Package documentation
AMDAPy

Schulz Alexandre

June 23, 2021

0.1 amdapy functionalities

List here basic functions that are within the scope of *amdapy* :

- Download public datasets, parameters from AMDA
- Download private parameters from user space
- Resample dataset (defaults to original sampling time)
- Data structures available :
  - Datasets
  - Parameters
  - Time tables
  - Catalogs
- Explore AMDA database, search criterion :
  - mission
  - instrument
  - start and stop time
  - units
  - observed region
  - measurement type
  - more, ...
- Retrieve SPASE metadata pretaining to mission, instruments, people, datasets
- Full documentation
- Jupyter notebook

List of objects :

- amdapy.Mission(id=None)
  - id : name of the mission, it is unique within AMDA
- amdapy.Instrument(id=None, mission=None)
  - id : name of the instrument, unique within its corresponding mission
- amdapy.AmdaDataFrame(id=None)
  - has a *pandas_df* method that returns the content of the datastructure in a dataframe object
  - can specify the start and stop times, sampling rate, ...
- amdapy.Dataset(AmdaDataFrame, parameters=[])
  - contains a list of parameters
- amdapy.Parameter(AmdaDataFrame)
  - has id, units, name, shape
- amdapy.TimeTable(AmdaDataFrame)
- amdapy.Catalog(AmdaDataFrame)

Iterators :

- amdapy.datasets(mission=None, instrument=None)
- amdapy.missions()
- amdapy.instruments(mission=None)
- amdapy.parameters(mission=None, instrument=None, dataset=None)
- amdapy.derived_parameters(userid=None, password=None)

0.2 amdapy.Mission

In AMDA all mission are uniquely identified by their name, which is stored in the Mission.id attribute.

Missions have SPASE

0.3 The AMDAPy package

The aim of this package is to propose a easy to use python interface for accessing data stored on the AMDA plateform (and similar plateforms), applying common transformation and visualizing data.

Data stored on AMDA is mostly used for astronomic and plasma physics, as such some functions and visualizing methods can be very specific to a certain used case.

We would like to offer a collection of well documented functions that can be used by researchers to rapidly test known models and transformation on new datasets and visualize the results. Ideally the output plots should be as close to possible to publication quality.

When working on a new subject concerning dataset X a researcher may produce a variety of plots and tranformation functions. When the results finaly appear on paper it is not always easy to reproduce the results. The plateform should offer a centralized and safe way of storing papers and the simulation codes used for producing the results.

Lets try to describe the structure of the plateform such that we imagine using it. The plateform can be installed locally on the users machine by installing the amdapy package, otherwise we imagine having a web interface that can be accessed by the public.

When used locally : user can create a local session for storing data, apply transformation and plotting functions and store the results on local machine. The user can search for datasets (by name, by type of content), transformations and plots. The last point reveals the importance of describing the data. Dataset can be described using various norms (SPASE, ...), the plateform should be able to extract the most important information. It is this information that allows us to determine if the dataset is compatible with the use of certain transformation functions. For example we might want to define a transformation that computed coordinate values in coordinate system B from data expressed in coordinate system A. The input data must verify some conditions : the data should have 3 spatial dimensions, if it is not the case then the dataset is not compatible with the transformation. These notions will need to be further defined in the rest of this document.

Dataset important components : variables : datatype, name, fillvalue

0.3.1 Used locally

It is common for researchers to retrive data they are interested in to their local machine and perform the desired task locally. As it is currently implemented AMDA only allows visualizing data from within the plateform itself. This creates some complications : data is retrieved for each plot request which can be time consumming for large datasets. To address this issue the user of the API can load a specific dataset to a local storage structure allowing much faster data retrieval. Additionnaly the user can then define transformations and visualization function tailored to his specific needs.

0.3.2 Web interface

The plateform should be accessible and editable through a web interface. The interface is composed of a dataset tree (containing datasets from various providers), a transformation tree (containing user defined functions that can be used on the data), a plot tree (containing the user defined plotting functions), etc... (notebook tree, contains user defined notebooks).

The user can then apply transformation and execute plotting. Code is executed on the server ? or on the local machine (need javascript). The user can see and edit the code of the transformation and plotting functions to adapt them to the researchers use case. Once edited the new function is saved to a user specific database. Functions defined this way can then be retrived through the API call (with user authentification).

0.3.3 Package structure

The package is composed of a number of subpackages that have various functionalities:

- core : functions and classes used throughout the project (base dataset representation, dataset identification, time manipulation, etc...)
- vi : functions and classes for manipulation Virtual Instruments (mostly used by the database administrator for adding datasets to AMDA.
- plots : functions and classes for visualizing the data.
- transform : functions and classes for transforming the data (Machine Learning module will be stored here)
- scripts : collection of scripts.
- db : collection of classes for managing the databases the packages needs.

The project will be composed of a variety of databases that each have different function :

- AMDA database : the main source of datafiles.
- Session database : a local database set up on the users machine allowing him to have access to the data even without an internet access.
- Transformation database : a database containing transformation functions that the used can query to retrieve commonly used functions.

- Plot database : similar to the transformation database but containing plotting functions.

0.4 Data providers

0.5 Transformations

0.6 Plotting