MTH5: A DATA FORMAT FOR MAGNETOTELLURIC DATA

Jared Peacock, US Geological Survey

Andy Frassetto, IRIS

Karl Kappler, *IMDEX Technology USA*

Anna Kelbert, US Geological Survey

Tim Ronan, *IRIS*

Lindsey Heagy, *University of British Columbia*

Peacock et al., (2022) MTH5: An archive and exchangeable data format for magnetotelluric time series data, Computers & Geoscience, 162, https://doi.org/10.1016/j.cageo.2022.105102

USEFUL LINKS

Metadata Standards

- **Table of standards:** Peacock et al. (2021) http://dx.doi.org/10.5066/P9AXGKEV

MT-Metadata Open-Source Python Package

- **Base code:** mt_metadata
- **Documentation**: Welcome to MT Metadata documentation! — MT Metadata 0.1.8 documentation (mt-metadata.readthedocs.io)
- **Zero-install Jupyter Notebooks**: mt_metadata binder

MTH5 Open-Source Python Package

- **Base code:** mth5
- **Documentation**: Welcome to MTH5's documentation! — MTH5 0.2.5 documentation
- **Zero-install Jupyter Notebooks**: MTH5 Binder

Peacock et al., (2022) MTH5: An archive and exchangeable data format for magnetotelluric time series data, Computers & Geoscience, 162, https://doi.org/10.1016/j.cageo.2022.105102

MOTIVATION FOR STANDARDS

1. Funding agencies, government entities, research institutions are requiring data be
   - **F**indable (searchable metadata)
   - **A**ccessible (publicly available)
   - **I**nteroperable (any OS, various software)
   - **R**eproducible

   See Wilkinson et al. (2016) https://doi.org/10.1038/sdata.2016.18
2. Outside interest in MT data
   - Kelbert et al. (2018) https://doi.org/10.1029/2018EO112859
3. IRIS PASSCAL are adding MT instruments

BUILDING BLOCKS

Metadata

- A set of data that describes and gives information about other data
- For efficient reading standard descriptions of specific data are needed.

→ Common Keywords

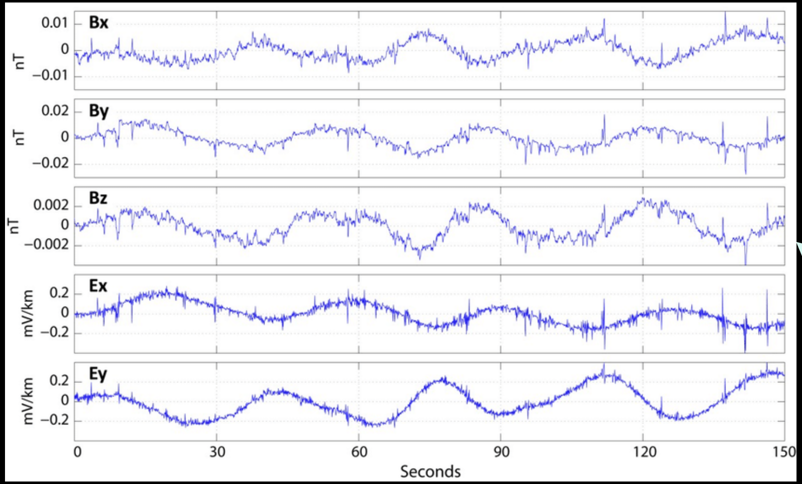
Data

- Values recorded
- Formatted and structured in a standard way to comply with FAIR principles
- Archive vs Working formats

→ Standard format

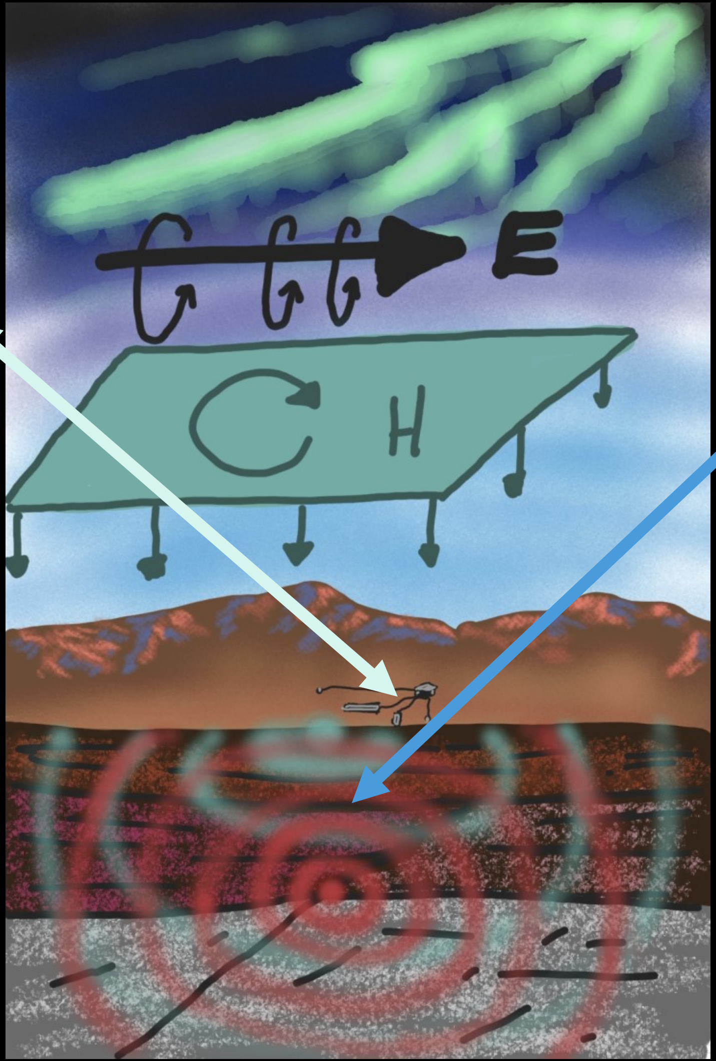
MT DATA TYPES

Time Series

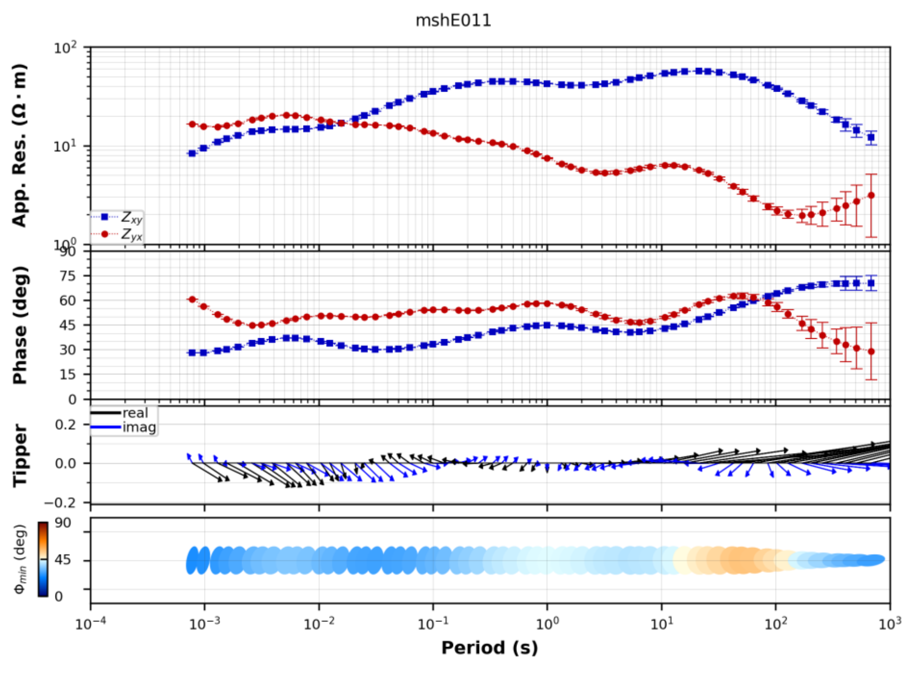


Formats:
Traditionally, whatever comes out of the data logger

- Phoenix
- Metronix
- Zonge
- NIMS
- LEMI
- EDL
- Homebrew



Transfer Functions



Formats:

- EDI
- EMTF XML
- Z-files
- J-files
- AVG files
- DAT files
- Others

PREVIOUS WORK ON STANDARDIZATION

Time Series

- EDI [fun fact]
- Suggested standards by Kirkby (2019) https://doi.org/10.1080/14432471.2019.1600210
- Geoscience Australia suggests MTH5 moving forward
  - Duan et al. (2021) https://doi.org/10.1080/14432471.2021.2012035

Transfer Functions

- EDI - Wright (1988) https://doi.org/10.1190/1.1892244
- EMTF XML - Kelbert (2020) https://doi.org/10.1190/geo2018-0679.1
  - For archiving see: Kelbert et al. (2011) https://doi.org/10.17611/DP/EMTF.1.
  - For Fortran tools see: Kelbert (2019) https://doi.org/10.5066/P98PPLDV

STANDARDIZING

- In 2019 IRIS established a working group to develop MT time series metadata standards.

- For 2 years standards were developed/argued/debated during monthly meetings

- Follow logically MT surveys, hierarchical

- Proposed standards are in Peacock et al. (2021) http://dx.doi.org/10.5066/P9AXGKEV

Jared Peacock	USGS	Member (Chair)
Bruce Beaudoin	IRIS PASSCAL Instrument Center	Observer
Ninfa Bennington	USGS-HVO	Ex Officio
Lloyd Carothers	IRIS PASSCAL Instrument Center	Observer
Jerry Carter	IRIS DMC	Member
Gary Egbert	Oregon State University	Observer
Andy Frassetto	IRIS	Project Lead
David Goldak	IRIS PASSCAL Instrument Center	Observer
Karl Kappler	DataCloud	Observer
Anna Kelbert	USGS	Member
Maeva Pourpoint	IRIS PASSCAL Instrument Center	Observer
Tim Ronan	IRIS DMC	Observer
Adam Schultz	Oregon State University	Observer
Maxim Smirnov	Luleå University of Technology	Member
Chad Trabant	IRIS DMC	Observer

METADATA STANDARDS

Keywords are defined by

- **Name:** A full descriptive name that is logical for the keyword. Forced to be all lower case and full words separated by _
- **Type:** Base data type [string | float | int | Boolean]
- **Style:** Describes how the string should be formatted.
- **Required:** True if required or False if optional
- **Units:** Physical units of the keyword. Given as full name all lowercase separated by _ and a – for multiplicative units [*ohm-meters*] and per for a ratio of units [*meters per second*]
- **Description:** Detailed description of what the keyword represents
- **Options:** If "style" is "controlled vocabulary" this is a list of accepted options
- **Example:** An example use of the keyword
- **Default:** A default value, only set if Required = True

Example

Name: measurement_azimuth

type: float

style: number

required: True

units: degrees

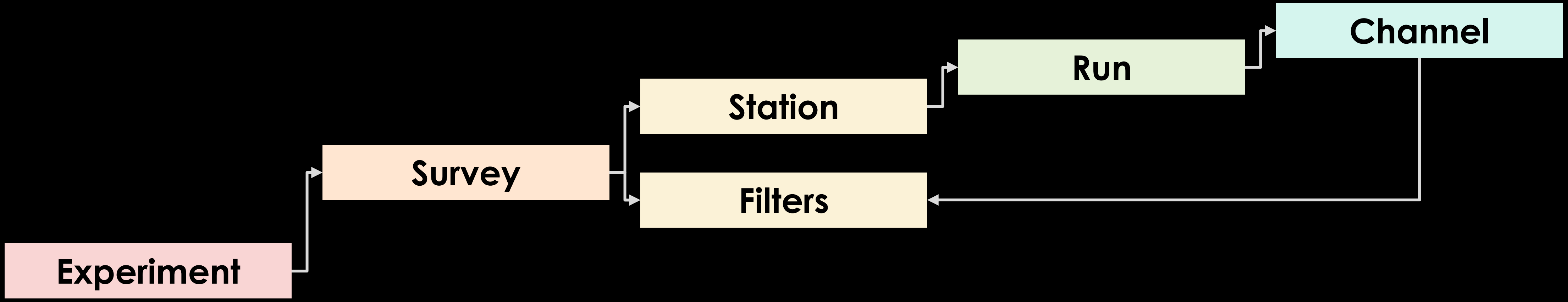
description: Measurement angle in specified coordinate system.

options: []

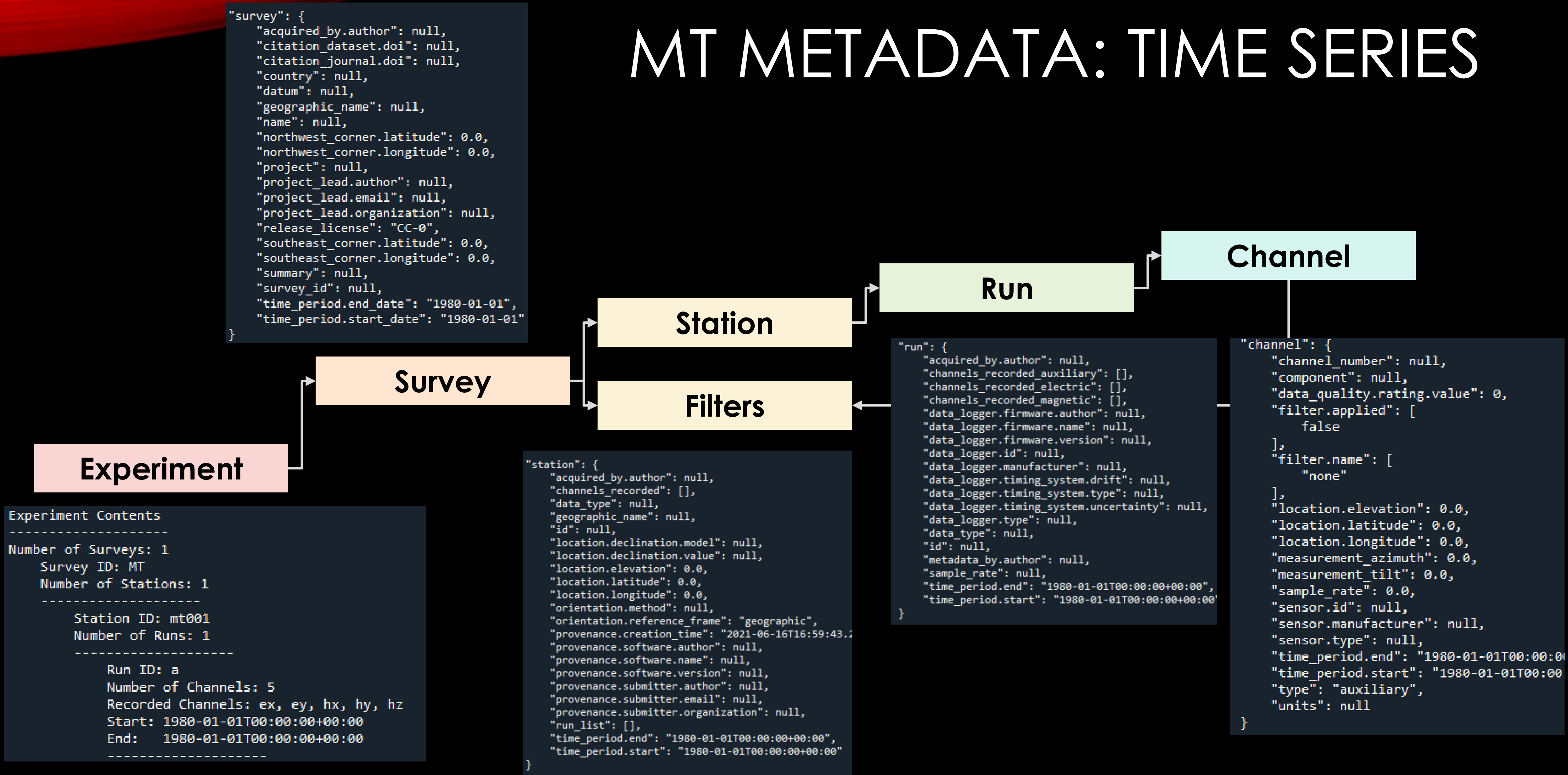
example: 23.134

default: 0.0

MT METADATA: TIME SERIES



MT METADATA: TIME SERIES



```
"survey": {
    "acquired_by.author": null,
    "citation_dataset.doi": null,
    "citation_journal.doi": null,
    "country": null,
    "datum": null,
    "geographic_name": null,
    "name": null,
    "northwest_corner.latitude": 0.0,
    "northwest_corner.longitude": 0.0,
    "project": null,
    "project_lead.author": null,
    "project_lead.email": null,
    "project_lead.organization": null,
    "release_license": "CC-0",
    "southeast_corner.latitude": 0.0,
    "southeast_corner.longitude": 0.0,
    "summary": null,
    "survey_id": null,
    "time_period.end_date": "1980-01-01",
    "time_period.start_date": "1980-01-01"
}
```

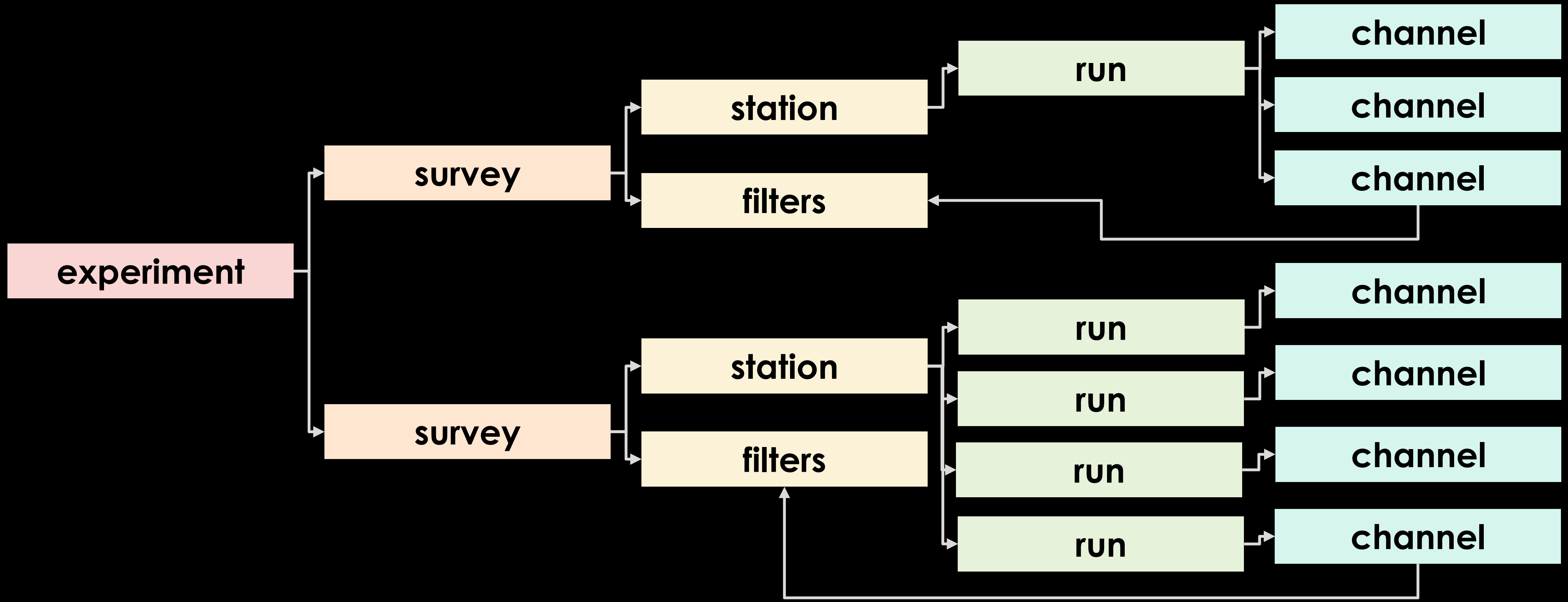
```
Experiment Contents
--------------------
Number of Surveys: 1
    Survey ID: MT
    Number of Stations: 1
    --------------------
        Station ID: mt001
        Number of Runs: 1
        --------------------
            Run ID: a
            Number of Channels: 5
            Recorded Channels: ex, ey, hx, hy, hz
            Start: 1980-01-01T00:00:00+00:00
            End:   1980-01-01T00:00:00+00:00
            --------------------
```

```
"station": {
    "acquired_by.author": null,
    "channels_recorded": [],
    "data_type": null,
    "geographic_name": null,
    "id": null,
    "location.declination.model": null,
    "location.declination.value": null,
    "location.elevation": 0.0,
    "location.latitude": 0.0,
    "location.longitude": 0.0,
    "orientation.method": null,
    "orientation.reference_frame": "geographic",
    "provenance.creation_time": "2021-06-16T16:59:43.2
    "provenance.software.author": null,
    "provenance.software.name": null,
    "provenance.software.version": null,
    "provenance.submitter.author": null,
    "provenance.submitter.email": null,
    "provenance.submitter.organization": null,
    "run_list": [],
    "time_period.end": "1980-01-01T00:00:00+00:00",
    "time_period.start": "1980-01-01T00:00:00+00:00"
}
```

```
"run": {
    "acquired_by.author": null,
    "channels_recorded_auxiliary": [],
    "channels_recorded_electric": [],
    "channels_recorded_magnetic": [],
    "data_logger.firmware.author": null,
    "data_logger.firmware.name": null,
    "data_logger.firmware.version": null,
    "data_logger.id": null,
    "data_logger.manufacturer": null,
    "data_logger.timing_system.drift": null,
    "data_logger.timing_system.type": null,
    "data_logger.timing_system.uncertainty": null,
    "data_logger.type": null,
    "data_type": null,
    "id": null,
    "metadata_by.author": null,
    "sample_rate": null,
    "time_period.end": "1980-01-01T00:00:00+00:00",
    "time_period.start": "1980-01-01T00:00:00+00:00'
}
```

```
"channel": {
    "channel_number": null,
    "component": null,
    "data_quality.rating.value": 0,
    "filter.applied": [
        false
    ],
    "filter.name": [
        "none"
    ],
    "location.elevation": 0.0,
    "location.latitude": 0.0,
    "location.longitude": 0.0,
    "measurement_azimuth": 0.0,
    "measurement_tilt": 0.0,
    "sample_rate": 0.0,
    "sensor.id": null,
    "sensor.manufacturer": null,
    "sensor.type": null,
    "time_period.end": "1980-01-01T00:00:0(
    "time_period.start": "1980-01-01T00:00
    "type": "auxiliary",
    "units": null
}
```

MT METADATA: TIME SERIES



MT_METADATA: OPEN-SOURCE SOFTWARE

GOAL: develop tools to validate standardized MT metadata

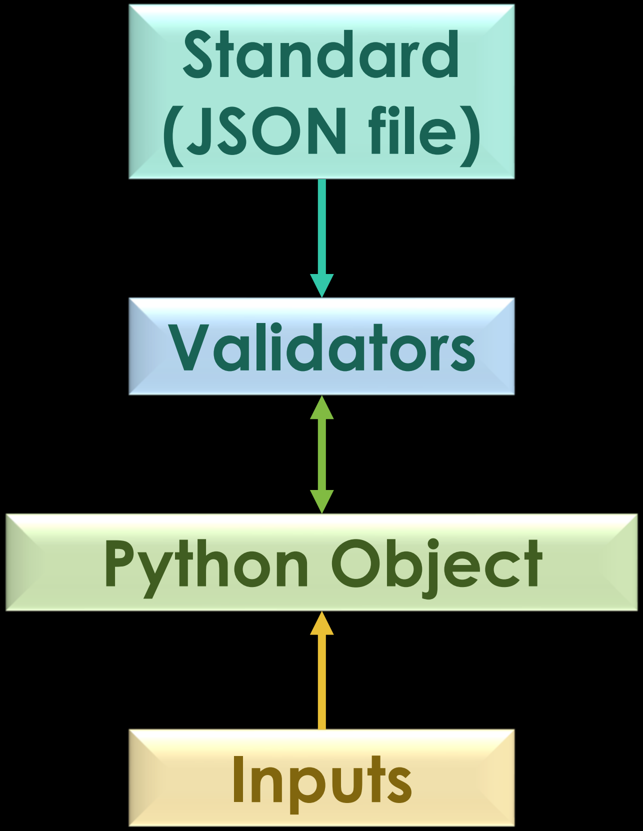
- **Time series** has been developed from a collaborative project between USGS and IRIS (IRIS MT Software Working Group) funded in part by USGS Community for Data Integration

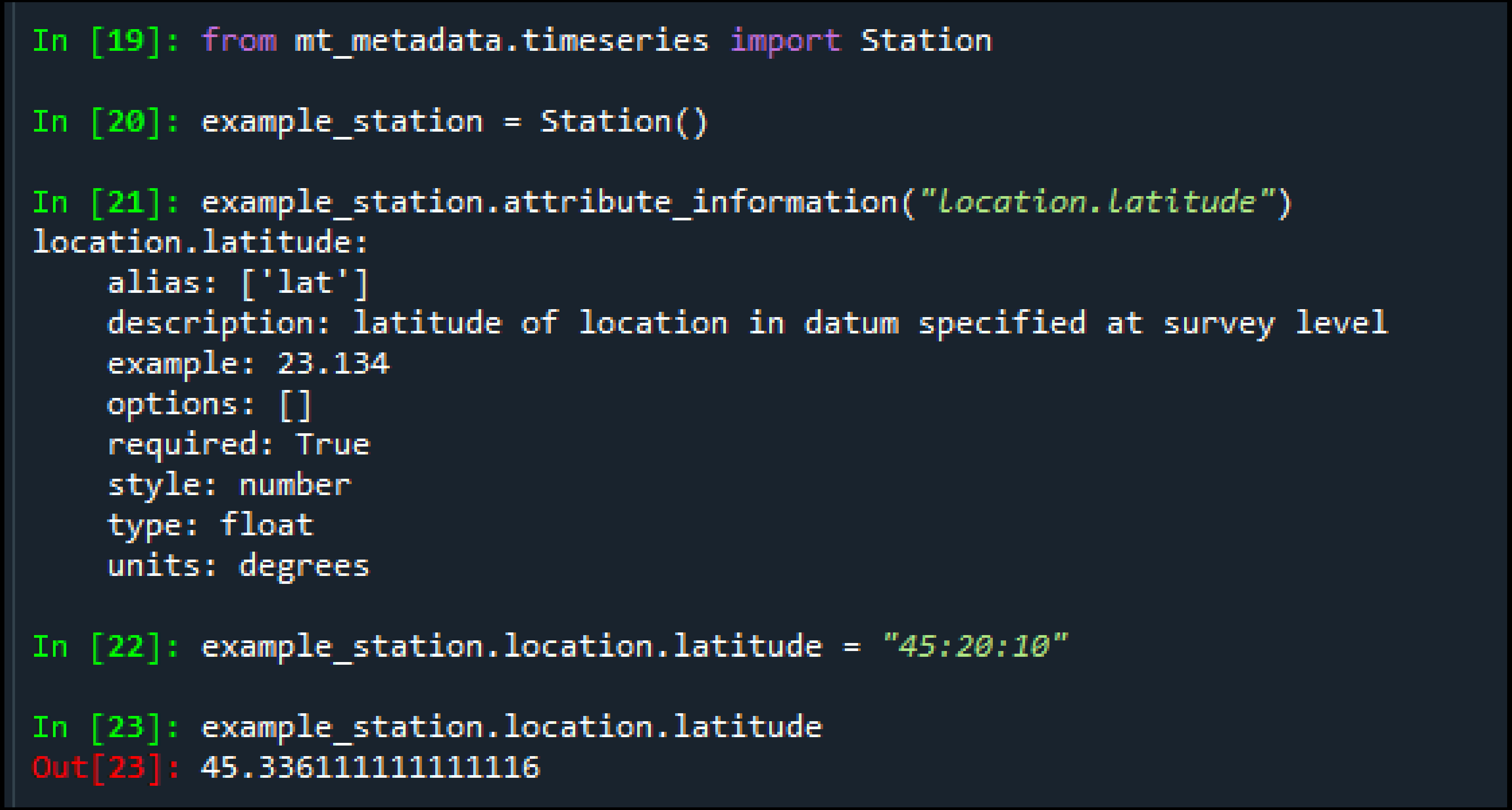
- **Transfer Functions** follows the time series with capability to read/write (EDI, EMTFXML, z-files, j-files, AVG files)

Code repository can be found at: **https://github.com/kujaku11/mt_metadata**

Documentation can be found at: **https://mt-metadata.readthedocs.io/en/latest/**

MT-METADATA OPEN-SOURCE SOFTWARE

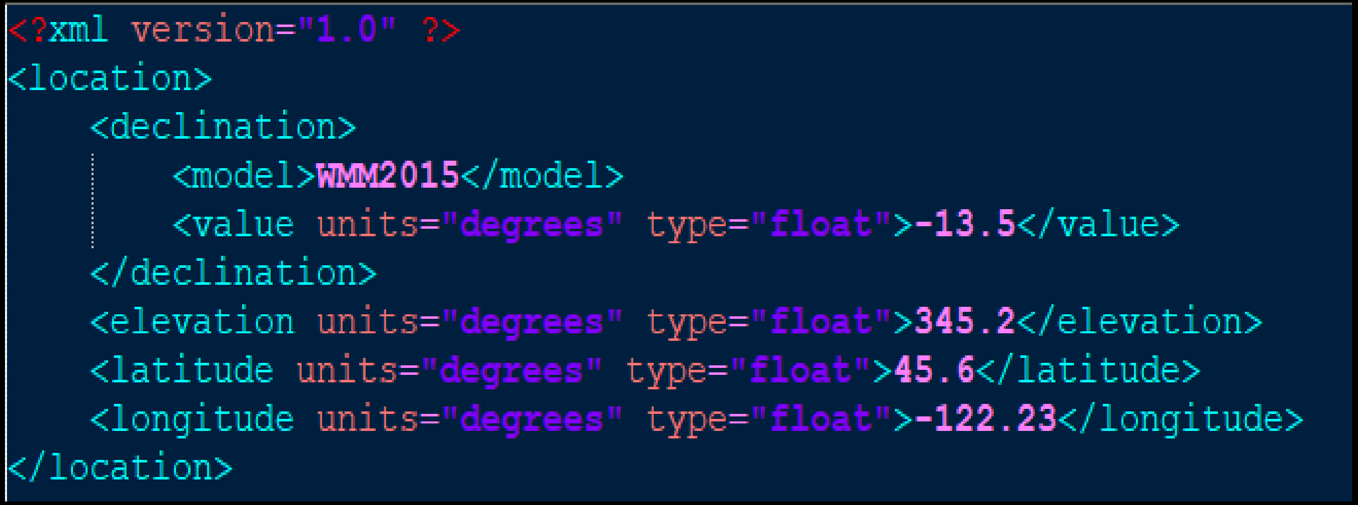




```
In [19]: from mt_metadata.timeseries import Station

In [20]: example_station = Station()

In [21]: example_station.attribute_information("location.latitude")
location.latitude:
    alias: ['lat']
    description: latitude of location in datum specified at survey level
    example: 23.134
    options: []
    required: True
    style: number
    type: float
    units: degrees

In [22]: example_station.location.latitude = "45:20:10"

In [23]: example_station.location.latitude
Out[23]: 45.336111111111116
```

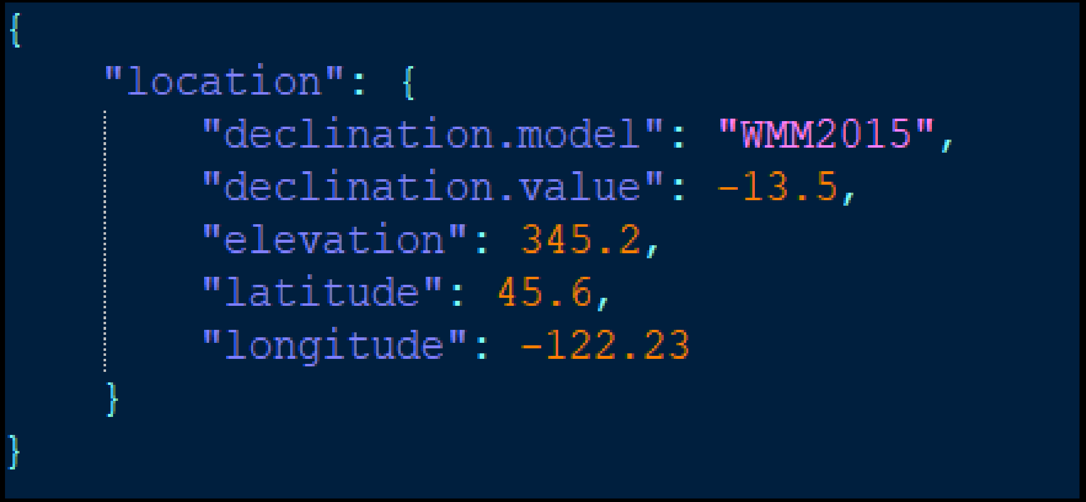
MT METADATA: REPRESENTATION

XML



```xml
<?xml version="1.0" ?>
<location>
    <declination>
        <model>WMM2015</model>
        <value units="degrees" type="float">-13.5</value>
    </declination>
    <elevation units="degrees" type="float">345.2</elevation>
    <latitude units="degrees" type="float">45.6</latitude>
    <longitude units="degrees" type="float">-122.23</longitude>
</location>
```

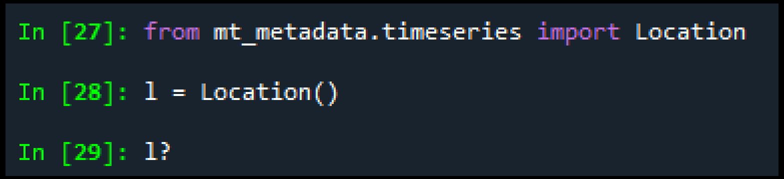
JSON



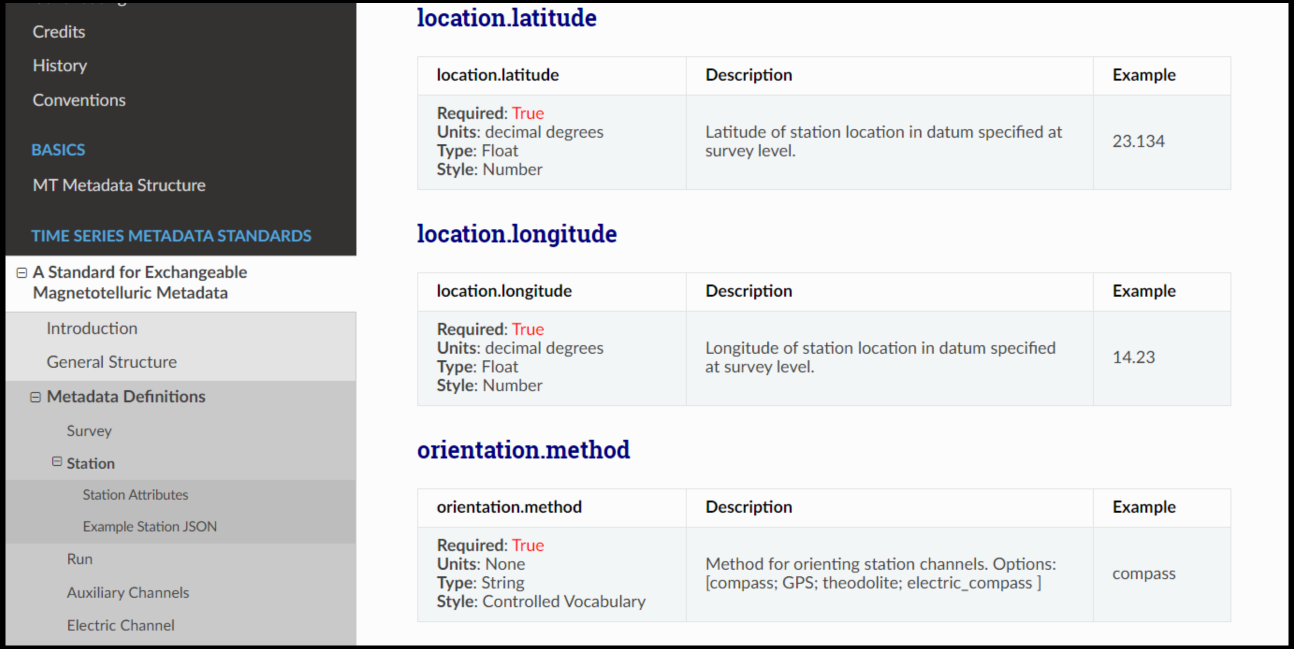
```json
{
    "location": {
        "declination.model": "WMM2015",
        "declination.value": -13.5,
        "elevation": 345.2,
        "latitude": 45.6,
        "longitude": -122.23
    }
}
```

MT METADATA STANDARDS: HELP

Station — MT Metadata 0.1.1 documentation (mt-metadata.readthedocs.io)



```
In [27]: from mt_metadata.timeseries import Location

In [28]: l = Location()

In [29]: l?
```



Credits
History
Conventions

BASICS

MT Metadata Structure

TIME SERIES METADATA STANDARDS

⊟ A Standard for Exchangeable Magnetotelluric Metadata
- Introduction
- General Structure
- ⊟ Metadata Definitions
  - Survey
  - ⊟ Station
    - Station Attributes
    - Example Station JSON
  - Run
  - Auxiliary Channels
  - Electric Channel

location.latitude

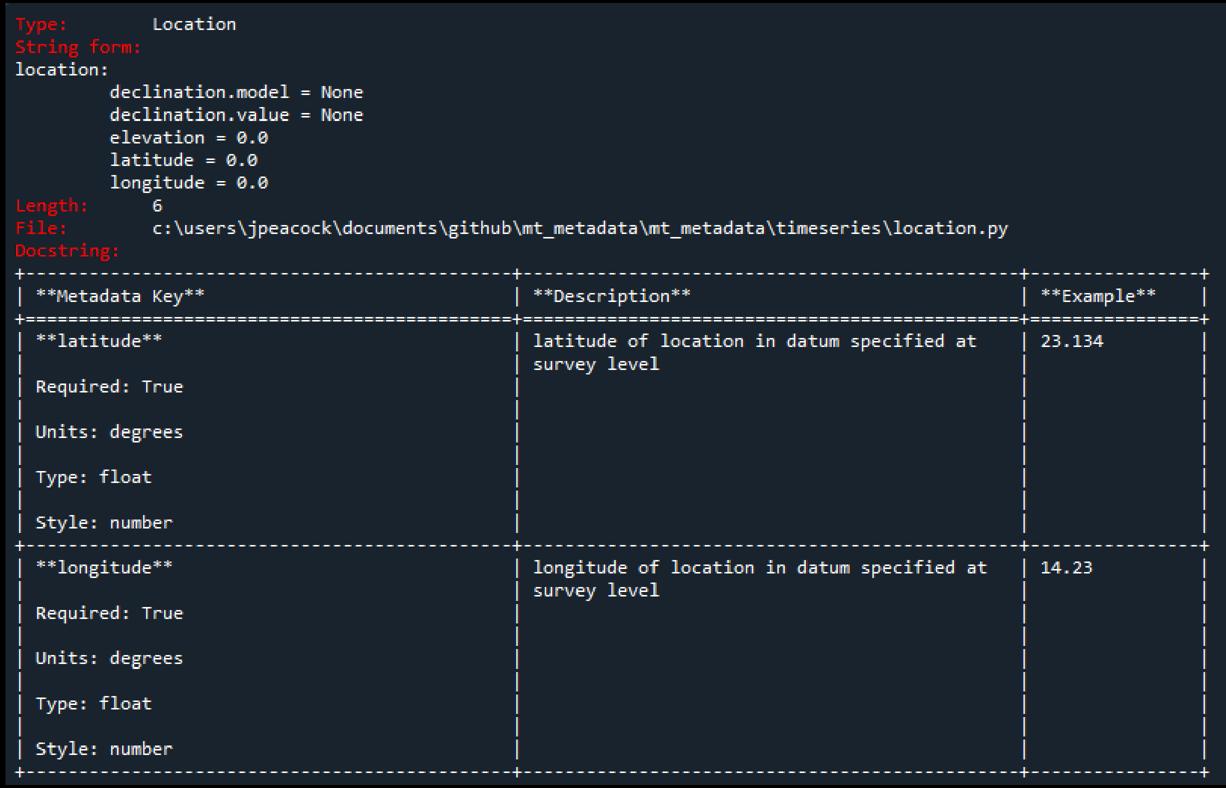
location.latitude	Description	Example
Required: True **Units:** decimal degrees **Type:** Float **Style:** Number	Latitude of station location in datum specified at survey level.	23.134

location.longitude

location.longitude	Description	Example
Required: True **Units:** decimal degrees **Type:** Float **Style:** Number	Longitude of station location in datum specified at survey level.	14.23

orientation.method

orientation.method	Description	Example
Required: True **Units:** None **Type:** String **Style:** Controlled Vocabulary	Method for orienting station channels. Options: [compass; GPS; theodolite; electric_compass]	compass



```
Type:           Location
String form:
location:
        declination.model = None
        declination.value = None
        elevation = 0.0
        latitude = 0.0
        longitude = 0.0
Length:         6
File:           c:\users\jpeacock\documents\github\mt_metadata\mt_metadata\timeseries\location.py
Docstring:
+----------------------------------------------+----------------------------------------------+--------------+
| **Metadata Key**                             | **Description**                              | **Example**  |
+==============================================+==============================================+==============+
| **latitude**                                 | latitude of location in datum specified at   | 23.134       |
|                                              | survey level                                 |              |
| Required: True                               |                                              |              |
|                                              |                                              |              |
| Units: degrees                               |                                              |              |
|                                              |                                              |              |
| Type: float                                  |                                              |              |
|                                              |                                              |              |
| Style: number                                |                                              |              |
+----------------------------------------------+----------------------------------------------+--------------+
| **longitude**                                | longitude of location in datum specified at  | 14.23        |
|                                              | survey level                                 |              |
| Required: True                               |                                              |              |
|                                              |                                              |              |
| Units: degrees                               |                                              |              |
|                                              |                                              |              |
| Type: float                                  |                                              |              |
|                                              |                                              |              |
| Style: number                                |                                              |              |
+----------------------------------------------+----------------------------------------------+--------------+
```

MT METADATA: TO DO

- Publish the standards
    - Create XSD and JSON Schema files to validate against
- Create a permanent working group within IAGA Division VI focused on metadata standards

MTH5

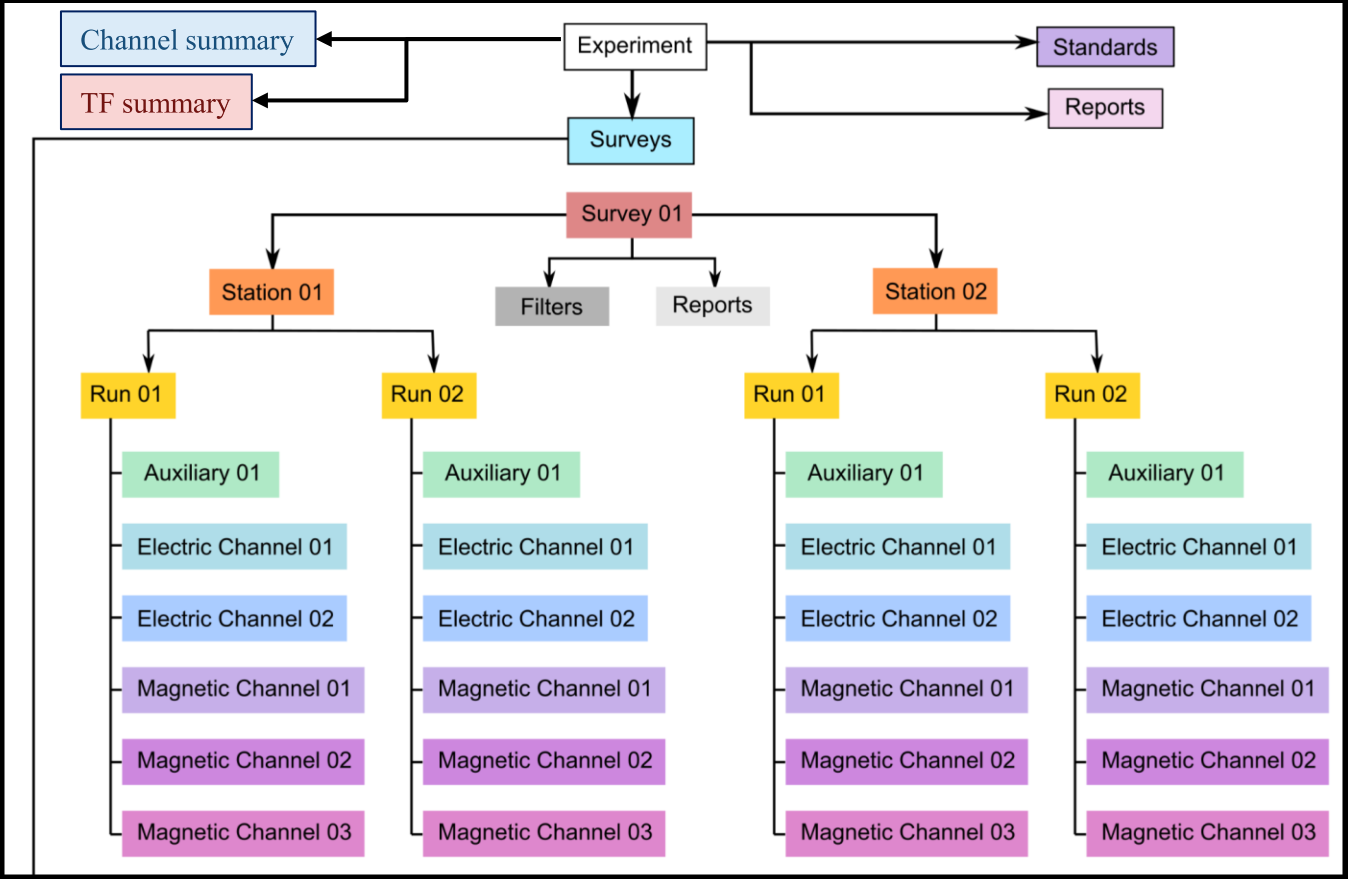
Goal: Develop a standardized HDF5 container to store time series data and transfer functions while providing open-source tools to read, write, and access the data from an HDF5 file

Benefits of HDF5

1. Base code is open-source and community driven
2. Files are flexible
3. File size is only limited by available resources
4. Files are portable across nearly all operating systems and platforms from laptops to cloud based parallel systems, and have no limitations on the number of data objects contained within the file
5. RAM requirements are optimized with a high-performance input/output system to only load requested data
6. Chunking and compression are inherent to optimize efficient storage and retrieval
7. A single writer with multiple readers (SWMR) is supported
8. Parallel reading/writing needs parallel HDF5 (PHDF5). For cloud environments the HDF Group provides highly scalable data services (HSDS)

Funded by IRIS and USGS
Community for Data Integration

MTH5



Each level has it own MTH5 group including:

- Metadata
- add/get/remove group

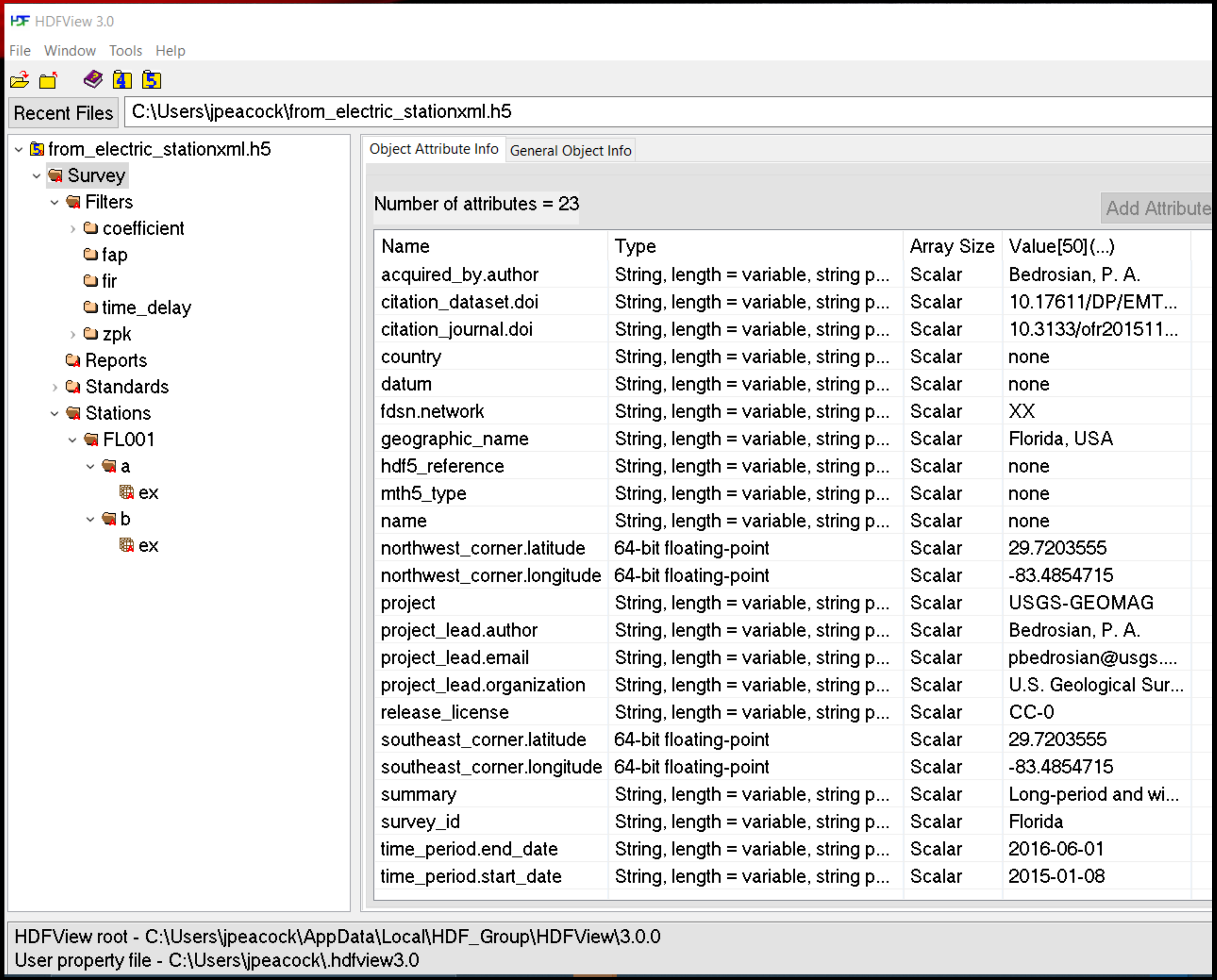
Data container is xarray

- Lazy access
- Has a container for metadata
- Indexed by time
- Easily searchable, indexed, etc like a Pandas Dataframe

Includes a summary tables of all channels and transfer functions for easy querying

MTH5

Example File (HDFView)

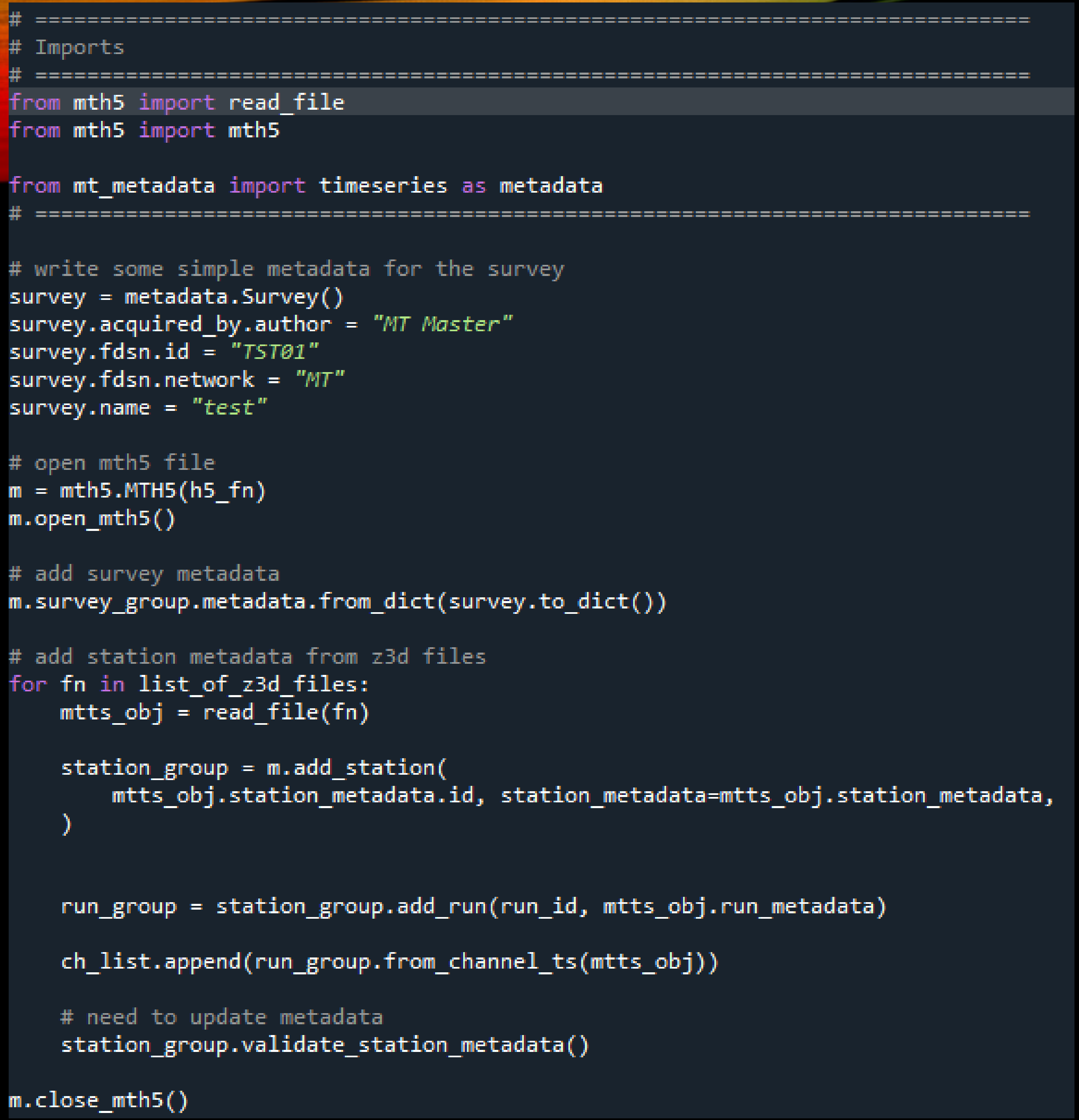


C:\Users\jpeacock\from_electric_stationxml.h5

Number of attributes = 23

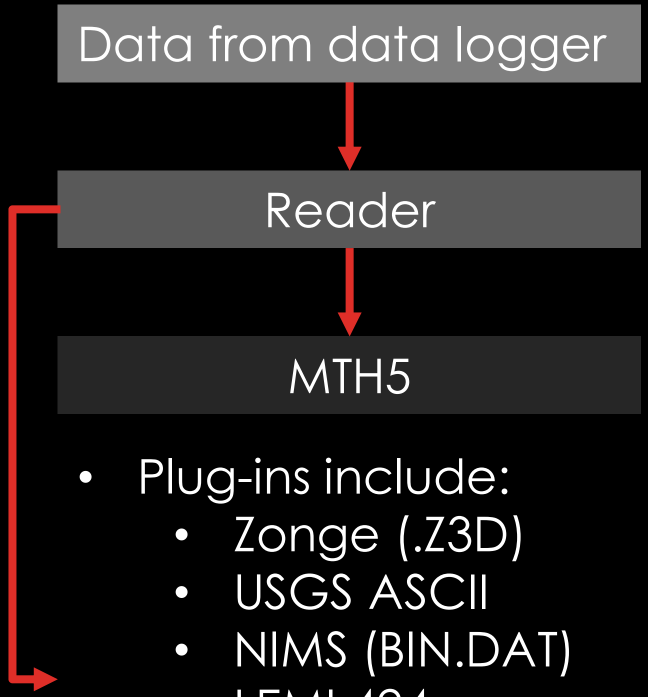
Name	Type	Array Size	Value[50](...)
acquired_by.author	String, length = variable, string p...	Scalar	Bedrosian, P. A.
citation_dataset.doi	String, length = variable, string p...	Scalar	10.17611/DP/EMT...
citation_journal.doi	String, length = variable, string p...	Scalar	10.3133/ofr201511...
country	String, length = variable, string p...	Scalar	none
datum	String, length = variable, string p...	Scalar	none
fdsn.network	String, length = variable, string p...	Scalar	XX
geographic_name	String, length = variable, string p...	Scalar	Florida, USA
hdf5_reference	String, length = variable, string p...	Scalar	none
mth5_type	String, length = variable, string p...	Scalar	none
name	String, length = variable, string p...	Scalar	none
northwest_corner.latitude	64-bit floating-point	Scalar	29.7203555
northwest_corner.longitude	64-bit floating-point	Scalar	-83.4854715
project	String, length = variable, string p...	Scalar	USGS-GEOMAG
project_lead.author	String, length = variable, string p...	Scalar	Bedrosian, P. A.
project_lead.email	String, length = variable, string p...	Scalar	pbedrosian@usgs....
project_lead.organization	String, length = variable, string p...	Scalar	U.S. Geological Sur...
release_license	String, length = variable, string p...	Scalar	CC-0
southeast_corner.latitude	64-bit floating-point	Scalar	29.7203555
southeast_corner.longitude	64-bit floating-point	Scalar	-83.4854715
summary	String, length = variable, string p...	Scalar	Long-period and wi...
survey_id	String, length = variable, string p...	Scalar	Florida
time_period.end_date	String, length = variable, string p...	Scalar	2016-06-01
time_period.start_date	String, length = variable, string p...	Scalar	2015-01-08

HDFView root - C:\Users\jpeacock\AppData\Local\HDF_Group\HDFView\3.0.0
User property file - C:\Users\jpeacock\.hdfview3.0



```python
# ==============================================================================
# Imports
# ==============================================================================
from mth5 import read_file
from mth5 import mth5

from mt_metadata import timeseries as metadata
# ==============================================================================

# write some simple metadata for the survey
survey = metadata.Survey()
survey.acquired_by.author = "MT Master"
survey.fdsn.id = "TST01"
survey.fdsn.network = "MT"
survey.name = "test"

# open mth5 file
m = mth5.MTH5(h5_fn)
m.open_mth5()

# add survey metadata
m.survey_group.metadata.from_dict(survey.to_dict())

# add station metadata from z3d files
for fn in list_of_z3d_files:
    mtts_obj = read_file(fn)

    station_group = m.add_station(
        mtts_obj.station_metadata.id, station_metadata=mtts_obj.station_metadata,
    )


    run_group = station_group.add_run(run_id, mtts_obj.run_metadata)

    ch_list.append(run_group.from_channel_ts(mtts_obj))

    # need to update metadata
    station_group.validate_station_metadata()

m.close_mth5()
```

MTH5



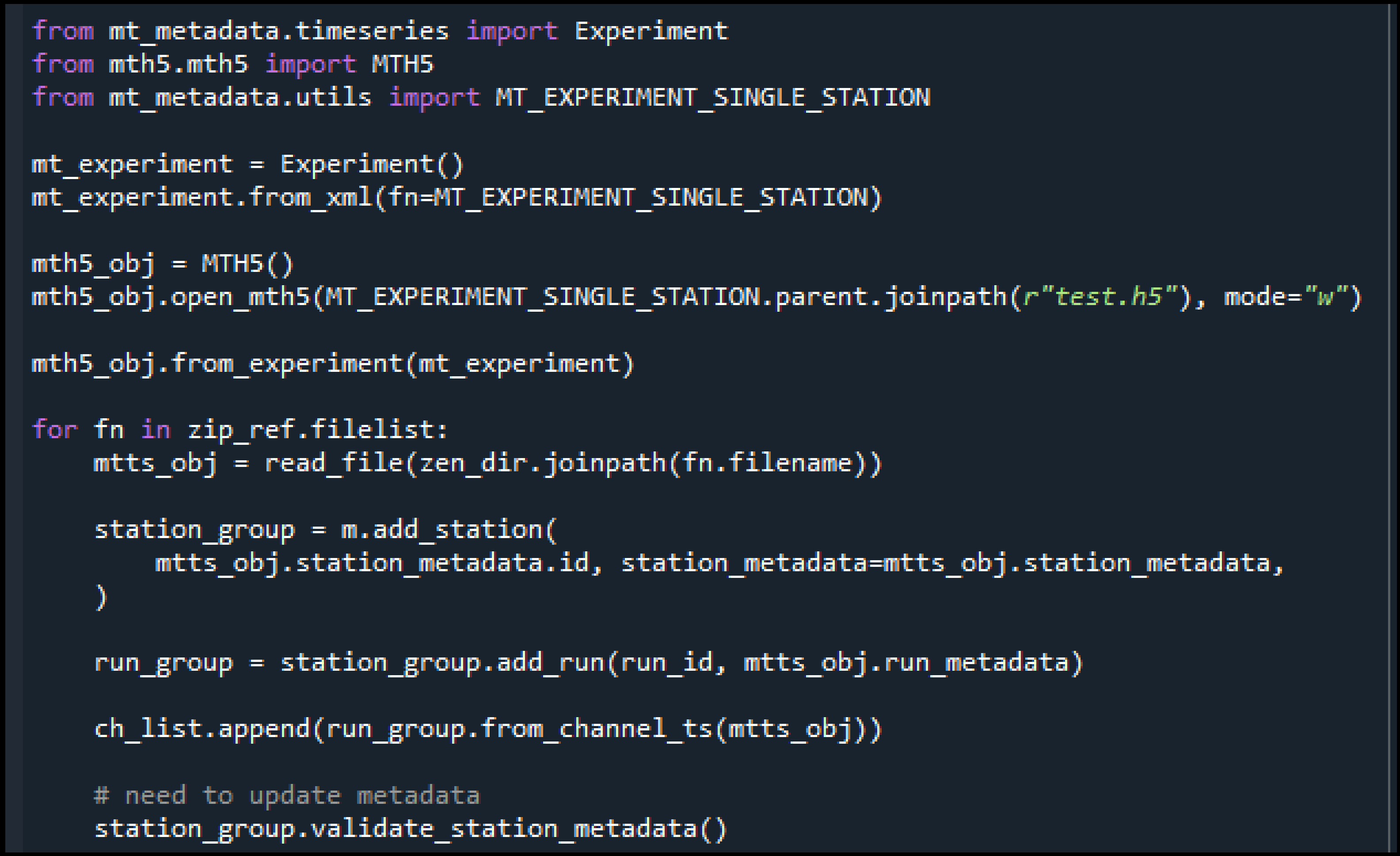
Data from data logger

Reader

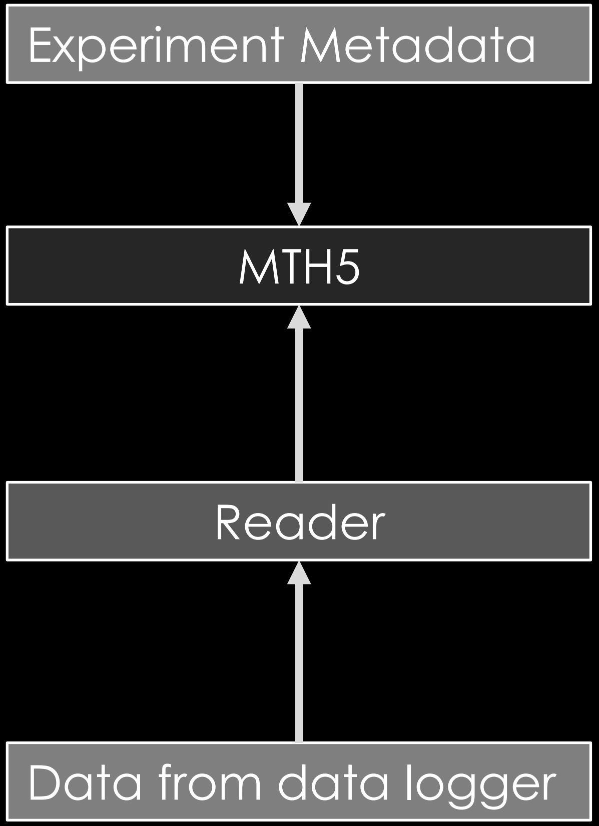
MTH5

- Plug-ins include:
  - Zonge (.Z3D)
  - USGS ASCII
  - NIMS (BIN.DAT)
  - LEMI 424
  - Needs to be extended to other data types like Phoenix (help)

MTH5



```python
from mt_metadata.timeseries import Experiment
from mth5.mth5 import MTH5
from mt_metadata.utils import MT_EXPERIMENT_SINGLE_STATION

mt_experiment = Experiment()
mt_experiment.from_xml(fn=MT_EXPERIMENT_SINGLE_STATION)

mth5_obj = MTH5()
mth5_obj.open_mth5(MT_EXPERIMENT_SINGLE_STATION.parent.joinpath(r"test.h5"), mode="w")

mth5_obj.from_experiment(mt_experiment)

for fn in zip_ref.filelist:
    mtts_obj = read_file(zen_dir.joinpath(fn.filename))

    station_group = m.add_station(
        mtts_obj.station_metadata.id, station_metadata=mtts_obj.station_metadata,
    )

    run_group = station_group.add_run(run_id, mtts_obj.run_metadata)

    ch_list.append(run_group.from_channel_ts(mtts_obj))

    # need to update metadata
    station_group.validate_station_metadata()
```



Experiment Metadata

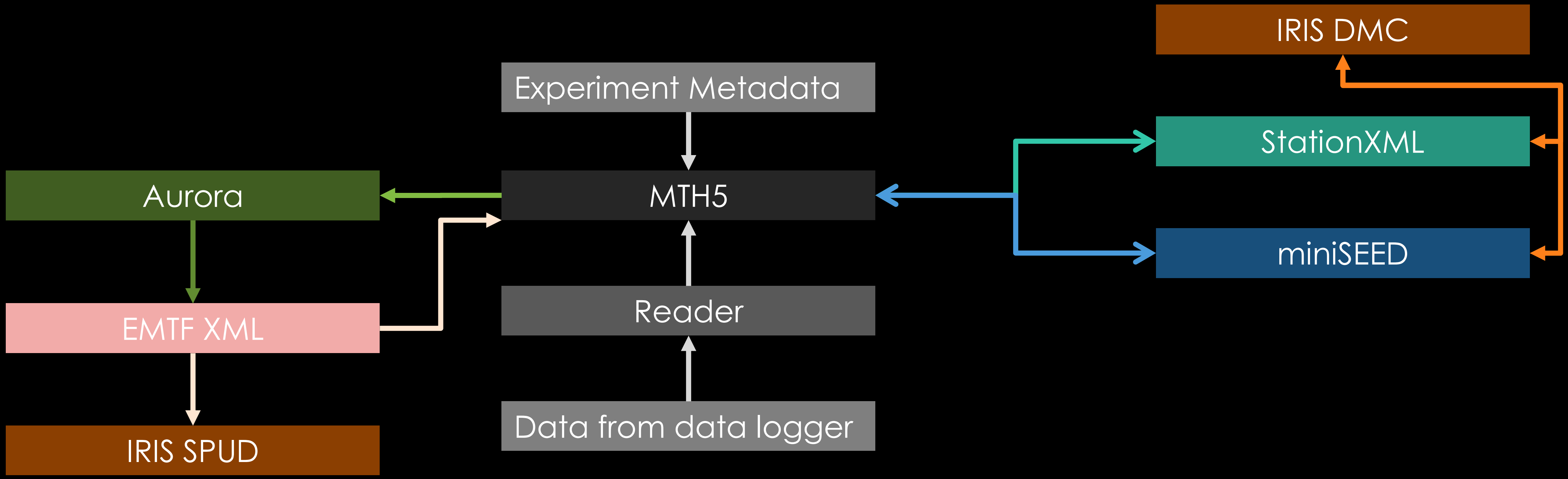
MTH5

Reader

Data from data logger

MTH5

Example for data collected by IRIS PASSCAL MT instruments



IRIS DMC

Experiment Metadata

StationXML

Aurora

MTH5

miniSEED

EMTF XML

Reader

Data from data logger

IRIS SPUD

MTH5

To Do:

- Extend data readers
- Time series Visualizer (in development)
- Extend parallel access
- Button push to calibrate data (should be done by early May)
- Figure out an efficient way to transfer an MTH5 file over a network
  → miniseed + Experiment metadata

OPEN-SOURCE FRAMEWORK

CURRENT REPOSITORIES

mth5

- Standardized HDF5 container for time series data
- Can store an entire survey in one file
- Provides tools to access data, read various file types

Developers: *Peacock, Kappler, Heagy*
Funding: *USGS CDI, IRIS*

mt_metadata

- A package to standardize metadata for time series and transfer functions
- All metadata is validated against "accepted" standards

Developers: *Peacock, Kappler, Heagy*
Funding: *USGS CDI, IRIS*

aurora

- Compute the transfer functions for MT and GDS
- Uses MTH5 files as input

Developers: *Kappler, Heagy, Peacock*
Funding: *IRIS*

MTpy

- Tools to read/write, analyze, plot transfer functions
- Read/write input output files for existing processing and modeling codes

Developers: *Kirkby, Zhang, GA, Peacock*
Funding: *Partial USGS, GA*

AURORA

- Transfer function estimation based mainly on Gary Egbert's EMTF
- Funded by IRIS
- Main developers: Karl Kappler, Lindsey Heagy, advised by Doug Oldenburg
- In development, first version will be released in September

- Will be hosted by SimPEG group
- Input is MTH5
- Output will be a Transfer Function object that can write EMTF XML, EDI,…

MTPY

GOAL: Develop tools to deal with MT transfer functions

Collaborative effort:

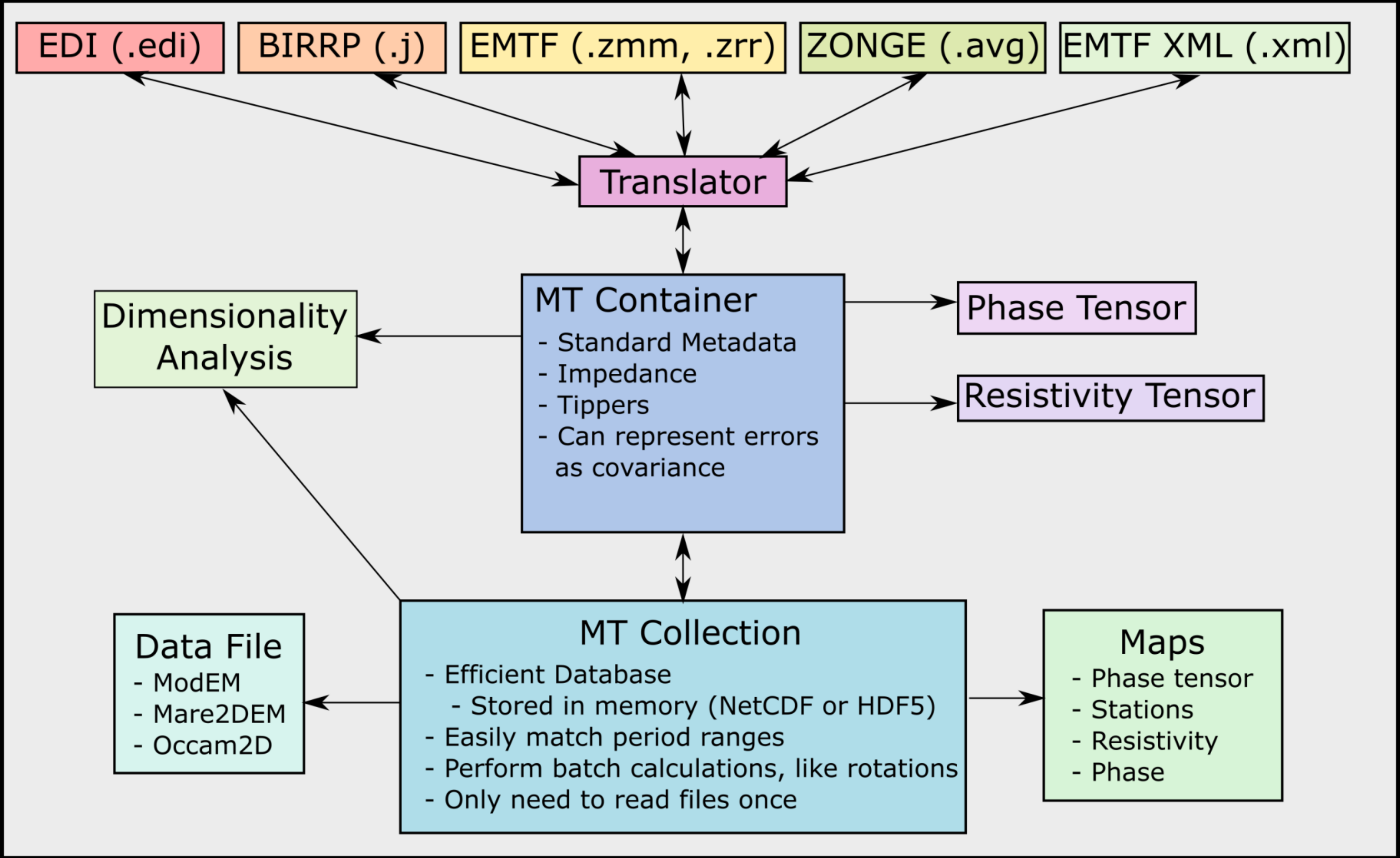
- Jared Peacock (USGS)
- Geoscience Australia (going in a different direction now)
- Seems to be growing, join the fun, don't be shy

For an in-depth tutorial see Alison's EMinar

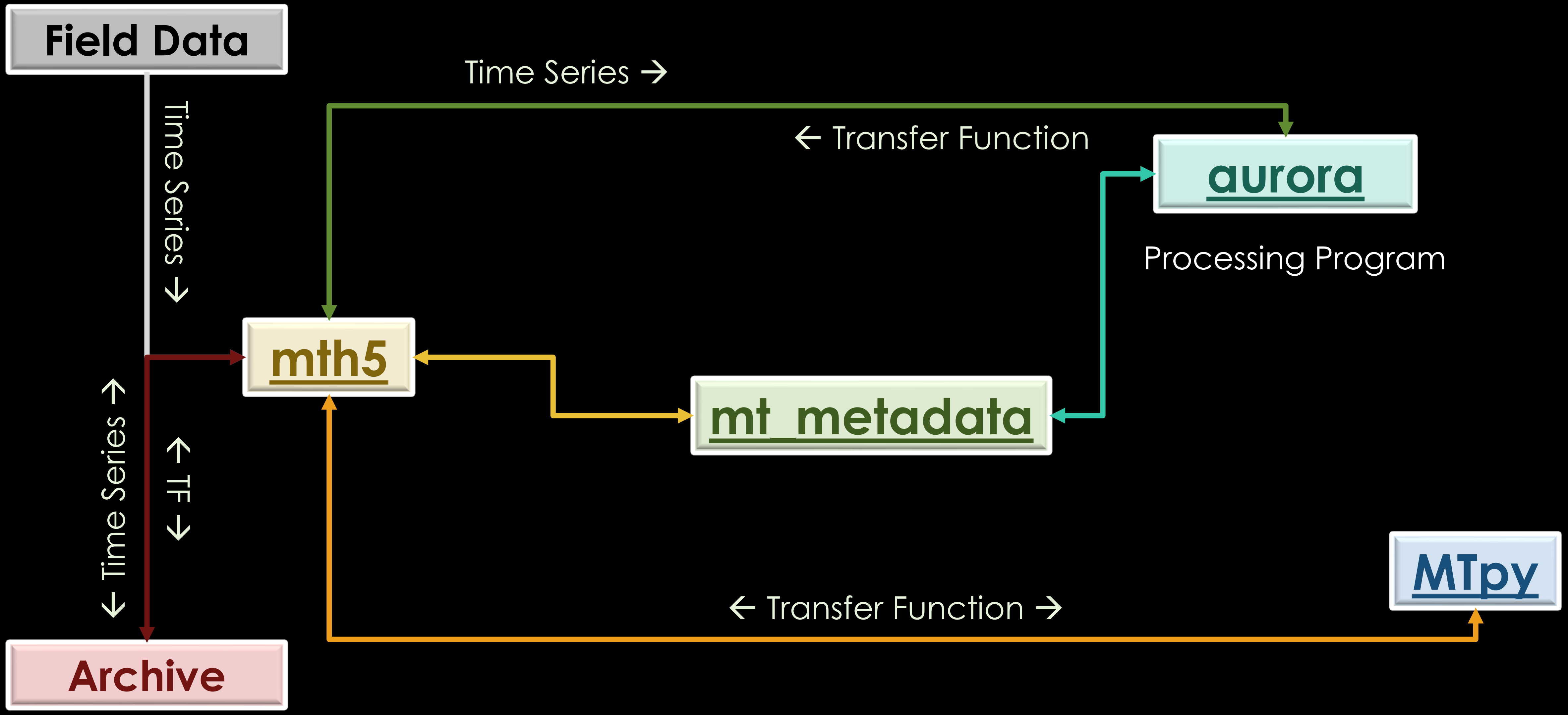
Alison's EMinar video
Alison's EMinar slides

- Publications:
  - Krieger, L. & Peacock, J. 2014. A Python Toolbox for Magnetotellurics. Computers and Geosciences, v.72, p167-175
  - Kirkby, A., Zhang, F., Peacock, J., Hassan, R., & Duan, J. (2019). The MTPy software package for magnetotelluric data analysis and visualisation. Journal of Open Source Software, 4(37), 1358-1358. doi:10.21105/joss.01358

MTPY VERSION 2

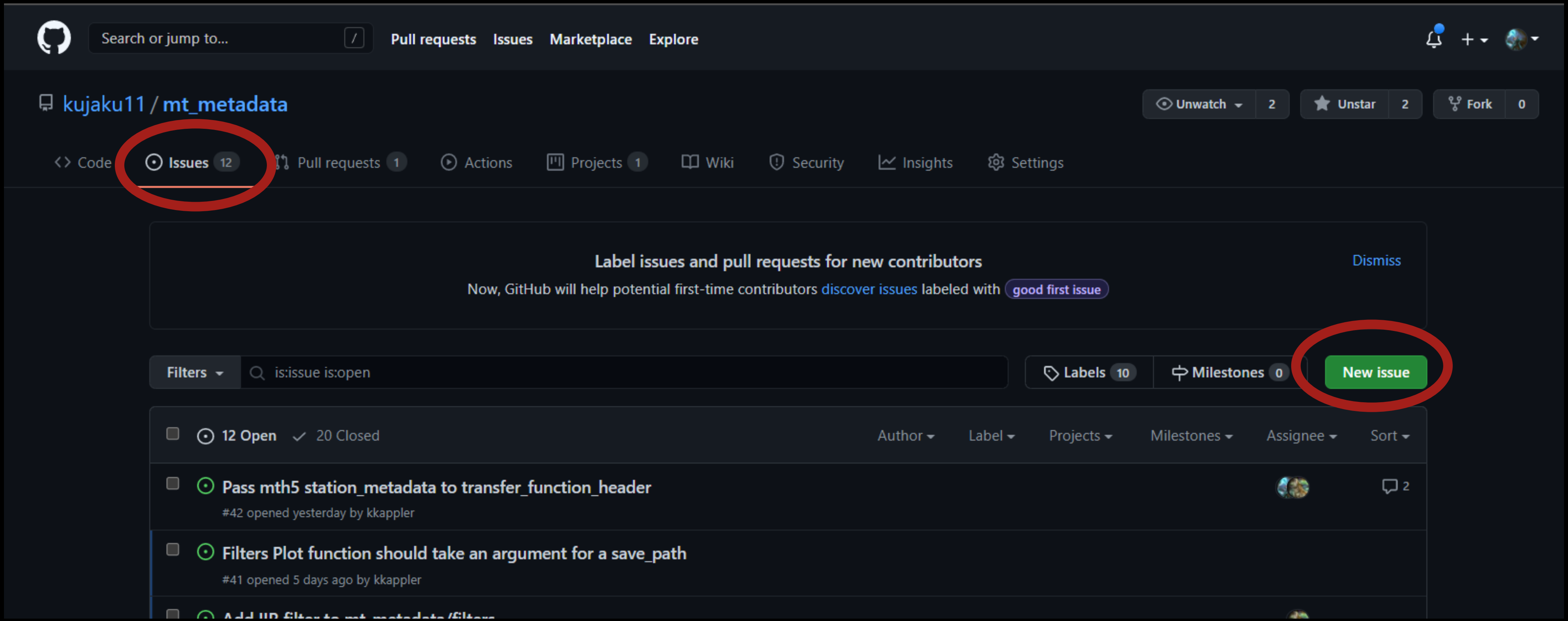


PACKAGE SYNERGY



GET INVOLVED

Git Issues



Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

is:issue is:open

12 Open ✓ 20 Closed

○ Pass mth5 station_metadata to transfer_function_header
#42 opened yesterday by kkappler — 2

○ Filters Plot function should take an argument for a save_path
#41 opened 5 days ago by kkappler

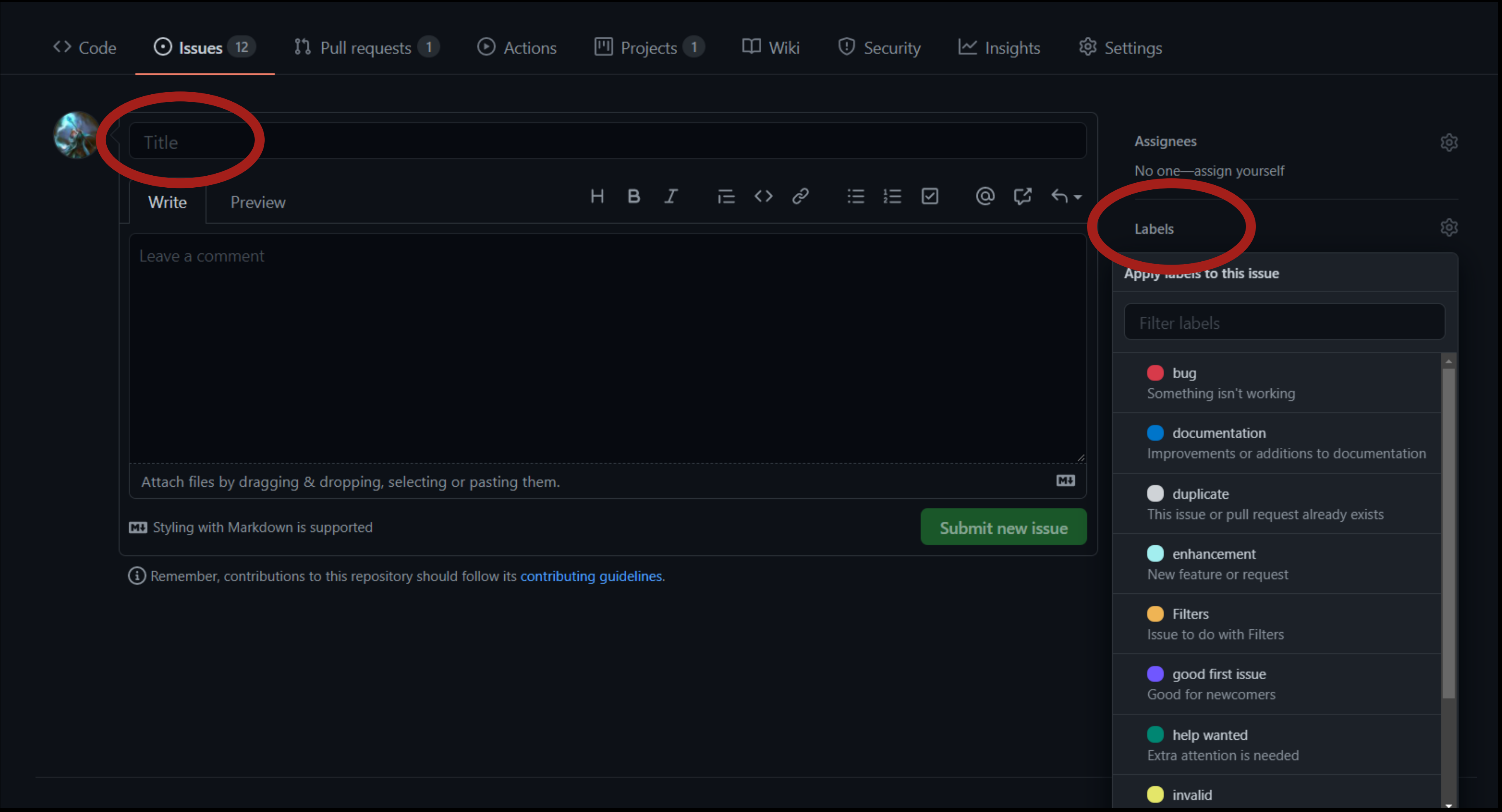
○ Add IIR filter to mt_metadata/filters

Can be:

- Bug
- Suggestion
- Broad idea

GET INVOLVED

Git Issues



Remember, contributions to this repository should follow its contributing guidelines.

bug	Something isn't working
documentation	Improvements or additions to documentation
duplicate	This issue or pull request already exists
enhancement	New feature or request
Filters	Issue to do with Filters
good first issue	Good for newcomers
help wanted	Extra attention is needed
invalid	

Create a useful and explanatory title: “Bug in mtpy.mt.core.MT”

Put a label on the issue so developers can sort and assign tasks.

FUTURE

How to make these packages Community driven?

- Start using and breaking things
  - *Suggest starting from the main or master branches on most of the packages, check develop branches too*
- Create issues
- Create pull requests
- Slack (SimPEG, COOPERATEM, MTNet)

How to get the greater community (young and old) to use these packages

- Make these packages standards with students and people new to the community
- Make the packages so good it would be hard not to use them

LIVE EXAMPLES

WHAT COULD GO WRONG