Goodman HTS Pipeline User Manual

version 0.10

Simón Torres, César Briceño and Bruno Quint

April 23, 2018

Contents

Introduction

This is the User Manual for the *Goodman Spectroscopic Data Reduction Pipeline*. It provides an overview of the pipeline's main features, instructions on its use and how to run it on our dedicated *SOAR Data Reduction Server*, and installation instructions for those who wish to run it on their own computers.

Overview

The Goodman Spectroscopic Data Reduction Pipeline - GOODSPEC - is a Python-based package for producing science-ready, wavelength-calibrated, 1-D spectra. The goal of **goodspec** is to provide SOAR users with an easy to use, very well documented software for reducing spectra obtained with the Goodman spectrograph. Though the current implementation assumes offline data reduction, our aim is to provide the capability to run it in real time, so 1-D wavelength calibrated spectra can be produced shortly after the shutter closes.

The pipeline is primarily intended to be run on a data reduction dedicated computer. Instructions for running the software are provided in the Running Pipeline section of this guide. The Goodman Spectroscopic Data Reduction Pipeline project is hosted at GitHub at it's GitHub Repository.

Currently the pipeline is separated into two main components. The initial processing is done by `redccd`, which trims the images, and carries out bias and flat corrections. The spectroscopic processing is done by `redspec` and carries out the following steps:

- Identifies multiple targets (spectra of more than one object in the slit)
- Trace the spectra
- Extract the spectra
- Estimate and subtract background
- Saves extracted (1D) spectrum, without wavelength calibration.
- Find the wavelength solution. Defaults to automatic wavelength solution, but can be done interactively
- Linearize data (resample)
- Write wavelength solution to FITS header
- Create a new file for the wavelength calibrated 1D spectrum

Features

- Self-contained, full data reduction package for the most commonly used predefined setups with Goodman HTS. Given the almost limitless number of possible configurations available with the Goodman instrument, only the most popular configurations will be supported, though we will try to add as many modes as possible.
- Python based, using existing Astropy libraries as much as feasible.
- Extensively documented, using general coding standards: PEP8 – Style Guide, PEP257 – Docstrings Convention (in-code documentation) – Google Style
- Multiplataform compatibility (tested on Linux Ubuntu, CentOS and MacOSX).
- Modular design. Could be used as a library within other Python applications.

Ways to run the pipeline

There are two ways to use the pipeline.

1. **Run it directly on a SOAR data reduction server** that you can access using VNC.
2. **Download and install the pipeline** (go to the Installing Pipeline section of this manual). Though we will try our best to provide answers to quick and simple installation issues, we cannot provide general installation support.

What the pipeline does not do

- In its current version the pipeline does not perform combination of individual spectra. If you obtained several individual exposures of the same object, they will be output as separate 1-D, wavelength-calibrated spectra.
- It does not combine lamps. If there are more than one comparison lamp associated with an spectrum they will be processed and saved separately.
- There is yet no flux calibration. We are working on a module that will do this. But this will be left for a later release.
- This pipeline does not evaluate nor select data by quality. It will simply try to run using all existing files. **Make sure you only have good data in the folder that will be reduced**.

General Considerations on using the pipeline

The Goodman Spectroscopic Pipeline is meant to work as a single package. However, the full process is split in two separate modules: `redccd` and `redspec`. The first does the basic 2D image reduction, applying bias, flat field corrections, and cosmic ray removal. The second module, `redspec`, takes the corrected 2D images output by `redccd` and produces wavelength-calibrated 1D spectra.

The pipeline is run from the command line in a terminal window. Each module is run separately, first `redccd` followed by `redspec`, however, you could run both sequentially from e.g. a shell script, just make sure you move to the the right directory.

In order to make things easier you should organize your data:

1. Make sure all the data in your folder corresponds to the same binning, readout mode, region of interest (ROI), and grating/wavelength mode combination.
2. You should have bias, flats (quartz or dome flats), and the appropriate comparison lamps. Other files like acquisition images, slit images and focus images should be deleted.
3. Do not mix dome flats with quartz lamp flats. As an example, suppose you took both quartz lamps and dome flats for your targets. You could create two folders, one with the science data and the dome flats, and another with the same science data and the quartz lamps. Then, if you run the pipeline in each folder you can compare the results and decide which type of flat works best for my particular case.

Command line arguments

For a list of the options and command line arguments type `--help` argument:

For `redccd`

```
usage: redccd [-h] [--auto-clean] [--cosmic <method>] [--combine]
              [--dcr-par-dir <dcr.par_directory>] [--debug]
              [--flat-normalize <normalization_method>]
              [--flat-norm-order <order>] [--ignore-bias] [--ignore-flats]
              [--keep-cosmic-files] [--raw-path <raw_path>]
              [--red-path <red_path>] [--saturation <value>]

Goodman CCD Reduction - CCD reductions for Goodman spectroscopic data.
Pipeline Version: 0.10.1rc

optional arguments:
  -h, --help            show this help message and exit
  --auto-clean          Automatically clean reduced data directory
  --cosmic <method>     Clean cosmic rays from all data. Options are: 'dcr',
                        'lacosmic' or 'none'. Default is 'dcr'. See manual for
                        full description of dcr.
  --combine             Combine compatible data (experimental)
  --dcr-par-dir <dcr.par_directory>
                        Directory of default dcr.par file
  --debug               Show detailed information of the process.
  --flat-normalize <normalization_method>
                        Choose a method to normalize the master flat for
                        spectroscopy. Choices are: mean, simple (model) and
                        full (fits model to each line).
  --flat-norm-order <order>
                        Defines the order of the model to be fitted. Default
                        to 15
  --ignore-bias         Ignore bias correction
  --ignore-flats        Ignore flat field correction
  --keep-cosmic-files   After cleaning cosmic rays with dcr, do not remove the
                        input file and the cosmic rays file.
  --raw-path <raw_path>
                        Path to raw data.
```

```
  --red-path <red_path>
                        Path to reduced data.
  --saturation <value>  Saturation limit. Default to 65.000 ADU (counts)
```

And for `redspec`

```
usage: redspec [-h] [--data-path <Source Path>]
               [--proc-path <Destination Path>]
               [--search-pattern <Search Pattern>]
               [--output-prefix <Out Prefix>] [--extraction <Extraction Type>]
               [--reference-files <Reference Dir>] [--debug]
               [--max-targets <max targets>] [--save-plots] [--plot-results]

Extracts goodman spectra and does automatic wavelength calibration. Pipeline
Version: 0.10.1rc

optional arguments:
  -h, --help            show this help message and exit
  --data-path <Source Path>
                        Path for location of raw data. Default <./>
  --proc-path <Destination Path>
                        Path for destination of processed data. Default <./>
  --search-pattern <Search Pattern>
                        Pattern for matching the goodman's reduced data.
  --output-prefix <Out Prefix>
                        Prefix to add to calibrated spectrum.
  --extraction <Extraction Type>
                        Only fractional pixel extraction is implemented.
  --reference-files <Reference Dir>
                        Directory of Reference files location
  --debug               Debugging Mode
  --max-targets <max targets>
                        Maximum number of targets to be found in a single
                        image. Default 3
  --save-plots          Save all plots in a directory
  --plot-results        Show wavelength calibrated spectrum at the end.
```

Lists of Reference Lamps Available

The automatic wavelength calibration relies on having previously calibrated reference lamps obtained in the same configuration or mode. It is also important that the lamp names are correct, for instance `HgAr` is quite different than `HgArNe`. For interactive wavelength calibration, reference lamps are used as a visual aid only. It lets you find the matching laboratory lines values that will be used to fit a pixel to wavelength relation that we call *Wavelength Solution* The list of lamps is the following.

Grating	Mode	Filter	Lamp
400	M1	None	HgAr
400	M1	None	HgArNe
400	M2	GG455	Ar
400	M2	GG455	Ne
400	M2	GG455	HgAr
400	M2	GG455	HgArNe
400	M2	GG455	CuHeAr
400	M2	GG455	FeHeAr

Important

More lamps will be made public shortly.

Adding new reference lamps

It is possible to add new lamps, you just need a raw lamp that meets the following specifications with respect to your science project:

- Same instrument configuration or mode
- Same Grating
- Same order blocking filter if present
- Same binning
- Same lamp/combination that you use in your observations
- Smallest slit possible. Equal is OK too.

See Creating Reference Lamps for instructions on how to proceed. Now you have two options, identify the system folder where the lamps that come with the package are saved and simply put it there or put it in another directory and use the argument `--reference-files`

```
redspec --reference-files /path/to/ref-lamp-location
```

If you require that your lamps are made available as a part of the package you can contact `storres [at] ctio [dot] noao [dot] edu` and we will handle your request.

Headers Requirements

Goodman HTS spectra have small non-linearities on their wavelength solutions. They are small but big enough that they MUST be taken into account.

It was necessary to implement a custom way of storing non-linear wavelength solutions that at the same time allowed for keeping data as *untouched* as possible. The main reason is that linearizing the reference lamps made harder to track down those non-linearities on the new data being calibrated and also; The documentation on how to write non-linear solution to a FITS header is not good, besides it appears that nobody is trying to improve it neither trying to implement it. Below I compile a list of required keywords for comparison lamps if they want to be used as

reference lamps. The full list of keywords is listed under Headers.

General Custom Keywords:

Every image processed with the *Goodman Spectroscopic Pipeline* will have the general custom keywords. The one required for a reference lamp is the following:

```
GSP_FNAM = file-name.fits // Current file name
```

Record of line centers in Pixel and Angstrom:

Every line detected in the reference lamp is recorded both in its pixel value and later (most likely entered by hand) in angstrom value. The root string is `GSP_P` followed by a zero-padded three digit sequential number (001, 002, etc). For instance.

```
GSP_P001= 499.5377036976768  / Line location in pixel value
GSP_P002= 810.5548319623747  / Line location in pixel value
GSP_P003= 831.6984711087946  / Line location in pixel value
```

The equivalent values in angstrom are then recorded with the root string `GSP_A` and the same numerical pattern as before.

```
GSP_A001= 5460.75            / Line location in angstrom value
GSP_A002= 5769.61            / Line location in angstrom value
GSP_A003= 5790.67            / Line location in angstrom value
```

`GSP_P001` and `GSP_A001` are a match. If any of the angstrom value entries have a value of `0` (default value) the equivalent pixel entry is ignored.

Important

Those keywords are used to calculate the mathematical fit of the wavelength solution and are not used on normal operation. Our philosophy here is that the line identification has to be done only once and then the model can be fitted several time, actually you can try several models if you want.

Non-linear wavelength solution:

The method for recording the non-linear wavelength solution is actually very simple. It requires: `GSP_FUNC` which stores a string with the name of the mathematical model from `astropy.modeling.models`. `GSP_ORDR` stores the order or degree of the model. `GSP_NPIX` stores the number of pixels in the spectral axis. Then there is N+1 parameter keywords where N is the order of the model defined by `GSP_ORDR`. The root string of the keyword is `GSP_C` and the rest is a zero-padded three digit number starting on zero to N. See the example below.

```
GSP_FUNC= Chebyshev1D          / Mathematical model of non-linearized data
GSP_ORDR= 3                    / Mathematical model order
GSP_NPIX= 4060                 / Number of Pixels
GSP_C000= 4963.910057577853    / Value of parameter c0
GSP_C001= 0.9943952599223119   / Value of parameter c1
GSP_C002= 5.59241584012648e-08 / Value of parameter c2
GSP_C003= -1.2283411678846e-10 / Value of parameter c3
```

Warning

This method has been developed and tested to write correctly polynomial-like models. And ONLY reads `Chebyshev1D` models. Other models will just be ignored. More development will be done based on request, suggestions or needs.

Creating Your Own Reference Lamps

If you use a Custom mode you will need a Custom reference lamp too. You are welcome to try to do it, just make sure the Header Requirements are met. If not we might be able to help, depending on time availability, it will help a lot if you have the lines identified already.

> **Important**
>
> We are not ready to offer a simple tool to construct the reference lamps.

Cosmic Rays Removal

The default cosmic ray removal tool has not been written in Python (neither for Python). However it has been successfully integrated into the Pipeline's workflow using `subprocess`. Despite the integration being successful, the fact that is a `C` program, designed to be run *standalone* from a terminal, it comes with several challenges in terms of flexibility. The addition of keywords has been dealt with by creating a modified version of the software (see Install DCR). But one of the trickiest part, and probably even more important, is the *parameter configuration*. It uses an external ASCII file and that file HAS TO be named exactly `dcr.par`, also that file has to be located on the folder where the program is called from (usually the data location folder).

The default parameters work very good for the default configuration of the Goodman HT Spectrograph (Spectroscopic 1x1 for instance), but if you have binning 2x2 you will need to tune the parameters.

Since you can't parse the parameters as arguments the only option is to change the values on the `dcr.par` file.

If you use several configurations your best option is to store custom `dcr.par` files and use the argument `--dcr-par-dir <dcr-par-location>` to tell the pipeline where to look for a `dcr.par` file, with the limitation that is has to be called `dcr.par`.

The pipeline will copy the file to the folder where the data is stored.

Headers

The pipeline adds several keywords to keep track of the process and in general for keeping important information available. In the following table is a description of all Goodman Spectroscopic Pipeline keywords added, though not all of them are added to all the images.

General Purpose Keywords

These keywords are used for record purpose, except for `GSP_FNAM` which is used to keep track of the file name.

Keyword	Purpose
GSP_VERS	Pipeline version.
GSP_ONAM	Original file name, first read.
GSP_PNAM	Parent file name.
GSP_FNAM	Current file name.
GSP_PATH	Path from where the file was read.
GSP_TECH	Observing technique. Imaging or Spectroscopy.
GSP_DATE	Date of processing.
GSP_OVER	Overscan region.
GSP_TRIM	Trim section.
GSP_SLIT	Slit trim section. From slit-illuminated area.
GSP_BIAS	Master bias file used.

GSP_FLAT	Master flat file used.
GSP_NORM	Master flat normalization method.
GSP_COSM	Cosmic ray rejection method.
GSP_WRMS	Wavelength solution RMS Error.
GSP_WPOI	Number of points used to calculate RMS Error.
GSP_WREJ	Number of points rejected from RMS Error Calculation.
GSP_DCRR	Reference paper for DCR software (cosmic ray rejection).

Non-linear wavelength solution

Since writing non-linear wavelength solutions to the headers using the FITS standard (reference) is extremely complex and not necessarily well documented. We came up with the solution of simply describing the mathematical model from `astropy.modeling.models`. This allows for maintaining the data *untouched* while keeping a reliable description of the wavelength solution.

The way it is currently implemented will work for writting for any polynomial kind of model. Reading is implemented only for `Chebyshev1D` which is the model by default.

Keyword	Purpose
GSP_FUNC	Name of mathematical model. `astropy.modeling.models`
GSP_ORDR	Order of the model used.
GSP_NPIX	Number of pixels.
GSP_C000	Value of parameter `c0`.
GSP_C001	Value of parameter `c1`.
GSP_C002	Value of paramter `c2`. This goes on depending the order.

Combined Images

Every image used in a combination of images is recorded in the header of the resulting one. The order does not have importance but most likely the header of the first one will be used

Keyword	Purpose
GSP_IC01	First image used to create combined.
GSP_IC02	Second image used to create combined.

Detected lines

The *reference lamp library* maintains the lamps non-linearized and also they get a record of the pixel value and the equivalent in angstrom. In the following table a three-line lamp is shown.

Keyword	Purpose
GSP_P001	Pixel value for the first line detected.
GSP_P002	Pixel value for the second line detected.
GSP_P003	Pixel value for the third line detected.
GSP_A001	Angstrom value for the first line detected.
GSP_A002	Angstrom value for the second line detected.
GSP_A003	Angstrom value for the third line detected.

Running the pipeline in the SOAR data reduction computer

The Goodman Spectroscopic Data Reduction Pipeline has been installed on a dedicated computer at SOAR. The procedure is to open a VNC session, for which you need to be connected to the SOAR VPN. The credentials for the VPN are the same you used for your observing run, provided by your *Support Scientist*, who will also give you the information for the data reduction computer VNC connection.

Note

IRAF is available in all three data servers. Running `iraf` will open an *xgterm* and *ds9* windows. `iraf-only` will not open *ds9*

Establish a VNC connection

Separately, you should receive a server hostname, IP, display number and VNC-password. If you don't you can ask for it. We have decided to use a similar organization of vnc displays as for `soaric7`:

VNC display number and working folder assigned to each partner.

Display	Partner/Institution	Folder
:1	NOAO	`/home/goodman/data/NOAO`
:2	Brazil	`/home/goodman/data/BRAZIL`
:3	UNC	`/home/goodman/data/UNC`
:4	MSU	`/home/goodman/data/MSU`
:5	Chile	`/home/goodman/data/CHILE`

For the rest of this tutorial we will assume your host name is `vnc-server` the display number is `1` and your password is `password`. Though we recommend using RealVNC, most other VNC clients will work fine (e.g., Remmina in Linux). For GNU/Linux and Mac OSX machines we suggest the RealVNC Viewer client. For Windows machines, we suggest either the RealVNC Viewer client or the UltraVNC viewer client. We also know that Vinagre and vncviewer on GNU/Linux work fine.

VNC from the Terminal

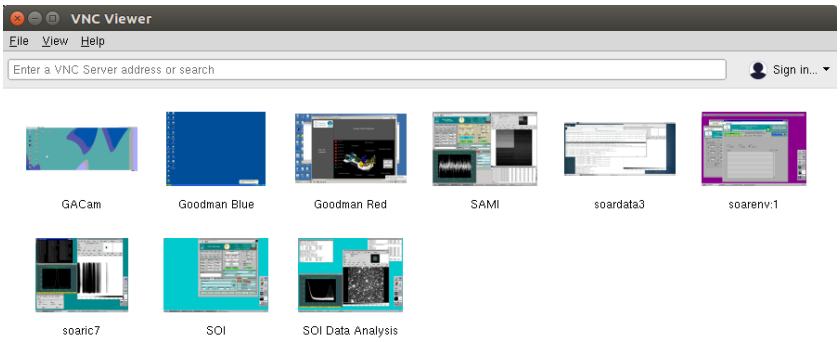
Open a terminal, and assuming you have installed `vncviewer`.

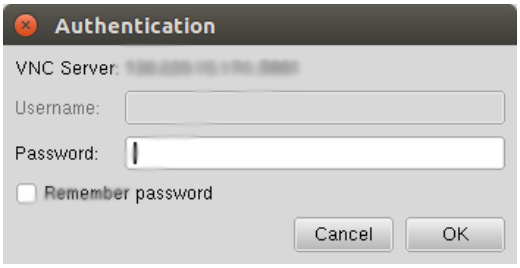
```
vncviewer vnc-server:1
```

You will be asked to type in the *password* provided.

VNC using a Graphical Client

Using a graphical VNC client is quite similar and intuitive





In this case the *IP address* was used, which is equivalent and sometimes better.

Dealing with Virtual Environments

The Goodman Spectroscopic Pipeline uses virtual environments since they allow for easy portability and also give a higher level of safety for the host system. If you know nothing about them, we encourage you to do a quick search to get at least the basics.

All terminals using bash are configured to start with the appropriate virtual environment activated by default. The default virtual environment is called `astroconda`.

If for any reason it is not activated you will need to activate it in order to use the pipeline.

```
source activate astroconda
```

Or if for some reason you need to deactivate it, with the virtual environment activated you do:

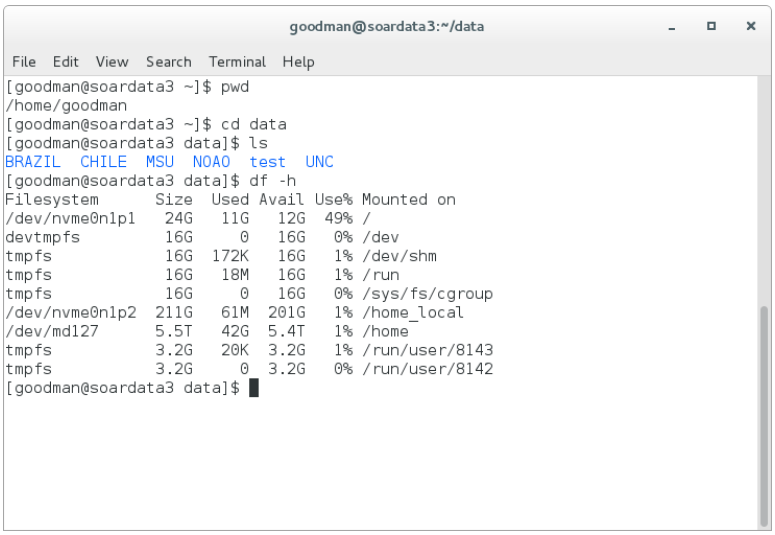
```
source deactivate
```

Running the Pipeline

1. Open a Terminal
2. Go to `/home/goodman/data`

   ```
   cd /home/goodman/data
   ```

3. Here you have a workspace to put your data according to your institution.



4. Create a data folder inside your workspace.

   ```
   cd NOAO
   ```

   ```
   mkdir 2017-07-05
   ```

   ```
   cd 2017-07-05
   ```

5. Copy your data from Goodman Computer

   ```
   scp observer@soaric7:/home3/observer/GOODMAN_DATA/NOAO/2017-07-05/ ./
   ```

6. Make sure you have a full data set. At this point your observing logs will become very useful, eliminate focus sequence, aquisition exposure and any other file present that will not be needed for the processing. The following list summarizes the kind of data that you need to fully process your data.

   - BIAS: Bias
   - FLAT: Flats
   - COMP: Comparison Lamps
   - OBJECT: Science Frames

   Also make sure your data has the same *readout speed*, *binning*, and *ROI*. If you used different configurations during the same night, we recommend you to set up a separate folder for each.

7. Run `redccd`:

   If you are running `redccd` for the first time you can use `redccd` alone but if it's a second or third time you will need to use `--auto-clean` which is a built-in protection for your data, in case you don't want to delete what has been done. Also you might want to consider `--saturation <new value>` to change the saturation level if you get all your flats rejected due to saturation. Sometimes there is a hot column at the end that produced very high values.

   ```
   redccd --auto-clean
   ```

   In case you want to use `--saturation` here is an example:

   ```
   redccd --auto-clean --saturation 70000
   ```

   This changes the saturation level to `70000 ADU` in this context the saturation value works as a threshold for rejecting images and it varies from one instrument configuration to another.

By default, `redccd` puts reduced data in a subdirectory `RED`, you can provide a different one by using `--red-path`.

An image `image_file.fits` that has been fully (and properly) processed should have the new name (including the reduced data folder `RED`):

```
RED/cfzsto_image_file.fits
```

8. Run `redspec`:

   All data processed with the `redccd` (previous step) will be saved in a separate folder this was intentionally made this way in order to avoid overwriting your original data.

   First, move to the folder were the data was stored.

   ```
   cd RED
   ```

   By default `redspec` will search for images with the prefix `cfzsto`, in case you have produced a different prefix you can change it by using `--search-pattern`

   You can just run `redspec` in case everything is the default but if this is the first time you run the pipeline we suggest:

   ```
   redspec --plot-results
   ```

   In that way two important plots will be shown full screen, the comparison lamp fitted to a reference comparison lamp and some values for the wavelength solution fit and the extracted spectrum plotted with the wavelength solution.

   Before the wavelength solution is calculated, the extracted spectrum (1D already) is saved with an `e` as prefix. The final image has a `w` added to the start of the name, following the above example your final 1D and wavelength calibrated image will be named:

   ```
   wecfzsto_image_file.fits
   ```

9. Finally, review the results. Below is a table with the definition of all letters used in the construction of the prefix.

   The meaning of every letter in `wecfzsto` is summarized in the following table:

Letter	Meaning
w	Wavelength calibrated.
e	Extracted spectrum, 1D, no wavelength solution.
c	Cosmic ray cleaned or mask created depending on the method.
f	Flat corrected.
z	Zero or Bias corrected.
s	Slit trimmed, trims off the non-illuminated sections of the detector.
t	Initial Image trimming.
o	Overscan corrected.

Troubleshooting

- The wavelength Solutions is way off: Check that the lamp was correctly registered in the header. Also check that the corresponding reference lamp exist. for instance is not the same to have `HgArNe` to `HgAr`
- Can't detect any objects: Check that the keyword `OBSTYPE` is correct.
- The reference data plot, in interactive mode, doesn't show anything or only vertical dotted lines: The reference lamp doesn't exist for that configuration, since this is used only for visual reference sometimes it will display the same lamp but in other instrument configuration, this will not affect the quality of the solution.

Apendix A: Installation Instructions

We strongly recommend installing the pipeline using *virtual environments*. Below you will find a summary of installation steps.

> ***Warning***
>
> Remember that we are not providing any kind of support for installation. This documentation will be the only existing.

The following list provides a summary of all the steps.

- Install `anaconda`
- Add `astroconda` channel
- Create virtual environment
- Activate environment
- Install requirements
- Install pipeline

Anaconda and Virtual Environment

For anaconda installation we recommend you to check the astroconda channel's documentation page. The instructions will be reproduced here but they might change for newer versions. Also they are limited to the *best case scenario*.

> ***Warning***
>
> Anaconda installer requires BASH. Don't try with other shell.

1. Installing `anaconda` - Go to https://www.anaconda.com/downloads and download the appropriate *anaconda installer* for your platform, most likely it has been automatically selected.
   - Run the installer.

     ```
     cd <download_directory>
     ```

     ```
     bash <install_script>
     ```
   - Once completed, check the bottom of `~/.bash_profile` or `~/.bashrc` there should be a new `PATH` definition with anaconda included.
2. Check `anaconda` installation

   ```
   which conda
   ```

   You should get a response similar to this:

   ```
   ~/bin/anaconda3/bin/conda
   ```

   If you don't get this response check the detailed instructions on the astroconda site. Otherwise continue to the next step.
3. Configure Conda to use the *Astroconda Channel*

   ```
   conda config --add channels http://ssb.stsci.edu/astroconda
   ```
4. Create a virtual environment. We have dropped support for python2.7 so you have to use >3.5.

   ```
   conda create -n astroconda python=3 stsci
   ```

   astroconda is the name of your environment, you can use any name you want.

5. Activate your environment.

   ```
   source activate astroconda
   ```

Goodman Spectroscopic Pipeline

6. Get latest release of the *Goodman Spectroscopic Pipeline*

   visit https://github.com/soar-telescope/goodman/releases/latest and download the `*.zip` or `*.tar.gz`

   ```
   cd <download_location>
   tar -xvf <pipeline_file>.tar.gz
   ```

   or

   ```
   unzip <pipeline_file>.zip
   ```

7. Install requirements from `requirements.txt`

   ```
   cd <goodman_pipeline_unpacked_location>
   pip install -r requirements.txt
   ```

8. Install the pipeline

   ```
   pip install .
   ```

9. Upgrading the pipeline

   ```
   pip install . --upgrade
   ```

Install DCR

> ***Warning***
>
> Don't forget to cite: Pych, W., 2004, PASP, 116, 148

In terms of cosmic ray rejection we shifted to a non-python package because the results were much better compared to LACosmic's implementation in astropy. LACosmic was not designed to work with spectroscopy though.

The latest version of the Goodman Spectroscopic Pipeline uses a modified version of `dcr` to help with the pipeline's workflow. It is included under

```
<path_to_download_location>/goodman/pipeline/data/dcr-source/dcr/
```

`goodman` is the folder that will be created once you untar or unzip the latest release of the *Goodman Spectroscopic Pipeline*.

> ***Important***
>
> The changes includes deletion of all `HISTORY` and `COMMENT` keywords, which we don't use in the pipeline. And addition of a couple of custom keywords, such as: `GSP_FNAM`, which stores the name of the file being created. `GSP_DCRR` which stores the reference to the paper to cite.

You are still encouraged to visit the official Link own by the author and let me remind you once more that you have to cite the paper mentioned several times in this manual.

Compiling DCR

Compiling `dcr` is actually very simple.

```
cd <path_to_download_location>/goodman/pipeline/data/dcr-source/dcr/
```

Then simply type:

```
make
```

This will compile *dcr* and also it will create other files. The executable binary here is `dcr`.

I have successfully compiled *dcr* in several platforms, such as:

1. Ubuntu 16.04
2. Centos 7.1, 7.4
3. MacOS Sierra
4. Solaris 11

Install binary DCR

This is a suggested method. If you are not so sure what you are doing, we recommend you following this suggestion. If you are more advanced user you just need the `dcr` executable binary in your `$PATH` variable.

1. Open a terminal
2. In your home directory create a hidden directory `.bin` (Home directory should be the default when you open a new terminal window)

   ```
   mkdir .bin
   ```

3. Move the binary of your choice and rename it `dcr`. If you compiled it, most likely it's already called `dcr` so you can ignore the renaming part of this step.

   ```
   mv dcr.Ubuntu16.04 ~/.bin/dcr
   ```

   Or

   ```
   mv dcr ~/.bin/dcr
   ```

4. Add your `$HOME/.bin` directory to your `$PATH` variable. Open the file `.bashrc` and add the following line.

   ```
   export PATH=$PATH:/home/myusername/.bin
   ```

   Where `/home/myusername` is of course your home directory.

5. Close and reopen the terminal or load the `.bashrc` file.

   ```
   source ~/.bashrc
   ```

Appendix B: On Goodman's Radial Velocity Precision

> ***Important***
>
> The calculations presented here are for a single line only. Using the `0.45` arcsecond slit.

Here we present a summary of the best *radial velocity* precision that can be obtained with a given configuration. The equations used are listed below.

$$R = \frac{\lambda}{\Delta\lambda} = \frac{c}{v}$$

Then,

$$v = \frac{c}{R}$$

And finally, the

$$v = c\frac{\Delta\lambda}{\lambda}$$

We can calculate the central wavelength for a given configuration which will be λ.

$\Delta\lambda$is the dispersion of the spectrograph in a given configuration and is given in units of *Angstrom/Pixel*. These values are obtained from the Goodman Spectrograph Cheat Sheet.

The smallest slit available is 0.45" wide, then:

$$FWHM = \frac{SlitSize}{PixelScale} = \frac{0.45}{0.15} = 3.0$$

Clearly the *FWHM* is larger by a factor of three to the largest dispersion possible, then the radial velocity precision limiting factor is not the spectrograph's dispersion but the *FWHM* due to slit width, then:

$$\Delta\lambda = FWHM$$

Grating	Mode	Central Wavelength	Dispersion	Resolving Power (R)	RV Limit
400	m1	505.281	1.00	1516	197.773 km / s
400	m2	700.154	1.00	2100	142.727 km / s
600	UV	442.270	0.65	2041	146.867 km / s
600	Blue	492.529	0.65	2273	131.881 km / s
600	Mid	578.827	0.65	2672	112.218 km / s
600	Red	777.885	0.65	3590	83.502 km / s
930	m1	384.521	0.42	2747	109.151 km / s
930	m2	469.125	0.42	3351	89.466 km / s
930	m3	554.787	0.42	3963	75.652 km / s
930	m4	639.418	0.42	4567	65.639 km / s
930	m5	724.936	0.42	5178	57.896 km / s
930	m6	809.084	0.42	5779	51.875 km / s
1200	m0	374.297	0.31	3622	82.765 km / s
1200	m1	424.181	0.31	4105	73.031 km / s
1200	m2	492.678	0.31	4768	62.878 km / s
1200	m3	561.804	0.31	5437	55.141 km / s
1200	m4	629.735	0.31	6094	49.193 km / s
1200	m5	699.087	0.31	6765	44.313 km / s
1200	m6	767.000	0.31	7423	40.389 km / s
1200	m7	835.851	0.31	8089	37.062 km / s