reducer Documentation

Release 0.3.3

Matt Craig

November 01, 2016

Contents

CHAPTER 1

Want to use reducer in a class?

Want to use reducer in a class and need some help getting started/advice on use? There is a *very* brief summary of how I've used with undergraduates at *Using with reducer with students*.

Please contact me at mattwcraig@gmail.com or open an issue on the GitHub site for reducer if you have questions, run into problems, or want to let me know how you are using it!

CHAPTER 2

Short walkthrough video

The YouTube video below has a walkthrough of the reducer notebook. No explanatory audio or text, check out the *Quickstart* for that.

Contents:

2.1 Quickstart

The `reducer` package generates a widget-based Jupyter notebook for reducing astronomical images. The actual reduction steps are done by ccdproc.

2.1.1 Installation

The recommended way to install `reducer`, especially on Windows, is with the Anaconda python distribution. Several of the packages that `reducer` depends on need to be compiled...and most people haven't installed a compiler on Windows.

Installing with anaconda

1. Download and install the Anaconda python distribution.
2. Depending on your platform:
1. Windows: Open the "Anaconda Command Prompt" from the start menu.
2. Mac: Open the Terminal app (it is in Applications/Utilities)
3. Linux: Open a terminal windows.
3. Install `reducer` by typing, in the terminal: `conda install -c mwcraig -c astropy reducer`

Installing with other python distributions

Remember, this route requires that you have a compiler installed and properly configured. On Windows, you do not have that unless you have set it up.

Install `reducer` with pip:

```
$ pip install reducer
```

2.1.2 Generate a template notebook

To generate a notebook, navigate to the directory in which you want the reduced data to end up. Then, at the command line, type:

```
$ reducer
```

That will create notebook called `reduction.ipynb`. Open that notebook with:

```
$ jupyter notebook
```

2.1.3 Using the notebook

The first rule of using the notebook is to read the text cells of the notebook.

There are three kinds of widgets in the reduction notebook.

Simple image browser

Images are arranged based on the values of keywords in their FITS headers. Clicking on a file name displays the image and a tab for displaying the header.

The FITS keywords used to construct the menu tree at the left are determined by this line in the notebook:

```
tt = msumastro.TableTree(images.summary_info,
                         ['imagetyp', 'exposure'],
                         'file')
```

In this example, images were grouped by `imagetyp` and `exposure`.

Reduction step

Each reduction step (bias subtraction, dark subtraction and flat correction) has a widget to go along with it. The example shown below is for a processing a light (science) image.

The key thing to understand here is how `reducer` is selecting the appropriate master (or synthetic) calibration image for each step. To be considered for matching an image file has to have a keyword `MASTER=True` and the correct `IMAGETYP` for the step (e.g. `FLAT` for flat correction). In addition, for dark subtraction, the master dark frame whose exposure most closely matches the image being reduced is selected. For flat frames the `FILTER` of the flat must match the `FILTER` of the image.

You can select the images to which the reduction step is applied. The widget is created in the notebook with this:

```
light_reduction = astro_gui.Reduction(description='Reduce light frames',
                                      toggle_type='button',
                                      allow_bias=True,
                                      master_source=reduced_collection,
                                      allow_dark=True,
                                      allow_flat=True,
                                      input_image_collection=images,
                                      destination=destination_dir,
                                      apply_to={'imagetyp': 'light'})
```

The `apply_to` argument selects the images to which the the reduction step will be applied. To reduce only V-band images of M101 you could (assuming the appropriate keywords are in the FITS header, of course) use:

```
apply_to={'imagetyp': 'light', 'filter': 'V', 'object': 'M101'}
```

Image combination

Calibration images can be combined to make a master (or synthetic) image. An example of the widget that does that is below, shown for creating master flats.

Note well that this will create *several* flats. To understand which images in the source directory will be identified as flats, how they will be grouped, and what the output files will be called let's look at the notebook code the created the widget above:

```
flat = astro_gui.Combiner(description="Make Master Flat",
                          toggle_type='button',
                          file_name_base='master_flat',
                          group_by='exposure, filter',
                          image_source=reduced_collection,
                          apply_to={'imagetyp': 'flat'},
                          destination=destination_dir)
flat.display()
```

The `apply_to` argument on line 6 controls which images in the directory of reduced files will be considered flat frames by this widget. It can be a dictionary with whatever keywords you want.

The `group_by` argument on line 4 sets the names of the FITS keywords that will be used to group the flat frames. The setting in this example makes sense for dome flats. For twilight flats you presumably want to group only by filter. This setting can also be modified in the widget.

The `file_name_base` argument on line 3 determines part of the output file name for the combined flats. One flat is produced for each unique combination and the file names generated include the values of the keywords used to group them. For the sample data set that comes with `reducer`, these files are produced:

```
master_flat_filter_B_exposure_120.0.fit
master_flat_filter_I_exposure_5.0.fit
master_flat_filter_R_exposure_15.0.fit
master_flat_filter_V_exposure_30.0.fit
```

It could also be used to combine science images in the unlikely case that you wanted to simply average the images without aligning them.

2.1.4 Short video walkthrough

The YouTube video below has a walkthrough of the reducer notebook. No explanatory audio or text, but it goes through the entire reduction process.

2.2 Using with reducer with students

Disclaimer: My experience using reducer is with undergraduates, primarily physics majors. It has been used by a couple of non-majors with no issues.

The first hurdle is installation and a really fast intro to the terminal on the platform of their choice, since you need to launch a jupyter (nee ipython) notebook from the terminal.

For installation and set up I point them to a set of notes about the mechanics of getting started.

I then have them walk through the notebook using the small dataset included with the notebook. This is data that I've reduced by two in each image dimension and converted from 16 bit to 8 bit. It is fine for learning how to use the notebook, but the reduced data will look awful because I was not very careful about converting the calibration images.

If you want the same data set, but at original resolution and 16-bit, download it here (WARNING: 1.5GB). It is images of part of the Landolt field SA112 SF1, taken over one night in July 2013 at the Feder Observatory. It covers a reasonably wide range of airmass, so can be used as an example calculating atmospheric extinction and for determining the transformation to the standard magnitude system. The images contain WCS information, so it shouldn't be too hard to identify the Landolt stars.

2.3 GUI API

2.3.1 reducer.gui Module

Functions

set_color_for(a_widget)	

set_color_for

reducer.gui.**set_color_for**(*a_widget*)

Classes

ToggleContainer(*args, **kwd)	A widget whose state controls the visibility of its chilren.
ToggleMinMax(*args, **kwd)	Widget for setting a minimum and maximum integer value, controlled by a toggle.
ToggleGo(*args, **kwd)	ToggleContainer whose state is linked to a button.

ToggleContainer

class reducer.gui.**ToggleContainer**(**args*, ***kwd*)

Bases: ipywidgets.widgets.widget_box.FlexBox

A widget whose state controls the visibility of its chilren.

Parameters **Same as parameters for a '~IPython.html.widgets.Box', but note that the description of the ToggleContainer is used to set the description of the checkbox that controls the display, AND**

toggle_type : {'checkbox', 'button'}, optional

Specify the type of boolean widget used to toggle the display

Notes

Do *NOT* set the children of the ToggleContainer; set the children of `ToggleContainer.children` or use the *add_child* method.

Attributes

container	Widget that contains the elements controlled by the toggle.
toggle	Toggle widget that controls other display elements.
disabled	True if widget is disabled.

Methods

action()	Subclasses should override this method if they wish to associate an action with
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

Continued on

Table 2.4 – continued from previous page

update_config(config)	Update config and load the new values

Attributes Summary

container	Widget that contains the elements controlled by the toggle.
disabled	True if widget is disabled.
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensible.
toggle	Toggle widget that controls other display elements.

Methods Summary

action()	Subclasses should override this method if they wish to associate an action with the widget.
add_child(child)	Append a child to the container part of the widget.
display()	Display and format this widget.
format()	Format widget.

Attributes Documentation

container
Widget that contains the elements controlled by the toggle.

disabled
True if widget is disabled.

is_sane
Subclasses can define a method that indicates whether the current combination of settings is sensible.

Returns **sanity** : bool or None

True if the settings are sensible, False if not, None if not overridden.

toggle
Toggle widget that controls other display elements.

Methods Documentation

action()
Subclasses should override this method if they wish to associate an action with the widget.

add_child(*child*)
Append a child to the container part of the widget.

Parameters **child** : IPython widget

display()
Display and format this widget.

format()
Format widget.

Must be called after the widget is displayed, and is automatically called by the *display* method.

ToggleMinMax

class reducer.gui.**ToggleMinMax**(**args*, ***kwd*)

Bases: *reducer.gui.ToggleContainer*

Widget for setting a minimum and maximum integer value, controlled by a toggle.

Parameters **description** : str

Text to be displayed in the toggle.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
max	Maximum value in the widget.
min	Minimum value in the widget.
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	Subclasses should override this method if they wish to associate an action with
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.

Continued on

Table 2.8 – continued from previous page

notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

max	Maximum value in the widget.
min	Minimum value in the widget.

Methods Summary

format()

Attributes Documentation

max
Maximum value in the widget.

min
Minimum value in the widget.

Methods Documentation

format()

ToggleGo

class reducer.gui.**ToggleGo**(**args*, ***kwd*)
Bases: *reducer.gui.ToggleContainer*

ToggleContainer whose state is linked to a button.

The intent is for that button to be activated when the contents of the container are in a "sane" state.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	
margin	
model_id	Gets the model id of this widget.
padding	
progress_bar	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	The default action is to invoke the action of each child with an update of the pr
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format the widget; must be invoked after displaying the widget.
get_state([key])	Gets the widget state, or a piece of it.
go()	Returns the action to be taken when the "Go" button is clicked.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.

Table 2.12 – continued from previous page

open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
state_change_handler()	Ties sanity state to go button controls and others
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unlock()	Handler for the unlock button.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

is_sane
progress_bar

Methods Summary

action()	The default action is to invoke the action of each child with an update of the progress bar along the
format()	Format the widget; must be invoked after displaying the widget.
go()	Returns the action to be taken when the "Go" button is clicked.
state_change_handler()	Ties sanity state to go button controls and others
unlock()	Handler for the unlock button.

Attributes Documentation

is_sane

progress_bar

Methods Documentation

action()
The default action is to invoke the action of each child with an update of the progress bar along the way.

format()
Format the widget; must be invoked after displaying the widget.

go()
Returns the action to be taken when the "Go" button is clicked.

state_change_handler()
Ties sanity state to go button controls and others

unlock()
Handler for the unlock button.

Class Inheritance Diagram

2.4 image browser API

2.4.1 reducer.image_browser Module

Functions

ndarray_to_png(x[, min_percent, max_percent])	

ndarray_to_png

reducer.image_browser.**ndarray_to_png**(*x*, *min_percent=20*, *max_percent=99.5*)

Classes

ImageTree(tree)	Create a tree view of a collection of images.
FitsViewer()	Display the image and header from a single FITS file.
ImageBrowser(collection[, allow_missing])	Browse a tree of FITS images and view image/header.

ImageTree

class reducer.image_browser.**ImageTree**(*tree*)
Bases: object

Create a tree view of a collection of images.

Parameters **tree** : *msumastro.TableTree*

Tree of images, arranged by metadata.

Attributes

top	Widget at the top of the tree.

Methods

display()	Display and format this widget.
format()	This gets called by the ImageBrowser so don't delete it.

Attributes Summary

top	Widget at the top of the tree.

Methods Summary

display()	Display and format this widget.
format()	This gets called by the ImageBrowser so don't delete it.

Attributes Documentation

top
Widget at the top of the tree.

Methods Documentation

display()
Display and format this widget.

format()
This gets called by the ImageBrowser so don't delete it.

For now it also closes all of the tabs after the browser is created because doing it before (at least ipywidgets 5.1.5 and lower) causes a javascript error which prevents properly setting the titles.

FitsViewer

class reducer.image_browser.**FitsViewer**
Bases: object

Display the image and header from a single FITS file.

Attributes

top	

Methods

display()	Display and format this widget.

Continued

Table 2.22 – continued from previous page

format()	Format widget.
set_fits_file_callback([demo, image_dir])	Returns a callback function that sets the name of FITS file to display and upda

Attributes Summary

top

Methods Summary

display()	Display and format this widget.
format()	Format widget.
set_fits_file_callback([demo, image_dir])	Returns a callback function that sets the name of FITS file to display and upda

Attributes Documentation

top

Methods Documentation

display()
Display and format this widget.

format()
Format widget.

Must be called after the widget is displayed, and is automatically called by the *display* method.

set_fits_file_callback(*demo=True*, *image_dir=None*)
Returns a callback function that sets the name of FITS file to display and updates the widget.

The callback takes one argument, the name of the fits file, or 'demo' to enable the display of a couple of sample images.

ImageBrowser

class reducer.image_browser.**ImageBrowser**(*collection*, *allow_missing=True*, **args*, ***kwd*)
Bases: ipywidgets.widgets.widget_box.FlexBox

Browse a tree of FITS images and view image/header.

Parameters **collection** : *ccdproc.ImageFileCollection*

Directory of images.

Attributes

border	
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.

Continued on next page

Table 2.25 – continued from previous page

fits_display	Widget that displays FITS image/header.
height	
margin	
model_id	Gets the model id of this widget.
padding	
tree_widget	Widget that represents the image tree.
width	

Methods

add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.

Continued on next page

Table 2.26 – continued from previous page

update_config(config)	Update config and load the new values

Attributes Summary

fits_display	Widget that displays FITS image/header.
tree_widget	Widget that represents the image tree.

Methods Summary

display()	Display and format this widget.
format()	Format widget.

Attributes Documentation

fits_display
Widget that displays FITS image/header.

tree_widget
Widget that represents the image tree.

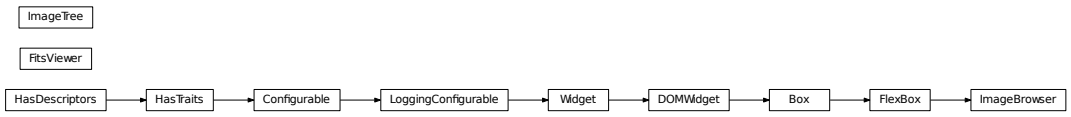
Methods Documentation

display()
Display and format this widget.

format()
Format widget.

Must be called after the widget is displayed, and is automatically called by the *display* method.

Class Inheritance Diagram



2.5 astro_gui API

2.5.1 reducer.astro_gui Module

Classes

Reduction(*arg, **kwd)	Primary widget for performing a logical reduction step (e.g.
Combiner(*args, **kwd)	Widget for displaying options for ccdproc.Combiner.
CosmicRaySettings(*args, **kwd)	**Attributes**
Slice(*arg, **kwd)	**Attributes**
CalibrationStep(*args, **kwd)	Represents a calibration step that corresponds to a ccdproc command.
BiasSubtract([bias_image])	Subtract bias from an image using widget settings.
DarkSubtract([bias_image])	Subtract dark from an image using widget settings.
FlatCorrect([bias_image])	Subtract dark from an image using widget settings.
Overscan(*arg, **kwd)	docstring for Overscan
Trim(*arg, **kwd)	Controls and action for trimming a widget.

Reduction

class reducer.astro_gui.**Reduction**(**arg*, ***kwd*)

Bases: reducer.astro_gui.ReducerBase

Primary widget for performing a logical reduction step (e.g. dark subtraction or flat correction).

Attributes

apply_to	
border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
destination	
disabled	True if widget is disabled.
height	
is_sane	
margin	
model_id	Gets the model id of this widget.
padding	
progress_bar	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.

Continued on next page

Table 2.31 – continued from previous page

class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format the widget; must be invoked after displaying the widget.
get_state([key])	Gets the widget state, or a piece of it.
go()	Returns the action to be taken when the "Go" button is clicked.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
state_change_handler()	Ties sanity state to go button controls and others
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unlock()	Handler for the unlock button.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

action()

Methods Documentation

action()

Combiner

class reducer.astro_gui.**Combiner**(**args*, ***kwd*)

Bases: reducer.astro_gui.ReducerBase

Widget for displaying options for ccdproc.Combiner.

Parameters **description** : str, optional

Text displayed next to check box for selecting options.

Attributes

apply_to	
border	
combined	The combined image.
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
destination	
disabled	True if widget is disabled.
height	
image_source	
is_sane	
margin	
model_id	Gets the model id of this widget.
padding	
progress_bar	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	
get_state([key])	Gets the widget state, or a piece of it.
go()	Returns the action to be taken when the "Go" button is clicked.

Continued on next page

Table 2.34 – continued from previous page

handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
state_change_handler()	Ties sanity state to go button controls and others
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unlock()	Handler for the unlock button.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

combined	The combined image.
image_source	
is_sane	

Methods Summary

action()
format()

Attributes Documentation

combined
The combined image.

image_source

is_sane

Methods Documentation

action()

format()

CosmicRaySettings

class reducer.astro_gui.**CosmicRaySettings**(**args*, ***kwd*)
Bases: *reducer.gui.ToggleContainer*

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	Subclasses should override this method if they wish to associate an action with
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.

Continued on

Table 2.38 – continued from previous page

notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

display()

Methods Documentation

display()

Slice

class reducer.astro_gui.**Slice**(**arg*, ***kwd*)
Bases: *reducer.gui.ToggleContainer*

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Determine whether combination of settings is at least remotely plausible.
margin	
model_id	Gets the model id of this widget.
padding	

Continued on next page

Table 2.40 – continued from previous page

toggle	Toggle widget that controls other display elements.
width	

Methods

action()	Subclasses should override this method if they wish to associate an action with
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

is_sane	Determine whether combination of settings is at least remotely plausible.

Methods Summary

format()

Attributes Documentation

is_sane
Determine whether combination of settings is at least remotely plausible.

Methods Documentation

format()

CalibrationStep

class reducer.astro_gui.**CalibrationStep**(**args*, ***kwd*)
Bases: *reducer.gui.ToggleContainer*

Represents a calibration step that corresponds to a ccdproc command.

Parameters None

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
match_on	List of keywords whose values should match in the image being calibated and the calibration image
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action()	Subclasses should override this method if they wish to associate an action with
add_child(child)	Append a child to the container part of the widget.

Continued on

Table 2.45 – continued from previous page

add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

match_on	List of keywords whose values should match in the image being calibated and the calibration image.

Attributes Documentation

match_on

List of keywords whose values should match in the image being calibated and the calibration image.

BiasSubtract

class reducer.astro_gui.**BiasSubtract**(*bias_image=None*, ***kwd*)
Bases: *reducer.astro_gui.CalibrationStep*

Subtract bias from an image using widget settings.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
match_on	List of keywords whose values should match in the image being calibated and the calibration image
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action(ccd)	
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.

Continued on next page

Table 2.48 – continued from previous page

on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

action(ccd)

Methods Documentation

action(*ccd*)

DarkSubtract

class reducer.astro_gui.**DarkSubtract**(*bias_image=None*, ***kwd*)

Bases: *reducer.astro_gui.CalibrationStep*

Subtract dark from an image using widget settings.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
match_on	List of keywords whose values should match in the image being calibated and the calibration image
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action(ccd)	
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

action(ccd)

Methods Documentation

action(*ccd*)

FlatCorrect

class reducer.astro_gui.**FlatCorrect**(*bias_image=None*, ***kwd*)
Bases: *reducer.astro_gui.CalibrationStep*

Subtract dark from an image using widget settings.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Subclasses can define a method that indicates whether the current combination of settings is sensibl
margin	
match_on	List of keywords whose values should match in the image being calibated and the calibration image
model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action(ccd)	
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	Format widget.
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits

Continued on next page

Table 2.54 – continued from previous page

hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

action(ccd)

Methods Documentation

action(*ccd*)

Overscan

class reducer.astro_gui.**Overscan**(**arg*, ***kwd*)

Bases: *reducer.astro_gui.Slice*

docstring for Overscan

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	

Continued on next page

Table 2.56 – continued from previous page

margin	
model_id	Gets the model id of this widget.
padding	
polynomial_order	
toggle	Toggle widget that controls other display elements.
width	

Methods

action(ccd)	Subtract overscan from image based on settings.
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.

Continued on next page

Table 2.57 – continued from previous page

unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Attributes Summary

is_sane
polynomial_order

Methods Summary

action(ccd)	Subtract overscan from image based on settings.
format()	

Attributes Documentation

is_sane

polynomial_order

Methods Documentation

action(*ccd*)
Subtract overscan from image based on settings.

Parameters **ccd** : *ccdproc.CCDData*

Image to be reduced.

format()

Trim

class reducer.astro_gui.**Trim**(**arg*, ***kwd*)
Bases: *reducer.astro_gui.Slice*

Controls and action for trimming a widget.

Attributes

border	
container	Widget that contains the elements controlled by the toggle.
cross_validation_lock	A contextmanager for running a block with our cross validation lock set to True.
disabled	True if widget is disabled.
height	
is_sane	Determine whether combination of settings is at least remotely plausible.
margin	

Continued on next page

Table 2.60 – continued from previous page

model_id	Gets the model id of this widget.
padding	
toggle	Toggle widget that controls other display elements.
width	

Methods

action(ccd)	Trim an image to bounds given in the widget.
add_child(child)	Append a child to the container part of the widget.
add_class(className)	Adds a class to the top level element of the widget.
add_traits(**traits)	Dynamically add trait attributes to the Widget.
class_config_rst_doc()	Generate rST documentation for this class' config options.
class_config_section()	Get the config class config section
class_get_help([inst])	Get the help string for this class in ReST format.
class_get_trait_help(trait[, inst])	Get the help string for a single trait.
class_own_trait_events(name)	Get a dict of all event handlers defined on this class, not a parent.
class_own_traits(**metadata)	Get a dict of all the traitlets defined on this class, not a parent.
class_print_help([inst])	Get the help string for a single trait and print it.
class_trait_names(**metadata)	Get a list of all the names of this class' traits.
class_traits(**metadata)	Get a dict of all the traits of this class.
close()	Close method.
display()	Display and format this widget.
format()	
get_state([key])	Gets the widget state, or a piece of it.
handle_comm_opened(comm, msg)	Static method, called when a widget is constructed.
has_trait(name)	Returns True if the object has a trait with the specified name.
hold_sync(*args, **kwds)	Hold syncing any state until the outermost context manager exits
hold_trait_notifications(*args, **kwds)	Context manager for bundling trait change notifications and cross validation.
notify_change(change)	Called when a property has changed.
observe(handler[, names, type])	Setup a handler to be called when a trait changes.
on_displayed(callback[, remove])	(Un)Register a widget displayed callback.
on_msg(callback[, remove])	(Un)Register a custom msg receive callback.
on_trait_change([handler, name, remove])	DEPRECATED: Setup a handler to be called when a trait changes.
on_widget_constructed(callback)	Registers a callback to be called when a widget is constructed.
open()	Open a comm to the frontend if one isn't already open.
remove_class(className)	Removes a class from the top level element of the widget.
section_names()	return section names as a list
send(content[, buffers])	Sends a custom msg to the widget model in the front-end.
send_state([key])	Sends the widget state, or a piece of it, to the front-end.
set_state(sync_data)	Called when a state is received from the front-end.
set_trait(name, value)	Forcibly sets trait attribute, including read-only attributes.
setup_instance(*args, **kwargs)	
trait_events([name])	Get a dict of all the event handlers of this class.
trait_metadata(traitname, key[, default])	Get metadata values for trait by key.
trait_names(**metadata)	Get a list of all the names of this class' traits.
traits(**metadata)	Get a dict of all the traits of this class.
unobserve(handler[, names, type])	Remove a trait change handler.
unobserve_all([name])	Remove trait change handlers of any type for the specified name.
update_config(config)	Update config and load the new values

Methods Summary

action(ccd)	Trim an image to bounds given in the widget.

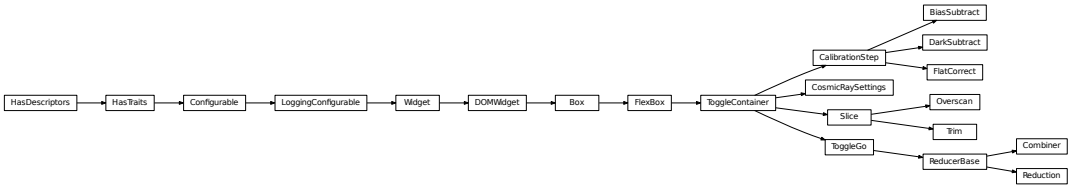
Methods Documentation

action (*ccd*)
Trim an image to bounds given in the widget.

Returns **trimmed** : *ccdproc.CCDData*

Trimmed image.

Class Inheritance Diagram



2.6 Contributors

2.6.1 Project Coordinator

- Matt Craig (@mwcraig)

2.6.2 Contributors

All package contributors, listed alphabetically.

- Juan Cabanela (@JuanCab)
- Nathan Heidt (@heidtna)
- Stuart Littlefair (@StuartLittlefair)
- Thomas Robitaille (@astrofrog)
- Christian Tismer (@ctismer)

2.7 Changes

2.7.1 1.0.0 (unreleased)

General

New Features

Other Changes

Bug fixes

2.7.2 0.3.0 (2016-07-17)

General

- This version only supports IPython 4 or higher, and requires `ipywidgets` version 4.
- The minimum required version of ccdproc is now 1.0.

New Features

- Images can now simply be copied from the source to the destination directory. [#137]

Other Changes

Bug fixes

2.7.3 0.2.9 (2016-06-16)

General

New Features

Other Changes

- Update package requirements to ipywidgets instead of ipython, and restrict version number.

Bug fixes

- Use numpy dtype name instead of dtype itself to determine output dtype. [#129]

2.7.4 0.2.8 (2016-05-31)

General

New Features

Other Changes

Bug fixes

- Check that the image collection for master images exists before refreshing it. [#128]

2.7.5 0.2.7 (2016-05-30)

General

New Features

Other Changes

Bug fixes

- The *ImageFileCollection* used to find masters was out of date and not refreshed if a reduction widget was created before the masters were created. [#127]

2.7.6 0.2.6 (2016-05-27)

General

New Features

Other Changes

- Use combine function for combining images to limit memory usage during image combination. [#120, #121]
- Use `median` and `median_absolute_deviation` in sigma clipping instead of the default `mean` and `std`. [#106]
- Discard mask/uncertainty from result of image combination unless input images have mask/uncertainty. [#119]
- Choose sensible data type for reduced images based on data type of original images. [#122]

Bug fixes

- Eliminate huge memory usage by reduction. [#118]

2.7.7 0.2.5 (2016-05-25)

General

New Features

Other Changes

- Improve display of images in file browser.

Bug fixes

- Work around a bug in ccdproc/astropy.nddata that incorrectly creates an uncertainty as a mask.
- Work around a bug in astropy.io.fits that results in writing incorrect data values in some cases.

2.7.8 0.2.3 (2016-05-23)

General

New Features

Other Changes

Bug fixes

- Ensure unsigned int images can be displayed. [#115, #116]
- Ensure that combined images can be written. [#117]

r

A

B

C

D

F

G

I

M

N

O

P

R

S

T

U