A brief manual and documentation of the

qc suite

Version 1.0

1. Introduction

The qc suite is Python based package to create plots from the quality control pipelines of several ESO instruments. It aims to be as user-friendly as possible. The package is very flexible and automatic. Everything can by done by calling a single Python script, which itself calls all other scripts in the right order with the right parameters. All adjustments to the program can be done in a single configuration file. Furthermore, the qc suite also contains all necessary data for creating the suitable plots for several ESO instruments.

2. Overview

The qc suits consists of six Python scripts, which are all written in Python 2.7. The can be found in the main repository of the package.

- *qc_fetch.py*
- *qc_main.py*
- *qc_parser.py*
- *qc_plotter.py*
- *qc_prepare.py*
- *qc_push.py*

In addition to that there is one configuration file, which is called

- *config.dat*

It contains the basic settings of the suite and can be edited by the user according to his/her needs. Furthermore, there is another file, called

- *time.dat*

This is just a dummy file, which is used to transfer information from *qc_prepare.py* to *qc_parser.py* and shouldn't be changed by the user, although any changes to the file, which are not done during run time of the qc suite, won't affect anything, since this file will be overwritten by *qc_prepare.py* anyway. Moreover the is file, which is called

- *[instrument]_qcdata.dat*

It will contain the required data, which is necessary for doing the quality control plots. The wild-card *[instrument]* stands for the name of the instrument for which the qc suite is run at moment. This file will be reset by *qc_prepare.py*, filled with data by *qc_parser.py* and finally used to produce plots by *qc_plotter.py*. Therefore, it shouldn't be edited by the user. All those files can be found in the main repository, but there are four additional folders aside from it. The contain the data and setting, which are required for the plotting process or are there to archive or save data.

- *instrument*
- *logs*
- *plots*
- *plotter*
- *plotted_data*

3. Installation

The installation package of qc suite version 1.0 contains three folders: *external*, *installation* and *qc_suite*. The folder *qc_suite* contains all the python scripts and all other the files of the qc suite. In addition to those, the qc suite requires some third party software (they can be found in the folder *external*), which are:

- Python 2.7.3
- numpy 1.6.2 (a Python library)
- matplotlib 1.1.1 (a Python library)
- setuptools 0.6c11 (only required for the installation of paramiko on some Linux distribution)
- paramiko 1.7.7.1 (a Python library)

The first step of the installation is to install Python and all the other required packages. The folder *installation* is meant to be the place, where all the third party software should be installed. We will call the path to it *installationpath* from now on. In the next step one simply installs Python by calling in its source directory:
run ./configure –prefix= *installationpath*
make
Then one has to go into the *installationpath* and call:
make install
there. Afterwards one has to create an environment variable for Python. In the next step one should install numpy. This can be done very easily by calling first configure and then install in its source directory. Depending on the Linux distribution, one has to install the setuptools now, which may be required by paramiko. Therefore, one has to download the .egg file for Python 2.7 from the setuptool webpage and put it into the source directory of setuptool. To conclude the installation of this package one simple has to run the .egg file as shell script. Now one is able to install paramiko, which will be the next step. In order to do this one simple goes to the source directory of paramiko and calls:
easy_install ./
The last package, which is required for the qc suite, is matplotlib. Its installation is rather simple. One goes to its source directory and calls the following commands:

python setup.py build
python setup.py install
After all those packages are installed one can go ahead and configure the qc suite. This is also fairly simple and straightforward. One only has to open and edit *config.dat* according to one's requirements. The main changes, which one has to make, are editing the relative paths to folders and the server connections. There is one change, which has to be done in every script: one has to replace the default Python path in the first line by the *installationpath*. After this one is able to run the qc suite.

4. qc_main.py

The script *qc_main.py* simply calls all other scripts of the qc suite. It requires a parameter with the name of the instrument from the configuration file, because the number of quality control plots depends on the instrument. Furthermore, it needs the path on the server, where the webpage with plots is hosted, its IP address, the username and the password to it. The script will publish the plot files by synchronizing the folder with Garching at its end. This script calls *qc_fetch.py* and *qc_prepare.py* in a row but then does a loop for all quality control files located in the folder *logs* applying *qc_parser.py* on them. Afterwards it creates the required quality control plots by doing a loop with *qc_plotter.py* and by getting the right plotting parameters from the files in the folder *plotter*. The scripts continues by calling *qc_push.py* in a loop for all plots thereby sending them to a server and and moving them to the folder *plots*. The final step in script is publishing the plots on the server.

4. qc_fetch.py

The script *qc_fetch.py* fetches new quality control files from a server. It uses 7 parameters from the configuration file *config.dat*, which are the name of the target folder (it should be *logs* by default and it is not recommended to change this), the path to the files on the server, the server IP itself, the user name, the password, the mode for this script (0: only copy files, which are not in the target folder and fit the restrictions for quality control files, 1: copy all quality control files, 2: just copy the two newest files (from today and yesterday)) and the name of the instrument. The script uses a ssh connection to the server and copies it to a predefined target folder. Usually, this script operates in mode 2, to get the new quality control file. It also copies the file from yesterday, because the date might have changed since the last call (it is intended to call it several times per day), but there have been an update yesterday's file since the last call of *qc_fetch.py*. It should be noted that the script makes use of the library *paramiko* to the handle the connection to the server.

5. qc_prepare.py

The script *qc_prepare.py* makes the final preparations before running *qc_parser.py*. Depending on the settings, it requires four to five para meters from the configuration file. The first parameter defines if (in the case of any number not equal 1) the script shall read a manually defined time interval from *config.dat* or if (the parameter has to be set to 1) it shall just read a time span from it. Of course in the case of the time interval, two values are required, a starting time and an end time of the interval. These values as well as the setting are the saved in the file *time.dat*. In case a time span is chosen, the script

will read the system time and writes it to the file *time.dat* alongside with the setting and the length of the time span, which is to be interpreted as days before today. Another function of the script is to delete the old *[instrument]_qcdata.dat* file and replace it with a new file with the same name, which only contains a proper header, since it will be filled with new data by *qc_plotter.py* anyway.

6. qc_parser.py

The script *qc_parser.py* gets data from the quality control files and parses everything, which is required for the plotting process into the *[instrument]_qcdata.dat* file. It is the only script, which doesn't need any data from the main configuration file *config.dat*, though it has to read some parameters from *time.dat* and from one of the instrument configuration files in the folder *instrument*. The name of the instrument and the quality control file which shall be processed are handed to *qc_parser.py* directly by *qc_main.py*. The instrument configuration file is called *[instrument]2.dat*, where the wild-card *[instrument]* stands for the name of the instrument. The file *time.dat* contains information on the time interval for which the plots are going to be created. If the given quality control file doesn't contain any data from within the time interval, the script will simply write nothing to the *[instrument]_qcdata.dat* file. But if there is some data from the right time span in the quality control file, then it will append all data which is necessary for the plotting process to the *[instrument]_qcdata.dat* file. It should be noted that the script internally works with Julian date. Furthermore, it is important to keep in mind, that *qc_parser.py* has to be called for every quality control file in order to get more than just one data point in the plot later on. Usually, this is automatically done by *qc_main.py*.

7. qc_plotter.py

The script *qc_plotter.py* creates a plot based on the data in the *[instrument]_qcdata.dat* file and the setting of the script itself. It only requires one parameter from the configuration file, which is 1 if one doesn't want to plot to be displayed on the screen. Furthermore, the scripts reads some data from another instrument calibration file called *[instrument].dat*, which is located in the folder *instrument*. The settings for *plotter.py* can be found the folder *plotter*. They are called *plotopt_[instrument] [number].dat* and the wild-card *[instrument]* stands for the name of the instrument as usual, but the wild-card *[number]* is a simply an integer number running from 1 to the number of quality control plots, which are necessary for the instrument. The plotting configuration file contains information how the plots should look like. Their names are handed to *qc_plotter.py* by *qc_main.py* directly. Internally, the scripts simply creates a set plots for each chip of the instrument using features of the library *matplotlib*. It saves the plot to file (*.png*), which name is defined in the plotting configuration file and it will be represented by the wild card *[plotname]* here. Furthermore, it creates two files in the folder *plotted_data*, which are called *[plotname]_plotted.dat* and *[plotname]_outliers.dat*. These files contain all plotted points and all found outliers respectively. Moreover, *qc_plotter.py* also searches the folder *plotted_data* for files, which are entitled *[plotname]_ignored.dat* and if such a file exists, the script reads a list of data points, which are to be ignored by the program. These points are being still plotted, but marked specially and they are no longer included in the calculations of the outliers or the median. It should be noted that *qc_plotter.py* can also be used without a plotting configuration file, if one sends the required data as options when calling the script.

8. qc_push.py

The script *qc_push.py* sends a plot file to a server. In order to do so, it requires some data from the configuration file, such as the path on the server, the IP address of the server, the username and the password to it. The name of the file, which is to be sent to server, is handed to the script directly by *qc_main.py*. It should be noted that the file name will be the same on the server. The plot file is then moved from the main repository of the qc suite to the folder *plots*. The sending process to the server is done the library *paramiko*. The shell command *mv* is used to move the plot files to their folder afterwards. Furthermore, the fifth parameter is used to create a *[instrument]_qcdata.dat* file with the right header for the instrument in use. Therefore, it also reads the required information from an instrument configuration file called *[instrument]2.dat*, which is located in the folder *instrument*.

9. The configuration file

The configuration file *config.dat* contains data, which is required by five of the six Python scripts. It is freely editable by the user and one can adjust some general features of the qc suite by it. It is very important that every data stands in the correct line in the file. Therefore, the structure of the configuration file is given here:

line 1: [header line – no data] (--- configuration for qc_fetch ---)
line 2: [name of the folder of the logs file on the computer] (logs)
line 3: [name of the folder of the logs file on the server]
line 4: [IP address of the server]
line 5: [username]
line 6: [password]
line 7: [mode for the qc_fetch] (2)
line 7: [header line – no data] (--- configuration for qc_prepare ---)
line 8: [1 if a time span before today selected, other if predefined a time interval is selected]
line 9: [length of time span] or [starting time of the interval]
line 10: [empty] or [end time of the interval]
line 11: [name of the quality control data file] ([instrument]_qcdata.dat)
line 12: [header line – no data] (--- configuration for qc_plotter ---)
line 13: [1 if qc_plotter shouldn't display plots on the screen]
line 14: [header line – no data] (---- configuration for qc_push ---)
line 15: [name of the folder for the plot files on the server]
line 16: [IP address of the server]
line 17: [username]
line 18: [password]
line 19: [header line – no data] (---- configuration fo qc_main ---)
line 20: [name of the instrument]

10. The instrument configuration files

The basic configuration data for the instrument is contained in the file called *[instrument].dat* and *[instrument]2.dat*, which are located in the folder *instrument*. One can find the chip layout and some other information in the file *[instrument].dat*, which is used by *qc_plotter.py*. The file *[instrument]2.dat* contains the names of the data blocks, which are going to be selected by *qc_parser.py* in order to be used in the plotting process later on.

11. The plotting configuration files

The plotting configuration files, which are usually named *plotopt_[instrument][number].dat*, contain the options for *qc_plotter.py*. The layout of those files is rather simple. One line denotes the name of option (like *[-option]*) and the next line the values assigned to this option. Valid options are:

-help
-qc_par
-xfield
-filter
-yrange
-selectpar
-selectval
-instrument
-filtpar1
-filtval1
-mjdmin
-mjdmax
-upthld
-lowthld
-medplot
-outfile

There are 2 files for Omega-Cam (*ocam*), 5 files for VIMOS (*vimos*), 28 files for VIRCAM (*vircam*) and 49 file for the Wide Field Imager (*wfi*) available in the basic qc suite. Plotting configuration files can be edited and written by every advanced user of the qc suite. Please, keep in mind that the number of files per instrument is written into the code of the *qc_main.py* script and has to be adapted to all changes.

12. Files of plotted points, outliers and ignored data points

The script *qc_plotter.py* creates and uses a whole bunch of files, which are all located in the folder *plotted*. These files, which all have the same format, can be divided into three groups: files of the plotted points, which are labeled *[plotname]_plotted.dat*, files of the outliers, which are labeled *[plotname]_outliers.dat*, and files of ignored data points, which are labeled *[plotname]_ignored.dat*. The files *[plotname]_plotted.dat* are created by *qc_plotter.py* and contain all the points, which were plotted by it. The files *[plotname]_outliers.dat*, are created by the script as well and contain all outliers, which were found. This means, if there are no outliers in the plot, this file will be empty (it

will only contain empty brackets). The files *[plotname]_ignored.dat* are a little different, because they are not created by any script, but they may be created by the user. They contain lists of points, which should be ignored in the analysis, but yet plotted, by *qc_plotter.py*. *qc_plotter.py* will only use such a file, if it exists, else the whole plotting process won't be affected. Concerning the format of these files: every line stands for a chip. Therefore, these files must have as many lines as the instrument has chips. Then each line contains tuples(*x* and *y*) of points. The better illustrate it, we give an example of file for a plot which has 3 chips and two data points each for the first two chips, but none for the last:

[(x1, y1), (x2, y2)]
[(x3, y3), (x4, y4)]
[ ]

When creating a *[plotname]_ignored.dat* file, we recommend to simply copy and modify the *[plotname]_plotted.dat* file or the *[plotname]_outliers.dat* file (which is of course more likely, since one might ignored identified outliers).

13. Other files

Other files which are part of the qc suite are *time.dat* and *[instrument]_qcdata.dat* and of course this manual itself. These file shouldn't be altered by any user, unless you really know what you are doing.

13. Supported instruments

So far, the supported instruments are the Wide Field Imager(WFI), VIMOS, VIRCAM and Omega-Cam. The corresponding wild-cards *[instrument]* for them are *wfi*, *vimos*, *vircam* and *ocam*. But since the whole qc suite is designed rather general, it can be easily adapted for other instruments with the similar quality control files.

14. Credits

The concept of the qc suite was developed by Fernando Selman and the scripts were written by Fernando Selman and Christoph Saulder. This manual was written by Christoph Saulder.

15. Appendix A: Overview of WFI plots

Due to the large number of different quality control plots for the Wide Field Imager, we decided to add a list of the plots here. The number in the list gives you the number in the *plotopt_wfi[number].dat* as well as the number in the *plot[number].png* and all other files created by *qc_plotter.py*.

1. Gain
2. Bias Mean

3. Bias Median
4. Beta Light Mean
5. Shutter Error
6. Linearity
7. Dome Flat Mean in U
8. Dome Flat Mean in B
9. Dome Flat Mean in V
10. Dome Flat Mean in R
11. Dome Flat Mean in I
12. Dome Flat Median in U
13. Dome Flat Median in B
14. Dome Flat Median in V
15. Dome Flat Median in R
16. Dome Flat Median in I
17. Dome Flat Average in U
18. Dome Flat Average in B
19. Dome Flat Average in V
20. Dome Flat Average in R
21. Dome Flat Average in I
22. Sky Flat Mean in U
23. Sky Flat Mean in B
24. Sky Flat Mean in V
25. Sky Flat Mean in R
26. Sky Flat Mean in I
27. Sky Flat Median in U
28. Sky Flat Median in B
29. Sky Flat Median in V
30. Sky Flat Median in R
31. Sky Flat Median in I
32. Zero Point in U
33. Zero Point in B
34. Zero Point in V
35. Zero Point in R
36. Zero Point in I
37. Image Quality
38. Read Noise
39. Beta Light Median
40. Dome Flat Standard Deviation in U
41. Dome Flat Standard Deviation in B
42. Dome Flat Standard Deviation in V
43. Dome Flat Standard Deviation in R
44. Dome Flat Standard Deviation in I
45. Sky Flat Standard Deviation in U
46. Sky Flat Standard Deviation in B
47. Sky Flat Standard Deviation in V
48. Sky Flat Standard Deviation in R
49. Sky Flat Standard Deviation in I