**HSLA Release Instructions**

September 4, 2018

This document describes the initial process of generating a new HSLA release. The steps described below outline the very beginning of the release process all the way up to the coaddition step (drive.py); this step, along with those that follow, will be described in more detail in additional documentation. The examples in this document assume this will be a COS release, but the software is designed to also work for STIS (not yet fully tested). All of the HSLA software is available on GitHub (<https://github.com/spacetelescope/hsla>).

Here is a quick summary of the necessary steps to complete the initial HSLA setup as well as their estimated execution times (each step will be described in more detail below):

1. Run hsla\_setup.py
   1. Generates a new datapile including all of the new and reprocessed COS data since the last HSLA release along with additional files that will be needed in future steps.
   2. Time to execute: Depends on the amount of new/reprocessed data but ~7 hours max.
2. Manually edit the \*new.alias file
   1. Choose the best alias for each new COS target.
3. Run combine\_alias
   1. Combines the chosen new aliases into the reference alias file.
   2. Time to execute: <1 minute
4. Run defeat\_alias
   1. Moves the x1d files from their original target directories into the chosen alias directory.
   2. Time to execute: ~5 minutes
5. Run make\_target\_list
   1. Makes a list of all of the COS targets that have data.
   2. Time to execute: <1 minute
6. Run get\_alt\_names.py
   1. Finds alternative NED/SIMBAD names for all of the COS targets.
   2. Time to execute: highly variable (30 minutes – 6 hours)
7. Run scrape\_headers.py \*\*\*see additional documentation
   1. Makes useful lists, etc. that later codes rely on.
   2. Time to execute: ~20 minutes
8. Run ban\_programs
   1. Bans bad proposals/visits/exposures from being included in the final coadds.
   2. Time to execute: ~20 minutes
9. Run drive.py \*\*\*see additional documentation
   1. Generates the final coadds, plots, webpages, etc.
   2. Time to execute: ~4 hours
10. The first step in any release is to run hsla\_setup.py. This code looks for any new or reprocessed COS data since the previous HSLA release and puts it (in addition to all of the remaining COS data) into a new datapile directory. It also generates a full catalog of all of the COS data, updates the listing of bad COS visits, records the differences between the data in this new release and the previous, sorts the x1ds into their original target directories, and generates alias files of alternative target names for each COS target. Because this code pulls data from the public archives, you need to run it on a server (plhstins1, etc.). This code is designed to build off of the most recent HSLA release rather than starting from scratch each time. To run:

>>> python hsla\_setup.py --b 05-10-2018 --e 09-04-2018 --n datapile\_09-04-2018\_COS

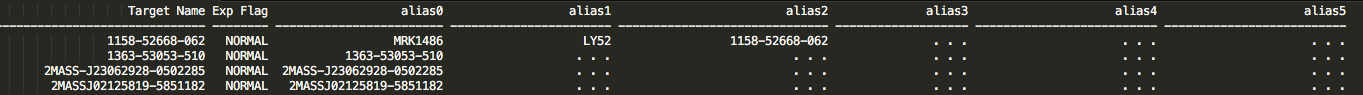
--o datapile\_05-15-2018\_COS --i COS

The above command will generate a new COS datapile named datapile\_09-04-2018\_COS in /grp/hst/HST\_Spectro/hsla\_releases/ that includes all public COS x1ds up to 09-04-2018, sorted into their original target directories. It builds off of information contained in the previous release directory (datapile\_05-15-2018\_COS) which included data up to 05-15-2018\_COS. The inputted date range (05-10-2018 to 09-04-2018) tells the code to look for new/reprocessed data in that time range; to be safe, the beginning date in this range should overlap the range used in the previous release (i.e. to make sure you don’t miss any COS data); if you’re not sure what time was previously used, check out the ‘update\_summary.txt’ file in the previous release’s directory which lists the range that was used in that release.

In addition to pulling new data, this code also generates a number of useful files. The most relevant are described below:

* update\_summary.txt : this file lists all of the new x1ds (those not included in the previous release) and reprocessed x1ds including the reason for reprocessing (e.g. the file is using a new reference file) since the previous release.
* banned\_visits\_COS.txt : this file lists all of the banned visits (i.e. visits to omit from the final coadds) and is generated by looking at the HST program status webpage for any visits that contain the word “Failed” in their status columns. To save time, only the proposals with new data since the previous release are checked, and the results are appended to the previous release’s version of this file.
* COS\_exposures\_Sep4\_2018\_all.txt : a catalog containing information about all of the COS targets in this new release.
* COS\_exposures\_Sep4\_2018\_new.alias : a table showing the new target names in this release as well as potential aliases (i.e. alternative names).

1. The next step is to edit the COS\_exposures\_Sep4\_2018\_new.alias file by choosing the alias you want to use for any target that has multiple aliases. These aliases exist because different PIs call the same target multiple names, so we have to choose the best one to use in HSLA. To choose an alias, simply copy it into the alias0 column (it’s recommended to keep all of the other aliases stored in the alias1-5 for record-keeping). Only those targets with multiple aliases need to be chosen; nothing has to be done for the other targets. For example, in the screenshot below, target 1158-52668-062 has been chosen to be aliased to MRK1486, and no decision has to be made for the others.



1. Once the aliases have been chosen for the new targets, we next run combine\_alias which takes those choices and combines them into a reference alias file that includes the chosen aliases for all of the COS targets, new and old (COS\_exposures\_Sep4\_2018\_final.alias).

>>> import hsla\_finish as h

>>> h.combine\_alias(‘/grp/hst/HST\_Spectro/hsla\_releases/datapile\_09-04-2018\_COS’,

‘/grp/hst/HST\_Spectro/hsla\_releases/datapile\_05-15-2018\_COS’)

where we have passed in the paths to the new and old hsla datapile. Sometimes, a new target name comes in that is better than all of the previous names for that target. In this scenario, the final reference alias file (COS\_exposures\_Sep4\_2018\_final.alias) needs to be manually edited after running combine\_alias, where the alias0 for these older target names gets changed to this new, better name.

1. After we’re satisfied with the final alias file, we run defeat\_alias, which moves all of the x1ds in each target directory into each target’s chosen alias directory. This code should be run within the new datapile directory (e.g. /grp/hst/HST\_Spectro/hsla\_releases/datapile\_09-04-2018\_COS).

>>> import defeat\_aliases as d

>>> d.defeat\_aliases(‘COS\_exposures\_Sep4\_2018\_final.alias’)

1. Once all of the x1ds are moved into their chosen alias directories, we next make a list of all of the targets that have COS data for this release (all\_targets\_COS.list).

>>> import hsla\_finish as h

>>> h.make\_target\_list(‘/grp/hst/HST\_Spectro/hsla\_releases/datapile\_09-04- 2018\_COS’, ‘COS’)

This file contains 1’s next to each target that should be included in this HSLA release (it’s recommended to remove the following target names from this list: ANY, WAVE, CCDFLAT). This file should be placed in a new samples directory for this release (e.g. /grp/hst/HST\_Spectro/hsla\_releases/samples\_09-04-2018\_COS).

1. Next, we run get\_alt\_names which will generate files that list alternative NED/SIMBAD names for all of the COS targets (e.g. target\_altnames\_NED.fits). These can then be used as inputs to scrape\_headers. This code should be run within the new datapile directory.

>>> import get\_alt\_names as g

>>> g.get\_alt\_names(‘COS\_exposures\_Sep4\_2018\_all.txt’, ‘NED’, ‘target\_altnames’)

This also works the same with ‘SIMBAD’.

1. Run scrape\_headers \*\*\*see additional documentation
2. After running scrape\_headers, it’s time to ban certain exposures from being included in the final coadds. This code should be run within the new datapile directory.

>>> import ban\_programs as b

>>> b.ban\_programs(‘../samples\_09-04-2018\_COS/all\_targets\_COS’)

This goes through each ‘all\_exposures.txt’ file in every target directory (these were created in scrape\_headers) and bans certain exposures (e.g. those from calibration proposals or bad visits) by setting their flag to 0.

1. Run drive.py \*\*\*see additional documentation