Quality control for WFI with qc_suite

Christoph Saulder and Fernando J. Selman

11/04/12 11:26:49 PM

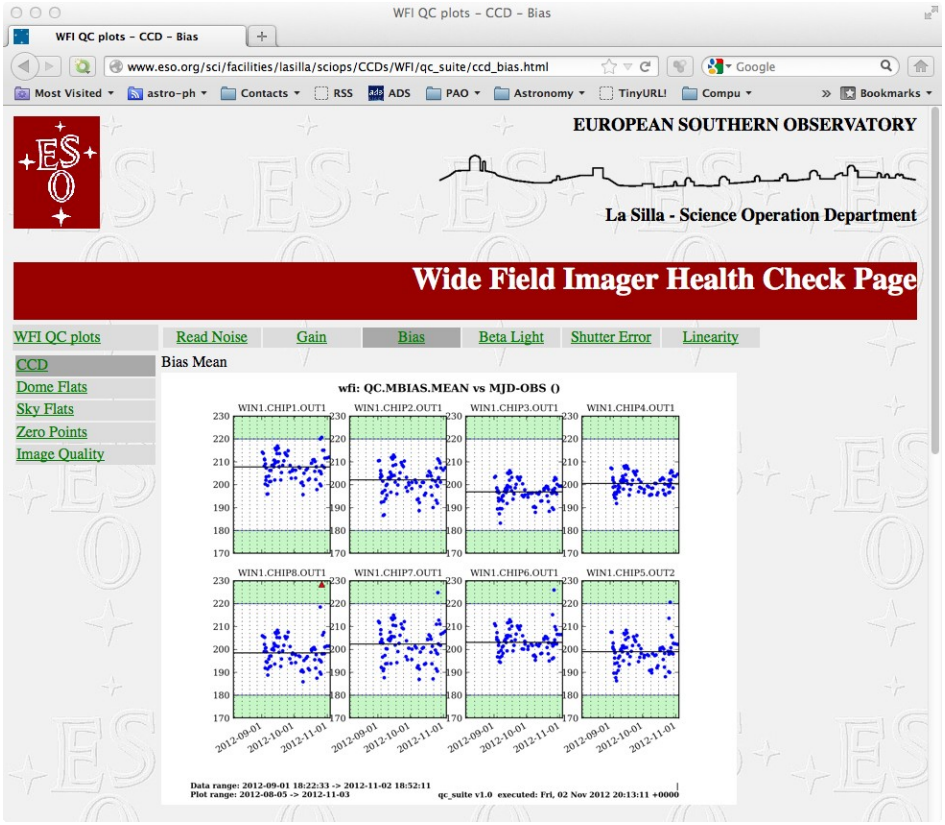


Table of Contents

Introduction

This document gives simple instructions for the use of the new system of quality control for the WFI instrument at the 2.2-m telescope at La Silla. All the parameters that are monitored are generated by the WFI ESO pipeline developed and installed at La Silla by Sandra Castro. This pipeline was a test bench for the OmegaCAM pipeline and shared some of the components.

The pipeline runs automatically in the w2p2pl machine. Every time a frame arrives to w2p2pl a process receives it and classify it according to certain rules. Then when the template that generated that file is finished and produced all the files, then and only then a process is triggered that will eventually result in the reduction of that frame. At the same time the pipeline produces quality control parameters, or QC parameters that are stored in an operation logs file. These files are stored at:

pipeline@w2p2pl:/data/msg. The files are named QC1_WFI.<ISO DATE>.ops.log.

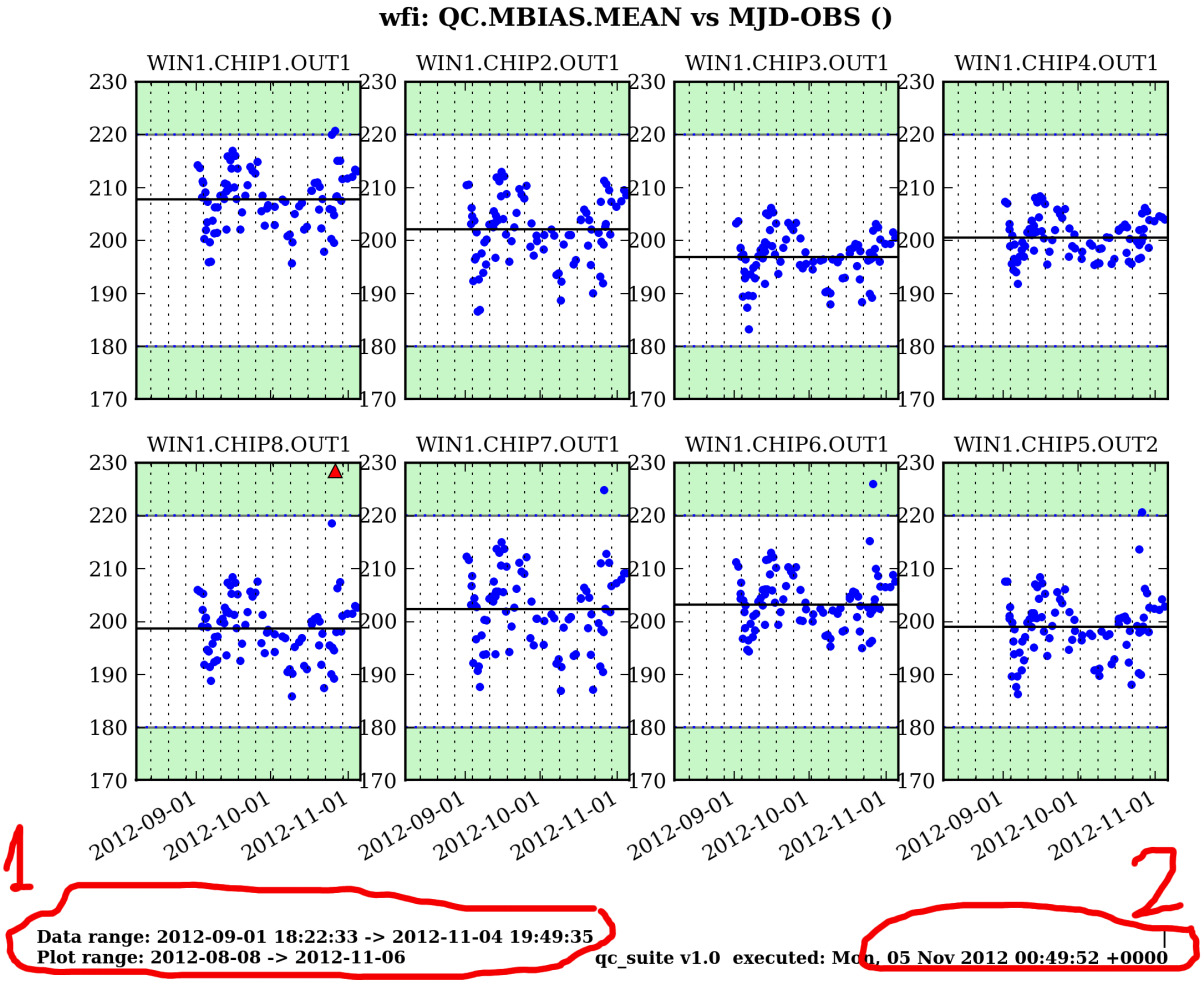
We wrote a set of scripts that run in w2p2off and does the following:

1. fetch the ops.log file from today and yesterday if available and stored them in a local directory
2. runs a parsing script that reads all the opslog within a configurable window and produces a simple ascii table with the QC parameters and their values
3. runs a script that plots each of the QC parameters against date (or any other available parameter in the QC file.
    1. At this stage the plotting program determines if any of the QC points falls above a pre-determined upper threshold, or below a lower threshold. If this is the case then that part of the graph is painted red (it is usually green). These points are written to a file that contains all outliers.
    2. The Instrument Scientist decides if an outlier situation has been resolved and add these points to a file that contains the outliers to ignore. After the next run of qc_main.py the plot area turns green again and a cross is written on top of the outlier point indicating that it has been ignored.
    3. NOT IMPLEMENTED YET: use green color for the label of graphs that pass quality control, and red for those that does not.
4. pushes the recently created graphs to a qc_suite directory in the La Silla web machine.
5. Synchronize this directory with the Garching web server

The La Silla operators are supposed to use these web pages to check that all parameters are within the predetermined ranges. We would like to note that no automatic QC checking system is 100% reliable, and that there is no replacing the frequent spot checking of the data as it is been displayed, and then a look at the reduced products. We emphasize this human intervention as there are effects, such as those we have experience with the recent contamination of the instrument, that are only caught by the ability of the eye to see subtle changes in the background or PSF of the stellar images.

What to check?

The format of the QC plots is shown in the image below:



The first things to check are the date range of the plotted data and of the plotting window. In normal operations should be a running window of 90 days up to the date qc_main was run. These data are labeled 1 in the plot with a red marker. The other date to check is that labelled 2 and which corresponds to the UT time in which the qc_suite was run.

The plots show a central white canvas with the allowed range for what is considered good data quality (note that not all plots have been properly configure in this respect yet). The areas painted green show areas were no point should be found. If there is an outlier there then the area will turn red (this is not properly working yet). These outliers can be eliminated with a procedure described below.

The current version of the qc_suite system for the WFI produces 57 graphs. Not all of them should be checked by all people. So here is a list of responsibilities.

Checks done by the day crew

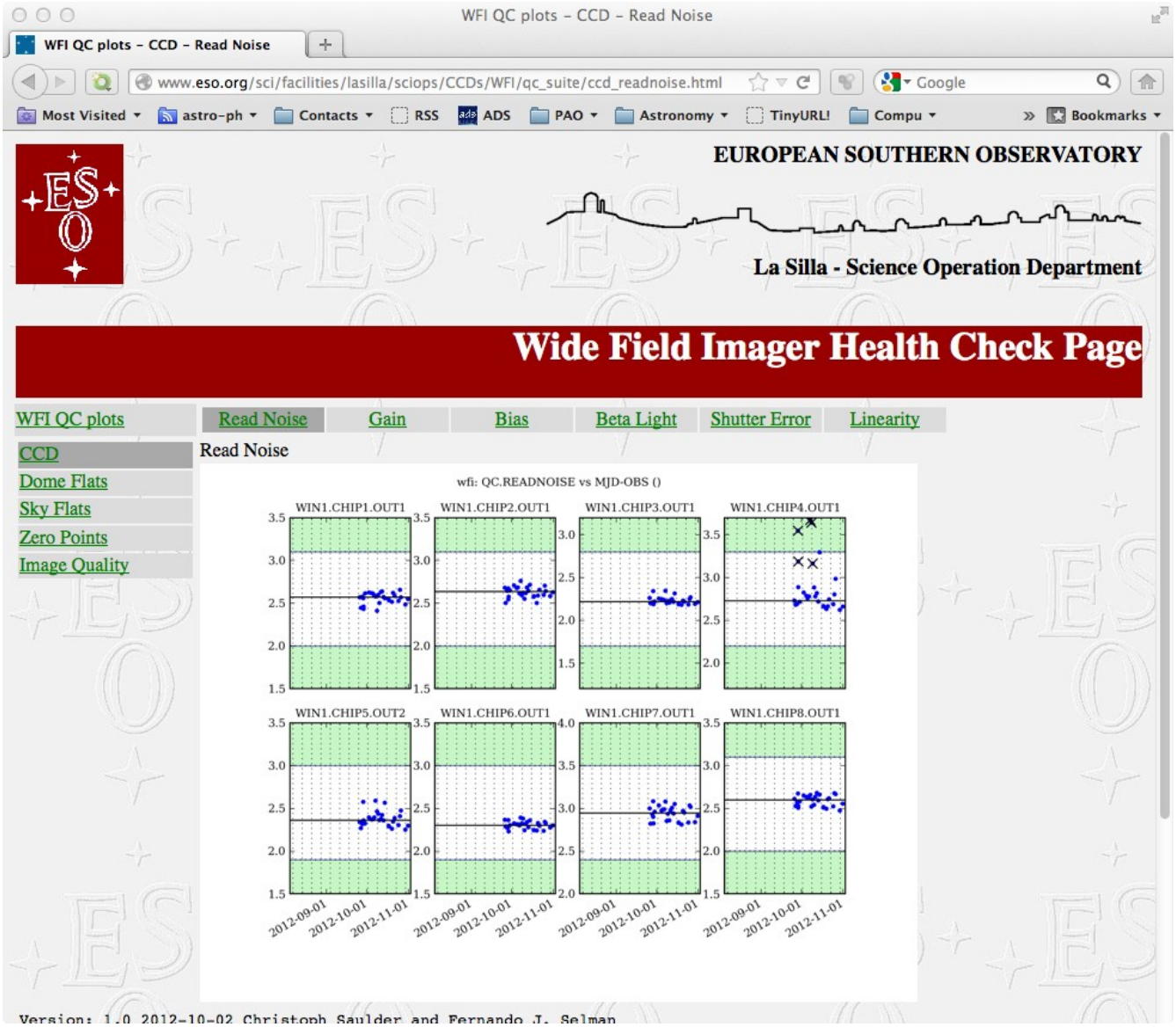
The day crew should take a quick look at the following graphs after running the daily health check (~30 minutes afterwards):

Readout noise

Frequency: daily.

Comment: The graph below show that there are some anomalies in the 4th chip. This merits contacting the electronic engineer and the IS. As can be seen in these graphs that chip is the only one that show these anomalies.

Action on detected anomalies: repeat health check and if they persist contact electronic engineer and IS

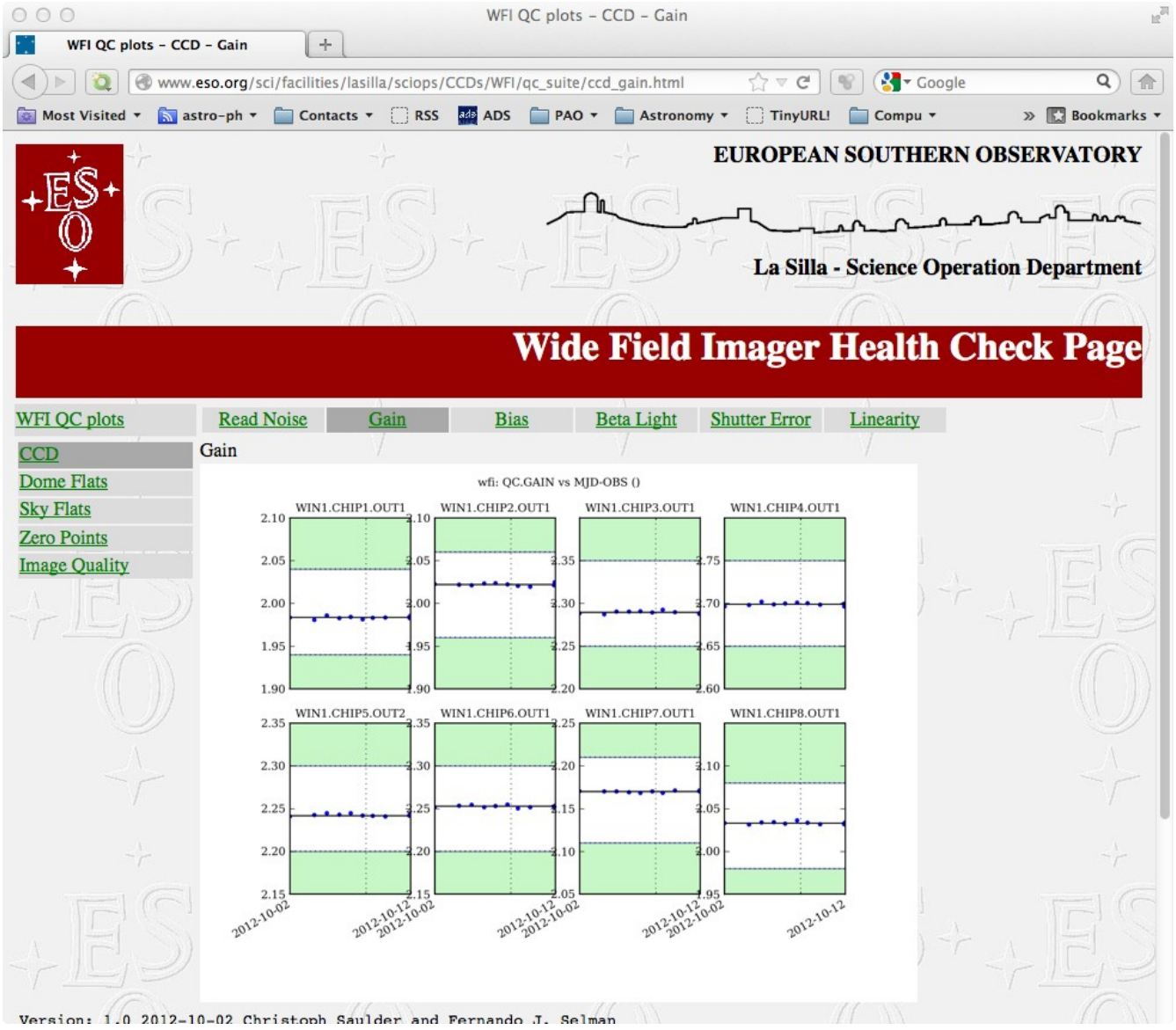


Gain

F*requency:* daily.

Comment: It is an important parameter as the quality of the photometry depends directly on the stability of this parameter. If this parameter changes then one can not use flat fields, or standard star observations from when it was different.

Action on detected anomalies: repeat health check and if they persist contact electronic engineer and IS

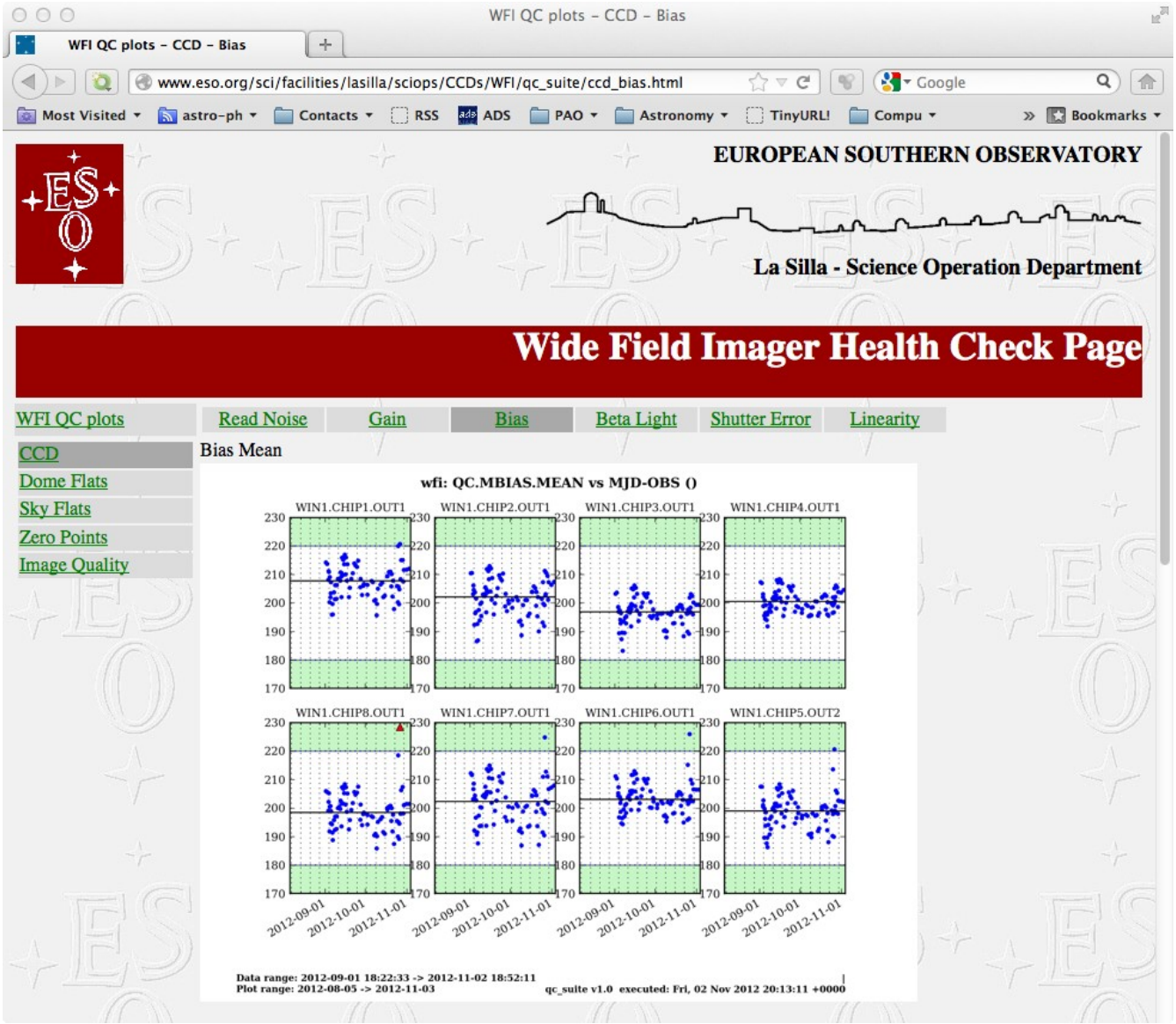


Bias level

Frequency: daily

Comment: the bias level show considerable daily and seasonal variations as it depends on the temperature of the FIERA electronics.

Action on detected anomalies: first check whether the temperature has been extreme. If not repeat the health check. If persistent alert the electronic engineer and the IS.

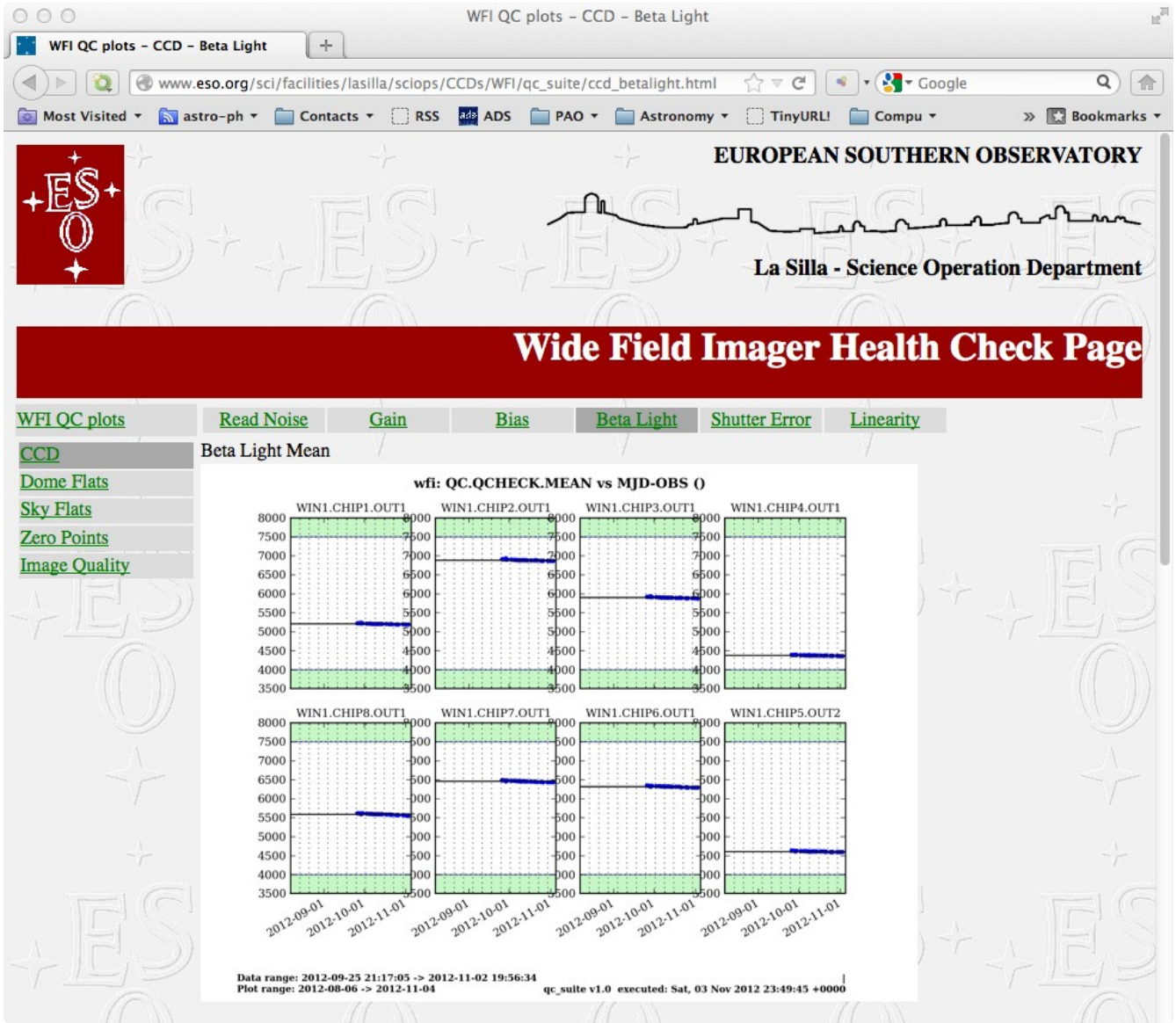


Beta light

Frequency: daily

Comments: the previous parameters checked only the electronics. This one also includes part of the instrument. The beta light is mounted in the filter wheel and illuminates the detector with a very stable illumination (actually an illumination which decreases with a half life of 12.32 years, that is, after 12.32 years it will have half its intensity. Thus, it is not really stable, but it is very predictable. Note that this source is used for the daily and weekly health check. The day crew should only monitor this intensity graph. The IS will be monitoring the linearity graphs.

Action on detected anomalies: repeat the health check and if the anomaly persists contact the electronic engineer and the IS.

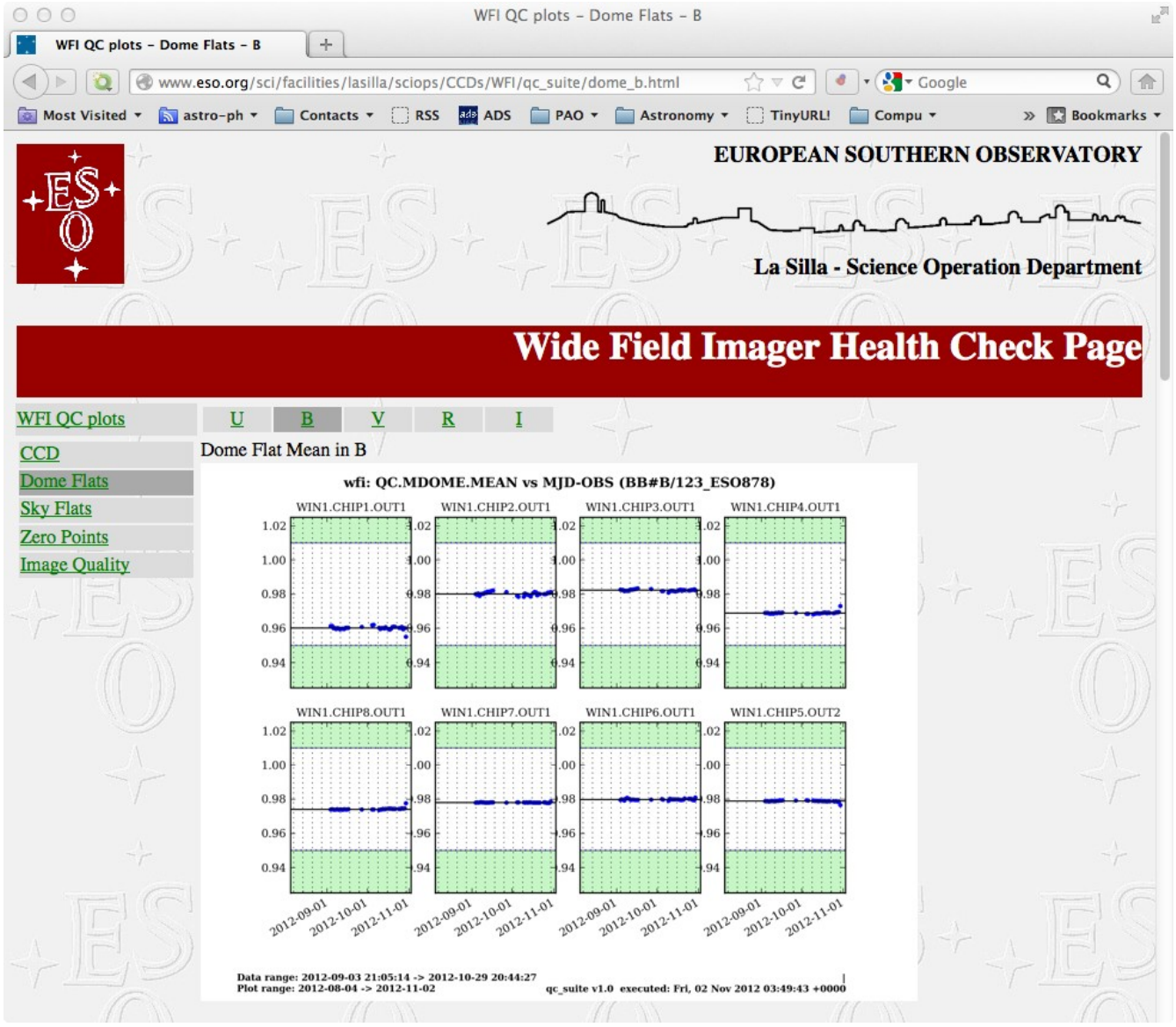


Dome flats

Frequency: every day they are taken

Comments: this is the only day calibration for which the whole light path is checked. The operator should become familiar with how the different dome flats normally look. The plots should be used to detect gross anomalies, such as when taking a dome flat with M1 cover closed (which off course never happens :-)).

Action on detected anomalies: repeat the offending dome flat. If anomaly persist check M1 cover, and the light source. It might be time to change the lamps.



Checks done by the night observer

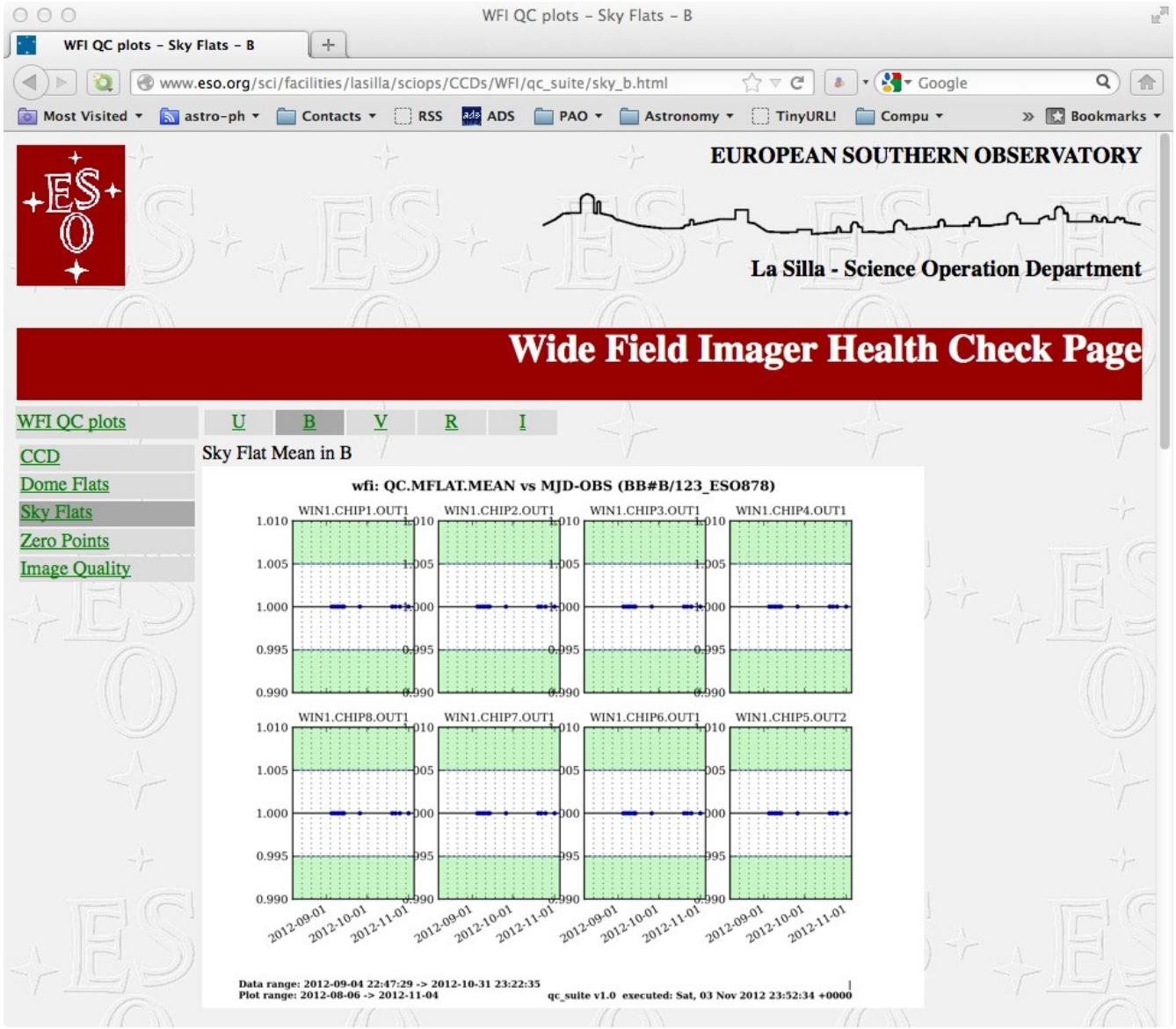
The night crew should take a quick look at the following graphs after executing the sky flat OB (~30 minutes afterwards). Currently these graphs are quite lame, but at least will show major problems. If the plot shows that the sky flat was not reduced then it is time to look at the logs of the pipeline machine to identify the source of the problem.

Sky flats

Frequency: on demand.

Comment: The graph below are quite constant because the pipeline reduces the data one chip at a time and normalize it to unity. A better idea of problems can be found on the standard deviation plot.

Action on detected anomalies: repeat health check and if they persist contact electronic engineer and IS



Zero points

Frequency: on demand.

Comment: these graphs are not really instrument health check graphs because they also contain a contribution from the atmosphere to the variations. These graphs should show quite a bit of variability if the standard star OBs have not been executed exclusively during photometric nights. One would expect a variation of up to 0.5 magnitudes from the photometric line (upper envelope). Another effect is that of the zero point variations across the field. These should result in a maximum spread of 0.2 magnitudes peak-to-valley.

Action on detected anomalies: use it with care to gauge whether the night is photometric. A sudden drop in the zero point, if persistent, should then be informed to the IS.

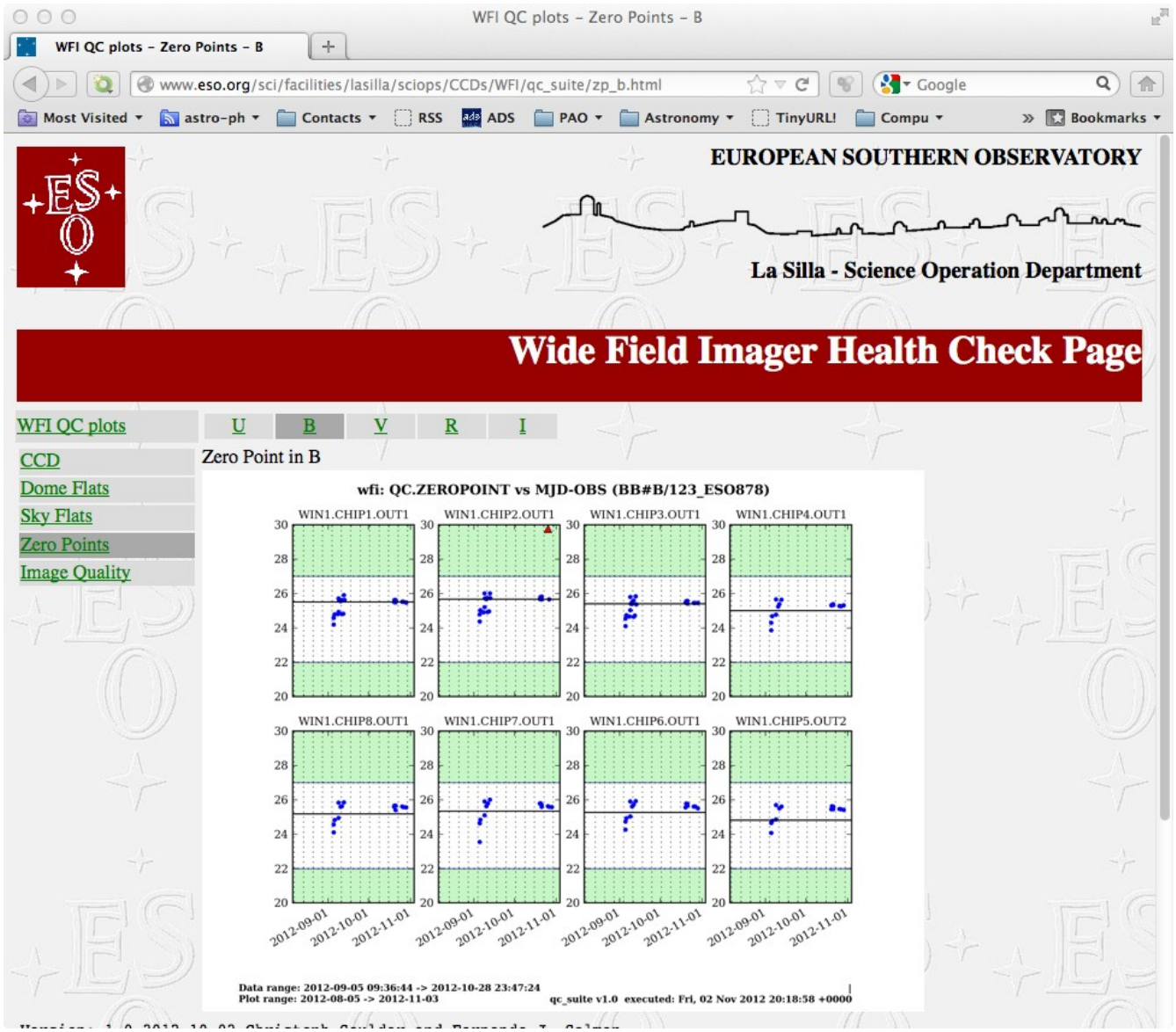
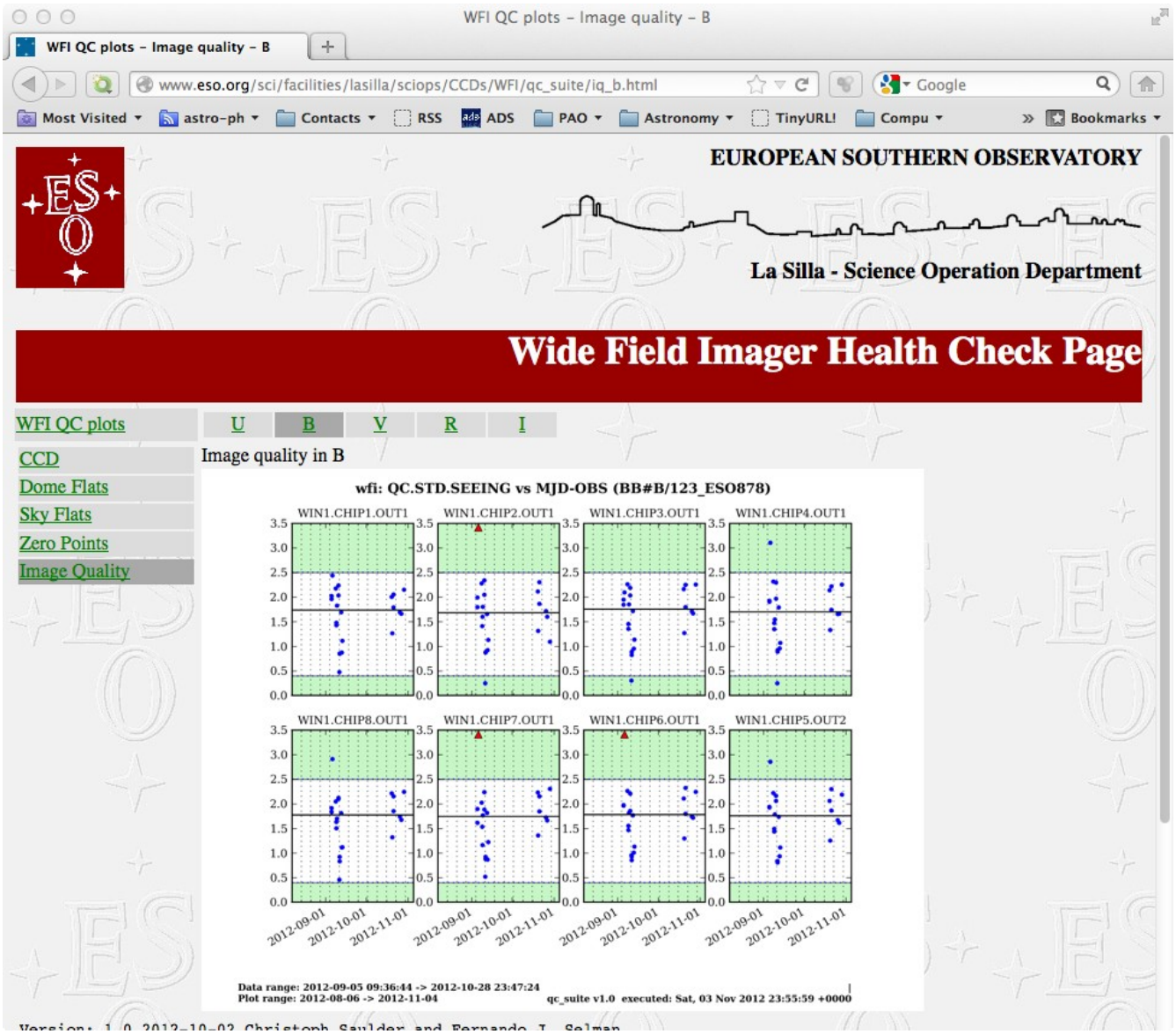


Image quality

Frequency: on demand

Comment: these graphs show the raw FWHM obtained from the standard star OB. It does not correct for airmass or passband. It just give an indication of the quality that the instrument deliver.

Action on anomalies: if the DIMM show low seeing and the instrument deliver degraded image quality: first check that the temperature has not increased: if the M1 is above 2 degrees from the ambient temperature then convection over M1 will degrade image quality. If the temperature is stable check the focus of the instrument and do a sequence.



A brief manual and documentation of the qc suite

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.

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*

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. 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 finishes 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*.

qc_fetch.py

The script *qc_fetch.py* fetches new quality control files from a server. It uses 5 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 and the password. The script uses a heavily modified version of *copytreemod* in order to compare the files (actually just file names) on the server with those, which are already in the target folder, and to copy new files from the server to the computer on which the script is running. It should be noted that the script makes use of the library *pexpect* to send the password, when asked.

qc_prepare.py

The script *qc_prepare.py* makes the final preparations before running *qc_parser.py*. Depending on the settings, it requires three to four 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.

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*.

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 (usually *.png*), which name is defined in the plotting configuration file. 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.

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 using *scp* and the library *pexpect*. The shell command *mv* is used to move the plot files to their folder afterwards.

The main 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: [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]

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.

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 3 file for the Wide Field Imager (*wfi*). 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.

The plotted data, outliers, and ignored data files

These are found in the subdirectory plotted_data in the QC_SUITE_DIR. For each plot there 3 files:

- plot<num>_plotted.dat contains the data been plotted,
- plot<num>_outliers.dat contains the outliers that have been identified by qc_plotter,
- plot<num>_ignored.dat contains the outliers that have been ignored, and that are thus plotted with an x on top.

Note that these files contain for each detector of the instrumen a python list of the form

[(x1, y1), … (xn,yn)],

so they are not very user friendly, but they are not too difficult either.

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.

Supported instruments

So far, the supported instruments are the Wide Field Imager, 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.

Testing mode

To save time when testing the software there are a couple of options that helps:

```
qc_main.py -skip_prepare \'T\' -skip_fetch \'T\' -plotnum 31
```

If run like this the ops.log files will not be fetched, and the qcdata file will not be erased and/or created. It will just use the existing file to create plot 31.

Software configuration

To be able to run the software stand alone there is a variable in qc_main.py that indicates the root diretory of the qc_suite installation. This variable is QC_SUITE_DIR. Note also that the headers of all the python scripts should contain the python interpreter distributed with the suite.

qc_suite installation

qc_suite needs several additional packages that are distributed inside the external directory of the main QC_SUITE_DIR. These packages are:

Python-2.7.3
matplotlib-1.1.1
numpy-1.6.2
paramiko-1.7.7.1
setuptools-0.6c11

The qc_suite scripts are in the src directory. They are

qc_fetch.py
qc_main.py
qc_parser.py
qc_plotter.py
qc_prepare.py
qc_push.py
qc_suite_lib.py
qc_suite_lib.py

The main Python2.7 installation is in the directory QC_SUITE_DIR/installation so the header line for these python scripts should read:

#!/home/astro/qc_suite/installation/bin/python2.7

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 and Fernando Selman.

Appendix

Config.dat for WFI

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.

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]	/data/msg
line 4:	[IP address of the server]	w2p2pl
line 5:	[username]	xxxx
line 6:	[password]	yyyy
		1
line 7:	[header line – no data]	--- configuration for qc_prepare
line 8:	[1 if a time span before today selected,]	1
line 9:	[length of time span] or [starting time of the interval]	90
line 10:	[empty] or [end time of the interval]	
line 11:	[name of the quality control data file] ([instrument]_qcdata.dat)	wfi_qcdata.dat
line 12:	[header line – no data] (--- for qc_plotter ---)	--- configuration for qc_plotter
line 13:	[1 if qc_plotter shouldn't display plots on the screen]	1
line 14:	[header line – no data] (---- configuration for qc_push ---)	--- configuration for qc_push ---
line 15:	[name of the folder for the plot files on the server]	/home/lssciops/sciweb/lasilla/sciops/CCDs/WFI/qc_suite/plots
line 16:	[IP address of the server]	epu.ls.eso.org
line 17:	[username]	xxxx
line 18:	[password]	yyyy
line 19:	[header line – no data]	--- configuration fo qc_main ---
line 20:	[name of the instrument]	wfi

wfi.dat

```
w2p2off astro:~/qc_suite 1216 > cat instrument/wfi.dat
 --- NCHIPS ---
8
 --- chipIndeces ---
WIN1.CHIP1.OUT1
WIN1.CHIP2.OUT1
WIN1.CHIP3.OUT1
WIN1.CHIP4.OUT1
WIN1.CHIP5.OUT2
WIN1.CHIP6.OUT1
WIN1.CHIP7.OUT1
WIN1.CHIP8.OUT1
 --- zcodes ---
2
4
1
 ----------------
2
4
2
 ----------------
2
4
3
 ----------------
2
4
4
 ----------------
2
4
8
 ----------------
2
4
7
 ----------------
2
4
6
 ----------------
2
4
5
 --- xtick_codes ---
0   0  0  0  1  1  1  1
 --- ytick_codes ---
1   1  1  1  1  1  1  1
```

wfi2.dat

```
w2p2off astro:~/qc_suite 1217 > cat instrument/wfi2.dat
zDATE zMJD zobs_id  zExtension zFilter zTPLID zARCFILE
DATE-OBS
MJD-OBS
OBS.ID
EXTNAME
INS FILT1 NAME
TPL ID
ARCFILE
```

Example of plot configuration file for WFI: plotopt_wfi27.dat

```
w2p2off astro:~/qc_suite 1215 > cat plotter/plotopt_wfi27.dat
-qc_par
"QC.MFLAT.MEDIAN"
-x_field
"MJD-OBS"
-filter
"BB#U/50_ESO877"
-instrument
"wfi"
-yrange
[(0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1)]
-lowthld
(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
-upthld
(0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9)
-medplot
True
-outfile
"plot27"
```

#__oOo__