VisData user manual

Toni Sagristà Sellés
Astronomisches Rechen-Institut
Zentrum für Astronomie Heidelberg
UNIVERSITÄT HEIDELBERG
Heidelberg, Baden-Württemberg, Deutschland

March 17, 2014

Abstract

The VisData module aims to provide a flexible solution for the extraction and transformation of astronomical data. This tool provides a fast and straightforward way of getting the right data in the right format without having to write a single line of code. Even though it was initially intended for the fetching of Gaia data for visualisation purposes, it can be used for any other tasks.

1 Introduction

The original aim of this package was to aid in the preparation of data from ESA's Gaia space mission for their visualisation in planetarium domes. It started as a straightforward module which loaded some data from a database using the TAP client in GAVO's VOTable python package, transformed it and wrote it to an output file. Then, in order to make it a bit more flexible it grew larger and larger until one could basically specify the desired output columns in any desired units and the module would do all the conversions and transformations for you. In the process, the astroutils package was created. It is a collection of utilities and functions that do common astronomical calculations, conversions and transformations.

2 VisData description

The VisData module gets a set of data rows, objects, as input, applies a series of transformations on these objects and writes the result in the output. Each object has a set of attributes, which are usually referred to as columns in data files. The objects are usually astronomical objects containing positional attributes (spherical or cartesian coordinates in any reference system), motion attributes (proper motions or cartesian velocities), brightness, temperature, etc. The module offers three basic types transformations:

Unit transformations You can define the units in which you want your data in the output. This will only work if the input data source has the units defined. If the input data comes from a **database** via a TAP interface, the database must have the meta-information about the column units. If the input data is a **file**, the first row must contain the column names separated by whitespaces and the second row must contain the column units also separated by whitespaces. This is also the format of the output files, so that output files can be fed as program input data.

Coordinate system transformations The program will convert positions and velocities between coordinate systems if needed. Right now cartesian coordinates and spherical polar coordinates can be converted back and forth. The transformations are determined by the input columns found in the input data source and the specified output columns. If the input data source contains the azimuth and declination angles (in whatever reference system) and the radius, and the output data contains cartesian coordinates (x, y, z), the spherical-to-cartesian transformation will be applied. The same is also true for the inverse transformation.

Reference system transformations Not only will the program perform coordinate system transformations, but also astronomical reference system transformations. The supported systems are `equatorial`, `ecliptic` and `galactic`. In the case of spherical coordinates the reference system is found out using the column names (alpha and delta or ra and dec for equatorial, lambda and beta for ecliptic and l and b for galactic). In the case of cartesian coordinates, the reference system is found out using the configuration file (see 2).

Configuration

The software is configured using a `.ini` configuration file whose location must be passed as an argument.

```
> main.py visdata.ini
```

The configuration file is composed of three compulsory sections labeled by a name in brackets. Each section contains the configuration for a particular domain, namely `conf`, `input` and `output`. The `conf` section contains global configuration options. The `input` section contains all the information about the input data stream like its location, its format, etc. The `output` section contains all the necessary information about the output. Each section must contain a series of defined `key:value` pairs in order to tell the program where to get the data from, how to process it and where to write the result to. Let's see a configuration file example:

```
[conf]
logpath: log

[input]
# The input data source: The URL of the TAP service or the path location of the input file
#datasource: http://dc.zah.uni-heidelberg.de/__system__/tap/run/tap
data_source: /home/user/data/data01.txt

# In case of type=vo, here goes the ADQL query to get the data
query: SELECT TOP 500 gums.mw.alpha,gums.mw.delta,gums.mw.mualpha,gums.mw.mudelta FROM gums.mw

# The reference system of input positions and velocities
# OPTIONS: equatorial, galactic, ecliptic
in_refsys: equatorial

[output]
# Output folder where the output files will be written
```

```
out_filepath: out

# The columns of the output file. It will match the coordinate column names such as 'alpha', 'delta',
    'distance', 'x', 'y', 'z',
# 'mualpha', 'mudelta', 'radialvelocity', 'xdot', 'ydot', 'zdot' to table column names, so that it knows when
    to transform coordinates.
# Note that all these data must somehow be in the input of the program. One can not make up data, one can only
    transform it, so if
# the input does not have position information, neither in spherical nor cartesian coordinates, the output can
    not have it either.
# This is a source for exceptions, so please modify with caution.
out_columns: x y z xdot ydot zdot
#out_columns: alpha delta distance

# These are the output units for each column, in order. You MUST specify these, there's no way around.
# Some of the supported units (more info
    http://astropy.readthedocs.org/en/latest/units/#module-astropy.units.cds):
# Length:             pc, km, m, lyr
# Time:               a, yr, s, ms
# Angle:              mas, as, deg, rad
# Magnitude:          mag
# Temperature:        K
out_units: pc lyr km km/s km/s pc/a

# All output positions and velocities will use this coordinate system
# If output are cartesian coordinates:
# X - Points to the 0 point in the XY fundamental plane.
# Z - In the north direction, above the fundamental plane.
# Y - Is orthogonal to X, using the right-hand rule. XY is the fundamental plane.
# If output are spherical coordinates, the out_refsys must match the out_columns names.
# OPTIONS: equatorial, galactic, ecliptic
out_refsys: galactic
```

The configuration file is heavily commented so that one can edit it right away without need for any documentation. We can see the three sections `[conf]`, `[input]` and `[output]` have all a set of properties, described in the following sections.

conf

The `conf` section only has one property.

logpath The path to store the log files. It may be a relative or an absolute path.

input

The `input` section describes the input data.

data_source Contains the URL to the data source. It may be the URL to a TAP interface or a path to a file containing the data. The system will parse its value and decide what is the adequate loader.

query This property must be present only if the `data_source` points to a TAP interface and it must contain the ADQL query to execute. In the case of very large queries, it is recommended to execute them separately with a software that allows for asynchronous queries and input the result as a text.

in_refsys Contains the astronomical reference system of the **cartesian** coordinates (if any). Accepted values are `equatorial`, `ecliptic` and `galactic`. Please, note this must be specified if cartesian coordinates are present in the input data source. Also, this only affects cartesian coordinates, for the reference system of any spherical coordinates will be worked out using their names.

output

The `output` section describes the desired output data.

out_filepath Folder where the output files will be written. This can either be an absolute or a relative path. The output file names are auto-generated.

out_columns Whitespace-separated list of column names to be generated in the output. Coordinates and proper motions will be worked out automatically providing they exist in some form in the input data source. For example, if the input contains any kind of coordinate information (either in spherical or in cartesian coordinates) one can specify as many coordinates in the output as desired. The transformations will be made automatically. The same is true for proper motions/velocities.

out_units List of units of the `out_columns`. The only restriction is that the unit type must match the column type (i.e. one can not specify 'm/s' or 'rad' as a unit of length, but 'm' or 'km' would do. Since we use astropy's unit conversion functionalities, more information on the supported units can be found in http://astropy.readthedocs.org/en/latest/units/#module-astropy.units.cds

out_refsys The astronomical reference system of the output **cartesian** coordinates (if any). Accepted values are `equatorial`, `ecliptic` and `galactic`. This only affects cartesian coordinates, for the reference system of any spherical coordinates will be worked out using their names.

3 References and acknowledgements

The VisData module uses the astroutils package, which was developed at the same time for this sole purpose. The astroutils package itself uses a set of third-party libraries that have proven very reliable and useful.

- Christoph Gohlke's transformations library. A library providing matrix transformations and quaternions handling. http://www.lfd.uci.edu/gohlke/.
- Astropy's unit conversion. http://www.astropy.org.
- Numpy 1.7 http://www.numpy.org.
- VOTable by GAVO http://vo.ari.uni-heidelberg.de/soft/subpkgs