MESSENGERuvvs

Release 1.2.4

unknown

Oct 25, 2019

CONTENTS

MESSENGERuvvs provides an interface to reduced MESSENGER (MErcury Surface, Space ENvironment, GEochemistry, and Ranging) MASCS (Mercury Atmospheric and Surface Composition Spectrometer) UVVS (UltraViolet and Visible Spectrometer) data and to the nexoclom Neutral Exospheres and Cloud Model. See McClintock et al. Space Science Reviews, 131, 481-521, 2007, and Solomon et al., Mercury - The View After MESSENGER, Cambridge University Press, 2019, Ch 14-15, for more details on the MESSENGER observations and models.

CHAPTER

ONE

MESSENGERDATA

MESSENGER UVVS data class

exception MESSENGERuvvs.MESSENGERdata.**InputError**(*expression*, *message*)
Raised when a required parameter is not included in the inputfile.

class MESSENGERuvvs.MESSENGERdata.**MESSENGERdata**(*species=None*, *comparisons=None*)
Retrieve MESSENGER data from database. Given a species and set of comparisons, retrieve MESSSENGER UVVS data from the database. The list of searchable fields is given at database_fields.

Returns a MESSENGERdata Object.

Parameters

species Species to search. This is required because the data from each species is stored in a different database table.

query A SQL-style list of comparisons.

The data in the object created is extracted from the database tables using the query:

```
SELECT *
FROM <species>uvvsdata, <species>pointing
WHERE <query>
```

See examples below.

Class Atributes

species The object can only contain a single species.

frame Coordinate frame for the data, either MSO or Model.

query SQL query used to search the database and create the object.

data Pandas dataframe containing result of SQL query. Columns in the dataframe are the same as in the database except *frame* and *species* have been dropped as they are redundant. If models have been run, there are also columns in the form modelN for the Nth model run.

taa Median true anomaly for the data in radians.

model_label If *N* models have been run, this is a dictionary in the form `{'model0':label0, ..., 'modelN':labelN}` containing descriptions for the models.

model_strength If *N* models have been run, this is a dictionary in the form `{'model0':strength0, ..., 'modelN':strengthN}` containing modeled source rates in units of 10^{26} atoms/s.

Examples

1. Loading data

```
>>> from MESSENGERuvvs import MESSENGERdata

>>> CaData = MESSENGERdata('Ca', 'orbit = 36')

>>> print(CaData)
Species: Ca
Query: orbit = 36
Frame: MSO
Object contains 581 spectra.

>>> NaData = MESSENGERdata('Na', 'orbit > 100 and orbit < 110')

>>> print(NaData)
Species: Na
Query: orbit > 100 and orbit < 110
Frame: MSO
Object contains 3051 spectra.

>>> MgData = MESSENGERdata('Mg',
        'loctimetan > 5.5 and loctimetan < 6.5 and alttan < 1000')

>>> print(len(MgData))
45766
```

2. Accessing data.

- The observations are stored within the MESSENGERdata object in a pandas dataframe attribute called *data*. Please see the pandas documentation for more information on how to work with dataframes.

```
>>> print(CaData.data.head(5))
                        utc  orbit  merc_year  ...  loctimetan          ␣
↪slit               utcstr
unum                                           ...
3329 2011-04-04 21:24:11.820     36          0  ...   14.661961 ␣
↪Atmospheric  2011-04-04T21:24:11
3330 2011-04-04 21:25:08.820     36          0  ...   12.952645 ␣
↪Atmospheric  2011-04-04T21:25:08
3331 2011-04-04 21:26:05.820     36          0  ...   12.015670 ␣
↪Atmospheric  2011-04-04T21:26:05
3332 2011-04-04 21:27:02.820     36          0  ...   12.007919 ␣
↪Atmospheric  2011-04-04T21:27:02
3333 2011-04-04 21:27:59.820     36          0  ...   12.008750 ␣
↪Atmospheric  2011-04-04T21:27:59
```

(continues on next page)

(continued from previous page)

```
[5 rows x 29 columns]
```

- Individual observations can be extracted using standard Python slicing techniques:

```
>>> print(CaData[3:8])
Species: Ca
Query: orbit = 36
Frame: MSO
Object contains 5 spectra.

>>> print(CaData[3:8].data['taa'])
unum
3332    1.808107
3333    1.808152
3334    1.808198
3335    1.808243
3336    1.808290
Name: taa, dtype: float64
```

3. Modeling data

```
>>> inputs = Input('Ca.spot.Maxwellian.input')
>>> CaData.model(inputs, 1e5, label='Model 1')
>>> inputs..speeddist.temperature /= 2.  # Run model with different␣
↪temperature
>>> CaData.model(inputs, 1e5, label='Model 2')
```

4. Plotting data

```
>>> CaData.plot('Ca.orbit36.models.html')
```

5. Exporting data to a file

```
>>> CaData.export('modelresults.csv')
>>> CaData.export('modelresults.html', columns=['taa'])
```

export (*self, filename, columns=['utc', 'radiance']*)
Export data and models to a file. **Parameters**

filename Filename to export model results to. The file extension determines the format. Formats available: csv, pkl, html, tex

columns Columns from the data dataframe to export. Available columns can be found by calling the *keys()* method on the data object. Default = ['utc', 'radiance'] and all model result columns. Note: The default columns are always included in the output regardless of whether they are specified.

Returns

No outputs.

keys (*self*)
Return all keys in the object, including dataframe columns

set_frame (*self*, *frame=None*)
Convert between MSO and Model frames.

More frames could be added if necessary. If Frame is not specified, flips between MSO and Model.

CHAPTER

TWO

DATABASE_SETUP

MESSENGERuvvs.database_setup.**database_connect**(*database=None*, *port=None*, *return_con=True*)

Return a database connection to saved atomic data Wrapper for `psycopg2.connect()` that determines database and port to use.

Parameters

database Database to connect to. If not given, it must be supplied in the $HOME/.nexoclom configuration file.

port Port the database server uses. If not given, it must be supplied in the $HOME/.nexoclom configuration file.

return_con False to return database name and port instead of connection. Default = True

Returns

Database connection with autocommit = True unless return_con = False

Examples

```
>>> from atomicdataMB import database_connect
>>> database, port = database_connect(return_con=False)
>>> print(f'database = {database}; port = {port}')
database = thesolarsystemmb; port = 5432
>>> with database_connect() as con:
...     cur = con.cursor()
...     cur.execute('SELECT DISTINCT species from gvalues')
...     species = cur.fetchall()
>>> species = [s[0] for s in species]
>>> print(species)
['Ca', 'OH', 'O', 'Ti', 'C', 'Mg+', 'Na', 'Mg', 'H', 'Mn', 'He',
 'Ca+', 'K', 'S']
```

MESSENGERuvvs.database_setup.**messenger_database_setup**(*force=False*)

Setup the database from SQL database dump files. Repopulates the database using a SQL backup rather than the original IDL save files. See database_fields for a description of the tables and fields used by MESSENGERuvvs.

Parameters

force If True, deletes old database tables and remakes them. Default is False, which only creates the tables if necessary.

Returns

No output.

CHAPTER

THREE

DATABASEBACKUPS

Backup the MESSENGERuvvs database tables.

MESSENGERuvvs.databasebackups.**databasebackups**()

Backup the MESSENGERuvvs database tables.

Dump the MESSENGERuvvs data into SQL files that can be restored if necessary. Tables that are backed-up are: cauvvsdata, capointing, mguvvsdata, mgpointing, nauvvsdata, napointing, mesmercyear.

This function takes no arguments. The path to save the database dumps must be specified in $HOME/.nexoclom in the format `datapath = <path to data>`. The default database and port are `thesolarsystemmb` and `5432`. These can also be specified in the .nexoclom file.

CHAPTER

FOUR

INSTALLATION

MESSENGERuvvs can be installed with pip:

```
$ pip install MESSENGERuvvs
```

The MESSENGERuvvs data is not provided with this software package. I was not responsible for reducing it, and I'm not sure who I'm allowed to distribute it to. Please contact me if you are interested in this. MESSENGER MASCS/UVVS data is available from the Planetary Data System, although in a different format. We can probably work on a way to get that data in the format used here if necessary.

CHAPTER

FIVE

REPORTING ISSUES

This project is hosted on github at MESSENGERuvvs. Please report bugs or make comments there.

CHAPTER

SIX

CONTRIBUTING

Please let me know if you would like to make contributions.

Authors Matthew Burger

License LICENSE

PYTHON MODULE INDEX

m

INDEX