GeoIPS data_fusion Documentation

Release 1.11.1.post4

U.S. NAVAL RESEARCH LABORATORY

Aug 21, 2023

CONTENTS

Date: Aug 21, 2023 **Version**: 1.11.1.post4

Download PDF documentation: `GeoIPS_data_fusion.pdf`

Previous versions: Documentation of previous data_fusion versions are available at github.com/NRLMMD-GEOIPS.

Useful links: Source Repository | GeoIPS License | NRLMMD |

`data_fusion` is a free software program, United States Governmnet NRLMMD licensed.

```
Distribution Statement A. Approved for public release. Distribution␣
↪unlimited.

Author:
Naval Research Laboratory, Marine Meteorology Division

This program is free software: you can redistribute it and/or modify it␣
↪under
the terms of the NRLMMD License included with this program. This program␣
↪is
distributed WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the included␣
↪license
for more details. If you did not receive the license, for more␣
↪information see:
https://github.com/U-S-NRL-Marine-Meteorology-Division/
```

The GeoIPS ® data_fusion Package provides a Python 3 plugin to GeoIPS.

The data_fusion plugin provides the capability for including an arbitrary number of data types within a single product or algorithm.

User Guide

The user guide provides in-depth information on the key concepts of data_fusion with useful background information and explanation.

User Guide

The API reference guide

The reference guide contains a detailed description of data_fusion API. The reference describes how the methods work and which parameters can be used. It assumes that you have an understanding of the key concepts.

API

To the release notes

Change logs, versioning and contribution history.

Release Notes

CHAPTER ONE

INTRODUCTION

CHAPTER TWO

USER GUIDE

CHAPTER

THREE

API REFERENCE

3.1 data_fusion package

3.1.1 Subpackages

data_fusion.commandline package

Submodules

data_fusion.commandline.args module

Command line script for kicking off geoips based procflows.

`data_fusion.commandline.args.`**`add_args`**`(`*parser*, *arglist=None*`)`

List of available standard arguments for calling data file processing command line.

Parameters

- **`parser`** (*`ArgumentParser`*) –
  - argparse ArgumentParser to add appropriate arguments
- **`arglist`** (*`list, default=None`*) –
  - **list of requested arguments to add to the ArgumentParser**
    - if None, include all arguments

Return type

No return values (parser modified in place)

`data_fusion.commandline.args.`**`check_command_line_args`**`(`*arglist*, *argdict*`)`

Check formatting of command line arguments.

Parameters

- **`arglist`** (*`list`*) –

  - List of desired command line arguments to check within argdict for appropriate formatting
- `argdict` (*dict*) –
  - Dictionary of command line arguments

Returns

- Return True if all arguments are of appropriate formatting.

Return type

bool

Raises

`TypeError` –

- Incorrect command line formatting

`data_fusion.commandline.args.`**`get_command_line_args`**(*arglist=None*, *description=None*)

Parse command line arguments specified by the requested list of arguments.

Parameters

- `list` (*arglist*) –
  - **list of requested arguments to add to the ArgumentParser**
    - if None, include all arguments
- `default=None.` –
  - **list of requested arguments to add to the ArgumentParser**
    - if None, include all arguments
- `str` (*description*) –
  - String description of arguments
- `default=None` –
  - String description of arguments

Returns

- Dictionary of command line arguments

Return type

dict

data_fusion.commandline.data_fusion_procflow module

Command line script for kicking off geoips based procflows.

MUST call with –procflow

`data_fusion.commandline.data_fusion_procflow.main()`

Script to kick off processing based on command line args.

Module contents

data_fusion.commandline init file.

data_fusion.plugins package

Subpackages

data_fusion.plugins.modules package

Subpackages

data_fusion.plugins.modules.algorithms package

Submodules

data_fusion.plugins.modules.algorithms.stitched module

Data manipulation steps for "stitched" algorithm.

`data_fusion.plugins.modules.algorithms.stitched.call`(*xarray_dict*, *parallax_correction=True*, *satzen_correction=True*)

Algorithm for stitching multiple datasets into a single combined product.

Parameters

`xobj` (*`xarray.dataset`*) –

- list of numpy.ndarray or numpy.MaskedArray of channel data, in order of sensor "channels" list
- Degrees Kelvin

Returns

- **dstacked numpy.ndarrays or numpy.MaskedArrays containing:**
  - np.ma.dstack((pcb_mask, mod_mask, bt110, bt110_night)).squeeze()

Return type

numpy.ndarray

Module contents

data_fusion.plugins.modules.algorithms init file.

data_fusion.plugins.modules.output_formatters package

Submodules

data_fusion.plugins.modules.output_formatters.layered_imagery module

Layered imagery output formatter module.

`data_fusion.plugins.modules.output_formatters.layered_imagery.`**`call`**(*xarray_dict*, *area_def*, *product_name*, *output_fnames*, *boundaries_info=None*, *gridlines_info=None*, *title_format=None*, *title_copyright=None*)

Plot the fused datasets.

`data_fusion.plugins.modules.output_formatters.layered_imagery.`**`create_all_colorbars`**(*fig*, *mai* *map* *xarray_*

Create colorbars for each product.

data_fusion.plugins.modules.output_formatters.layered_imagery.**get_final_mpl_colors_inf**

Get final mpl_colors_info dictionary.

data_fusion.plugins.modules.output_formatters.layered_imagery.**layered_title**(*area_def*, *xrdict*, *include_end_da* *dataset_dict=*

Create title for the fused data output.

Module contents

data_fusion.plugins.modules.output_formatters init file.

data_fusion.plugins.modules.procflows package

Submodules

data_fusion.plugins.modules.procflows.data_fusion module

Driver for standard single channel products.

data_fusion.plugins.modules.procflows.data_fusion.**call**(*fnames*, *command_line_args=None*)

Workflow for running multiple datatypes in a single call.

Parameters

- **fnames** (*list of strings*) –
  - List of strings specifying full paths to input file names to process
- **(dict)** (*command_line_args*) –
  - **dictionary of command line arguments**
    - **'reader_name': Explicitly request reader**
      - geoips*.readers.readername.readername
    - **Optional: 'sector_list': list of YAML sectorfiles**

  - tc<YYYY><BASIN><NUM><NAME> for TCs, ie tc2020sh16gabekile If sectorfiles and sectorlist not included, looks in database

Returns

- Return list of strings specifying full paths to output products that were produced.

Return type

list

`data_fusion.plugins.modules.procflows.data_fusion.`**`get_fused_xarray`**`(`*area_def*, *fuse_data*`)`

Get the fused xarray.

This loops through each “fuse” dataset, and calls single_source.get_alg_xarray to pre-process each appropriately. After pre-processing each dataset to their individual “products”, the final algorithm is applied to all datasets.

`data_fusion.plugins.modules.procflows.data_fusion.`**`get_overall_end_datetime`**`(`*fuse_dict*`)`

Get the ending datetime for all products.

`data_fusion.plugins.modules.procflows.data_fusion.`**`get_overall_start_datetime`**`(`*fuse_dict*`)`

Get the starting datetime for all products.

`data_fusion.plugins.modules.procflows.data_fusion.`**`run_fuse_alg`**`(`*fuse_xarrays*, *fuse_prod_plugin*, *fuse_source_name*`)`

Run the fusion algorithm.

`data_fusion.plugins.modules.procflows.data_fusion.`**`unpack_fusion_arguments`**`(`*argdict*`)`

Unpack fusion arguments.

Module contents

data_fusion.interface_modules.procflows init file.

Module contents

data_fusion.plugins.modules init file.

Module contents

data_fusion.plugins init file.

3.1.2 Module contents

The GeoIPS ® data_fusion Package provides a Python 3 plugin to GeoIPS.

The data_fusion plugin provides the capability for including an arbitrary number of data types within a single product or algorithm.

CHAPTER

FOUR

CONTACT

contact geoips@nrlmry.navy.mil

4.1 About Us

contact geoips@nrlmry.navy.mil

PYTHON MODULE INDEX

INDEX