)
```

for example:

```
SP_Red_400m2_GG455
```

1.7 Change History

1.7.1 0.3.4

- Fixed version of *ccdproc* to *1.3.0.post1*. *ccdproc==2.0.0* does have some problems reported on *astropy/ccdproc#699*

1.7.2 0.3.3

- Changed Sigma Clipping iterations from 1 to 3
- Added sigma clip iterations as argument to function *get_fwhm* though this is not exposed to the user.

1.7.3 0.3.2

- Changed logger setup
- Moved data directory validation from instantiation to execution.

1.7.4 0.3.1

- Fixed bug on the calculation of the pseudo-derivate used to find best focus value
- Updated hardcoded string that defines the Imaging wavmode from *Imaging* to the new *IMAGING*.
- Added docstrings

1.7.5 0.3.0

- Created dedicated documentation for readthedocs.
- Fixed bug where return was missing,
- GoodmanFocus need to be instantiated only once [#19]
- Calling instance of GoodmanFocus can receive a list of files as input [#19]
- Argument –file-pattern is now actually used in file selection [#18]
- Eliminated some warnings.
- Included plots in documentation.

1.7.6 0.2.0

- Added messages when no file matches the *–obstype* value, by default is *FOCUS* [#9]
- Replaced *parser.error* by *log.error* and *sys.exit* when the directory does not exist and when exists but is empty.
- Added test for cases when the directory does not exist, when is empty and when no file matches the selection on *–obstype* which by default is *FOCUS*.
- Replaced *logging.config.dictConfig* by *logging.basicConfig* which fixed several issues. For instance *–debug* was unusable, and also there were duplicated log entries for the file handler when used as a library in other application. [#10]
- Replaced the use of the function *get_args* by using arguments on class instantiation instead
- Created name for modes [#11]

1.7.7 0.1.3

- Fixed some issues with documentation
- Added .readthedocs.yml file for RTD builds
- Added `install_requires` field in `setup()`
- Removed python 3.5 from the supported versions.

1.8 License

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search