DRESSCode

(Data Reduction of Extended SWIFT Sources Code)

Installation instructions and User manual

Marjorie Decleir

February 2019

Content

A. HEASoft installation instructions

1) Download the software

-Go to https://heasarc.nasa.gov/lheasoft/download.html
-STEP 1 - Select the type of software: Choose Source code distribution.
-Choose your platform, e.g. PC - Linux – Ubuntu
-STEP 2 - Download the desired packages: Select the Swift mission (all FTOOLS and XANADU will automatically be selected).
-Submit and wait.
-Untar the file.

2) Install/Build the software

-STEP 3 - Install the software: Follow the installation guide for your platform, e.g. PC Linux – Ubuntu.
All subsequent steps are for Ubuntu platforms:
-Install the prerequisite packages listed here: https://heasarc.nasa.gov/lheasoft/ubuntu.html
-Build the software following the instructions in the installation guide (https://heasarc.nasa.gov/lheasoft/ubuntu.html)
-Add the following lines to your .bashrc (or equivalent):
export HEADAS=~/heasoft-6.25/x86_64-pc-linux-gnu-libc2.23
. $HEADAS/headas-init.sh

3) Install the calibration tree

-Create a new folder “caldb” in the heasoft-6.25 folder
-Follow the instructions on: https://heasarc.gsfc.nasa.gov/docs/heasarc/caldb/install.html (Sections 2 and 3)

4) Install wcstools

-Go to: http://tdc-www.harvard.edu/wcstools/
-Choose WCSTools FTP → *HTTP from* http://tdc-www.harvard.edu/software/wcstools/wcstools-3.9.5.tar.gz
-Put the tarfile in your home folder and untar
-Install the tools:
cd wcstools-3.9.5
make all
-Add the following line to your .bashrc (or equivalent):
export PATH=~/wcstools-3.9.5/bin:$PATH

5) Download the DRESSCode from GitHub

-Python is needed to run the scripts. The DRESSCode was written and tested in python 3.6.8. Make sure you have a working python environment, preferably using Anaconda (or Astroconda).

B. Download and prepare the SWIFT data

1) Download the data

- Go to NASA's HEASARC Archive: https://heasarc.nasa.gov/cgi-bin/W3Browse/swift.pl
- Next to "UVOT Log", click on "parameter search form" to open the SWIFT UVOT Data Query form.
- Use the following parameters (and leave the other parameters to the default values):

filter	'UVW2' OR 'UVM2' OR 'UVW1'
pointing_mode	POINTING
Object Name	Name of the galaxy, e.g. NGC0628

- Click on the "Start Search" button and wait for the query results to load.
- Check the box "Select All" to select all data.
- At the bottom: Select "Swift Auxiliary Data (aux)" and "Swift UVOT Data (uvot data)".
- Click on the "Create Download Script" button and wait for the new tab to load.
- Click on the "Download Commands To File" button and wait until the script is downloaded.
- Create a new directory somewhere on your computer with the name of the galaxy, e.g. "*NGC0628*", and create a new directory within that directory with the name "*Raw_data*".
- Move the download script to the "*Raw_data*" directory and run it:

  $ bash browse_download_script.txt

  Downloading the data can take a while!
- Delete the download script when it has finished running.

2) Sort and select the required data

- The download script will automatically sort the data into different directories according to the observing ID of the exposure. The directory structure is somewhat complex, but you should not worry about that. If you used the same directory structure as described above, the only thing you need to do to make all scripts work, is to change the name of the galaxy and the path of the main directory (in which the different galaxy directories are located) in the *config.txt* file.
- Run the script `collect_images.py` to collect all the UV raw image (*_rw.img.gz) files from the uvot/image/ folders, the UV event files (*w1po_uf.evt.gz) from the uvot/event/ folders, the attitude (*pat.fits.gz) files from the auxil/ folders and the aspect following (*uaf.hk.gz) files from the uvot/hk/ folders. This will create the directory "*Raw_images*" in the current directory (e.g. "*NGC0628*").
- To save space on your computer, you can now delete the "*Raw_data*" directory, if you want.
- Extract (decompress) all files in the "*Raw_images*" directory. To save space on your computer, you can now delete the compressed (*.gz) files, if you want.
- Copy the entire "*Raw_images*" directory to a new working directory with the name "*working_dir*", in which all the temporary files will be stored during the reduction process. Make sure you always keep the raw files in the "*Raw_images*" directory in case you want to restart.

C. Data reduction with the DRESSCode

1) Sky images part 1

Run the script `uvotimage.py` to create sky images from the raw images and event files. When your data contains event files, you will get the following warning:

uvotimage: skipping event based image HDU w1386106344E
in file sw00032766001uw1_rw.img

This means that the code is skipping the event based frames in the raw images to prevent using the event data twice. This is perfectly normal, and you can thus ignore this warning.

2) Aspect correction part 1

- Run the script `uvotskycorr.py` to calculate an aspect correction for the sky images.
  If no aspect correction could be found, the following warning will appear:

  !! No aspect correction found for frame sw00450884000_evt_uw2_sk.img[1]!!

  To solve this, you can try to increase the number of reference stars used by the task. In the script on line 45 you find: *n.reference=200 n.observation=40. n.reference* is the maximum number of reference stars that will be used from the catalog. *n.observation* is the maximum number of observed stars in the image that will be used to match with the catalog. Increasing one (or both) of these values can help to find an aspect correction. Of course, this will also increase the running time. Too high values can cause the process to crash, for example when your computer is short of memory. Frames for which no aspect correction was found, will not be taken into account in the summed image (see 7).
- Run the script `uvotattcorr.py` to adjust the attitude files with the calculated aspect corrections.

3) Sky images part 2

Run the script `uvotimage2.py` to create new sky images from the raw images, using the updated attitude files.

4) Auxiliary files part 1

- Run the script `uvotbadpix.py` to create quality maps for all sky images.
- Run the script `uvotexpmap.py` to create exposure maps for all sky images, to flag the sss patches in the quality maps, and to create mask maps based on the quality maps.

Small scale sensitivity (sss) patches are detector regions with a lower throughput, probably caused by dust on the photocathode. There is no way to correct for the count loss in the affected pixels. Therefore, the only solution is to mask these regions in the images. The script `uvotexpmap.py` will flag these pixels in the quality maps. At this stage, only raw image files of 1024x1024 or 2048x2048 pixels can be used. Some galaxies have images with a different dimension. For these images, an sss mask cannot be created, because it is a priori not known which part of the detector was exposed. These images should thus not be used in the pipeline. When the script encounters an image with a different dimension, the following warning will appear:

> *Quality map quality_sw00032081026_uat_img_uw1_badpix.img[1] does not have the correct dimensions, and cannot be combined with an sss mask.*

Make sure to delete these images before continuing!

5) Aspect correction part 2

Run the script `uvotskycorr2.py` to calculate an aspect correction and apply it to the sky images, the exposure maps and the mask files, using the updated attitude files.

6) Auxiliary files part 2

Run the script `uvotskylss.py` to create large scale sensitivity (lss) maps for all sky images.

7) Summing images per observing period

- At this stage, you can only sum images that were taken more or less during the same period. Sort the corrected sky images (and their corresponding exposure maps, lss maps and mask maps) into separate sub-folders according to their observation date, e.g. per year (2007, 2013, …). Observation dates can be checked with the script `header_info.py`.
- In the *config.txt* file change the “years” according to the names of the sub-folders you created.
- Run the script `uvotimsum.py` to sum all frames per type and per filter and to normalize the total sky images. Image frames for which no aspect correction was found, will automatically be excluded from the sum.

8) Flux corrections

Run the script `corrections.py` to correct the normalized images for coincidence loss, large scale sensitivity variations and loss of detector sensitivity (i.e. zero point correction).

9) Combining the different observing periods

Run the script `combine.py` to combine the images from the different observation periods.

The script will first align and reproject all corrected images from the different periods before stacking them (per filter). It will use the largest image as the reference image, and will report this as follows:

> *Image sum_uw2_nm_coilsszp_c.img of year 2018B will be used as reference image. Please verify whether this image is large enough to cover all other images.*

Verify whether this image covers the FOVs of all other images that need to be stacked. If this is not the case, change the flag “enlarge” to “yes” in the *config.txt* file, and specify the number of pixels in the x- and y-direction that need to be added to the reference image in order to cover all other images (e.g. add_xpix = 200, add_ypix = 100).

10) Calibration and aperture correction

Use the script `calibration.py` to convert the units of the final images from counts/s to Jy and to perform an “inverse” aperture correction.

D. General notes to the user

- Make sure the HEASoft and WCSTools are correctly installed and initialized (see section A). Without these tools, you won't be able to run the pipeline.
- Make sure you are using Python 3.6, and that all required modules are installed properly. The script will fire an error message if a module or package is missing.
- Please, check the output after each step.
- Questions/problems can be reported through the GitHub issues.