pysatSpaceWeather Documentation

Release 0.0.6-alpha

Burrell, Angeline G., Klenzing, Jeff, Stoneback, Russell, Pembroke

Jul 01, 2022

CONTENTS

This documentation describes the pysatSpaceWeather module, which contains routines to load common historic, real-time, and forecasted space weather indices as pysat.Instrument objects.

CHAPTER

ONE

OVERVIEW

Space weather indices are useful for a range of applications, including observation-based scientific analysis, model-based scientific analysis, and space weather predictions. pysatSpaceWeather addresses each of these needs by providing a pysat interface for solar and geomagnetic space weather indices. Different methods exist to easily load a single index from multiple sources and to convert between common equivalent indices (e.g., between Kp and Ap).

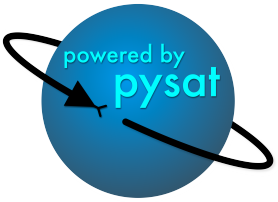
CHAPTER

TWO

INSTALLATION

The following instructions will allow you to install pysatSpaceWeather.

2.1 Prerequisites



pysatSpaceWeather uses common Python modules, as well as modules developed by and for the Space Physics community. This module officially supports Python 3.7+.

Common modules	Community modules
netCDF4	pysat >= 3.0.0
numpy	
pandas	
requests	
xarray	

2.2 Installation Options

2.2.1 PyPi

All official pysatSpaceWeather releases are available through the PyPi package manager.

```
pip install pysatSpaceWeather
```

2.2.2 GitHub

You can keep up to date with the latest changes at the GitHub repository.

1. Clone the git repository

```
git clone https://github.com/pysat/pysatSpaceWeather.git
```

2. Install pysatSpaceWeather: Change directories into the repository folder and run the setup.py file. There are a few ways you can do this:

   A. Install on the system (root privileges required):

   ```
   sudo python3 setup.py install
   ```

   B. Install at the user level:

   ```
   python3 setup.py install --user
   ```

   C. Install with the intent to develop locally:

   ```
   python3 setup.py develop --user
   ```

CHAPTER

THREE

CITATION GUIDELINES

When publishing work that uses pysatSpaceWeather, please cite the package and any package it depends on that plays an important role in your analysis. Specifying which version of pysatSpaceWeather used will also improve the reproducibility of your presented results.

3.1 pysatSpaceWeather

The most recent citation can be found at Zenodo. The examples here are from the first release.

- Burrell, A. G., et al. (2020). pysat/pysatSpaceWeather: Initial Release (Version 0.0.1). Zenodo. doi:10.5281/zenodo.3986139

```
@Misc{pysatSpaceWeather,
      author = {Burrell, A. G. and Klenzing, J. H. and Stoneback, R.
                and Pembroke, A. and Spence, C.},
      title  = {pysat/pysatSpaceWeather: Initial Release},
      year   = {2020},
      date   = {2020-08-14},
      url    = {https://github.com/pysat/pysatSpaceWeather},
      doi    = {10.5281/zenodo.3986139},
      publisher = {Zenodo},
      version = {v0.0.1},
      }
```

CHAPTER

FOUR

SUPPORTED INSTRUMENTS

4.1 ACE

The Space Weather Prediction Center (SWPC) provides several Advanced Composition Explorer (ACE) instrument data sets for use as real-time and historic measurements of the solar wind. This differs from the ACE scientific data, which is available at a greater latency from CDAWeb. Information about these data sets can be found at the SWPC ACE Solar-Wind page.

4.1.1 ACE EPAM

EPAM is the Electron, Proton, and Alpha Monitor onboard ACE.

Supports ACE Electron, Proton, and Alpha Monitor data.

Properties

platform
'ace' Advanced Composition Explorer

name
'epam' Electron, Proton, and Alpha Monitor

tag

- 'realtime' Real-time data from the Space Weather Prediction Center (SWPC)
- 'historic' Historic data from the SWPC

inst_id

- ''

Note

This is not the ACE scientific data set, which will be available at pysatNASA

Examples

The real-time data is stored by generation date, where each file contains the data for the current day. If you leave download dates empty, though, it will grab today's file three times and assign dates from yesterday, today, and tomorrow.

```
epam = pysat.Instrument('ace', 'epam', tag='realtime')
epam.download(start=epam.today())
epam.load(date=epam.today())
```

Warnings

The 'realtime' data contains a changing period of time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for 'realtime' data.

pysatSpaceWeather.instruments.ace_epam.**clean**(*self*)

Clean the real-time ACE data using the status flag.

pysatSpaceWeather.instruments.ace_epam.**init**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.ace_epam.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load the ACE space weather prediction data.

Parameters

fnames
[array-like] Series, list, or array of filenames.

tag
[str] Instrument tag (default='').

inst_id
[str] ACE instrument ID (default='').

Returns

data
[pandas.DataFrame] Object containing instrument data

meta
[pysat.Meta] Object containing metadata such as column names and units

See also:

pysat.instruments.methods.general.load_csv_data

4.1.2 ACE MAG

Supports ACE Magnetometer data.

Supports ACE Magnetometer data.

Properties

platform
'ace' Advanced Composition Explorer

name
'mag' Magnetometer

tag

- 'realtime' Real-time data from the Space Weather Prediction Center (SWPC)
- 'historic' Historic data from the SWPC

inst_id

- ''

Note

This is not the ACE scientific data set, which will be available at pysatNASA

Examples

The real-time data is stored by generation date, where each file contains the data for the current day. If you leave download dates empty, though, it will grab today's file three times and assign dates from yesterday, today, and tomorrow.

```
mag = pysat.Instrument('ace', 'mag', tag='realtime')
mag.download(start=mag.today())
mag.load(date=mag.today())
```

Warnings

The 'realtime' data contains a changing period of time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for 'realtime' data.

pysatSpaceWeather.instruments.ace_mag.**clean**(*self*)
Clean real-time ACE data using the status flag.

pysatSpaceWeather.instruments.ace_mag.**init**(*self*)
Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.ace_mag.**load**(*fnames*, *tag=''*, *inst_id=''*)
Load the ACE space weather prediction data.

Parameters

fnames
[array-like] Series, list, or array of filenames

tag
[str] Instrument tag, not used. (default='')

inst_id
[str] ACE instrument ID, not used. (default='')

Returns

data
[pandas.DataFrame] Object containing instrument data

meta
[pysat.Meta] Object containing metadata such as column names and units

See also:

`pysat.instruments.methods.general.load_csv_data`

4.1.3 ACE SIS

Supports ACE Solar Isotope Spectrometer data.

Supports ACE Solar Isotope Spectrometer data.

Properties

platform
'ace' Advanced Composition Explorer

name
'sis' Solar Isotope Spectrometer

tag

- 'realtime' Real-time data from the Space Weather Prediction Center (SWPC)
- 'historic' Historic data from the SWPC

inst_id

- ''

Note

This is not the ACE scientific data set, which will be available at pysatNASA

Examples

The real-time data is stored by generation date, where each file contains the data for the current day. If you leave download dates empty, though, it will grab today's file three times and assign dates from yesterday, today, and tomorrow.

```
sis = pysat.Instrument('ace', 'sis', tag='realtime')
sis.download(start=sis.today())
sis.load(date=sis.today())
```

Warnings

The 'realtime' data contains a changing period of time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for 'realtime' data.

pysatSpaceWeather.instruments.ace_sis.**clean**(*self*)

Clean real-time ACE data using the status flag.

pysatSpaceWeather.instruments.ace_sis.**init**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.ace_sis.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load the ACE space weather prediction data.

Parameters

fnames
[array-like] Series, list, or array of filenames.

tag
[str] Instrument tag, not used. (default='')

inst_id
[str] ACE instrument ID, not used. (default='')

Returns

data
[pandas.DataFrame] Object containing instrument data

meta
[pysat.Meta] Object containing metadata such as column names and units

See also:

pysat.instruments.methods.general.load_csv_data

4.1.4 ACE SWEPAM

Supports ACE Solar Wind Electron Proton Alpha Monitor data.

Supports ACE Solar Wind Electron Proton Alpha Monitor data.

Properties

platform
'ace' Advanced Composition Explorer

name
'swepam' Solar Wind Electron Proton Alpha Monitor

tag

- 'realtime' Real-time data from the Space Weather Prediction Center (SWPC)
- 'historic' Historic data from the SWPC

inst_id

- ''

Note

This is not the ACE scientific data set, which will be available at pysatNASA

Examples

The real-time data is stored by generation date, where each file contains the data for the current day. If you leave download dates empty, though, it will grab today's file three times and assign dates from yesterday, today, and tomorrow.

```
swepam = pysat.Instrument('ace', 'swepam', tag='realtime')
swepam.download(start=swepam.today())
swepam.load(date=swepam.today())
```

Warnings

The 'realtime' data contains a changing period of time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for 'realtime' data.

pysatSpaceWeather.instruments.ace_swepam.**clean**(*self*)

Clean real-time ACE data using the status flag.

pysatSpaceWeather.instruments.ace_swepam.**init**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.ace_swepam.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load the ACE space weather prediction data.

Parameters

fnames
[array-like] Series, list, or array of filenames.

tag
[str] Instrument tag, not used. (default='')

inst_id
[str] ACE instrument ID, not used. (default='')

Returns

data
[pandas.DataFrame] Object containing instrument data

meta
[pysat.Meta] Object containing metadata such as column names and units

See also:

pysat.instruments.methods.general.load_csv_data

4.2 Dst

The Disturbance Storm Time (Dst) Index is a measure of magnetic activity associated with the ring current. The National Geophysical Data Center (NGDC) maintains the current database from which the historic Dst is downloaded. LASP performs the calculates and provides the predicted Dst for the last 96 hours. You can learn more about the Dst Index at the WDC Kyoto Observatory page.

Supports Dst values. Downloads data from NGDC.

4.2.1 Properties

platform
'sw'

name
'dst'

tag
'noaa' - Historic Dst data coalated by and maintained by NOAA/NCEI 'lasp' - Predicted Dst from real-time ACE or DSCOVR provided by LASP

inst_id
''

4.2.2 Note

Will only work until 2057.

This material is based upon work supported by the National Science Foundation under Grant Number 1259508.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

`pysatSpaceWeather.instruments.sw_dst.`**`clean`**(*self*)

Clean the Dst index, empty function.

`pysatSpaceWeather.instruments.sw_dst.`**`download`**(*date_array*, *tag*, *inst_id*, *data_path*)

Download the Dst index data from the appropriate repository.

Parameters

date_array
[array-like or pandas.DatetimeIndex] Array-like or index of datetimes for which files will be downloaded.

tag
[str] Instrument tag, used to determine download location.

inst_id
[str] Instrument ID, not used.

data_path
[str] Path to data directory.

`pysatSpaceWeather.instruments.sw_dst.`**`init`**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.sw_dst.**list_files**(*tag=''*, *inst_id=''*, *data_path=''*, *format_str=None*)

List local data files for Dst data.

Parameters

tag
[str] Instrument tag, accepts any value from *tags*. (default='')

inst_id
[str] Instrument ID, not used. (default='')

data_path
[str] Path to data directory. (default='')

format_str
[str or NoneType] User specified file format. If None is specified, the default formats associated with the supplied tags are used. (default=None)

Returns

files
[pysat.Files] A class containing the verified available files

pysatSpaceWeather.instruments.sw_dst.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load the Dst index files.

Parameters

fnames
[pandas.Series] Series of filenames

tag
[str] Instrument tag string. (default='')

inst_id
[str] Instrument ID, not used. (default='')

Returns

data
[pandas.DataFrame] Object containing satellite data

pysat.Meta
Object containing metadata such as column names and units

4.3 $F_{10.7}$

$F_{10.7}$ is the 10.7 cm radio solar flux (measured in solar flux units, sfu) [Cortie 1912]. Historic indices, real-time indices, and forecasted indices are available from LASP and the SWPC F107 page.

Supports F10.7 index values. Downloads data from LASP and the SWPC.

4.3.1 Properties

platform
'sw'

name
'f107'

tag

- 'historic' LASP F10.7 data (downloads by month, loads by day)
- 'prelim' Preliminary SWPC daily solar indices
- 'daily' Daily SWPC solar indices (contains last 30 days)
- 'forecast' Grab forecast data from SWPC (next 3 days)
- '45day' 45-Day Forecast data from the Air Force

4.3.2 Examples

Download and load all of the historic F10.7 data. Note that it will not stop on the current date, but a point in the past when post-processing has been successfully completed.

```
f107 = pysat.Instrument('sw', 'f107', tag='historic')
f107.download(start=f107.lasp_stime, stop=f107.today(), freq='MS')
f107.load(date=f107.lasp_stime, end_date=f107.today())
```

4.3.3 Note

The forecast data is stored by generation date, where each file contains the forecast for the next three days. Forecast data downloads are only supported for the current day. When loading forecast data, the date specified with the load command is the date the forecast was generated. The data loaded will span three days. To always ensure you are loading the most recent data, load the data with tomorrow's date.

```
f107 = pysat.Instrument('sw', 'f107', tag='forecast')
f107.download()
f107.load(date=f107.tomorrow())
```

4.3.4 Warnings

The 'forecast' F10.7 data loads three days at a time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for 'forecast' data.

Like 'forecast', the '45day' forecast loads a specific period of time (45 days) and subsequent files contain overlapping data. Thus, loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for '45day' data.

pysatSpaceWeather.instruments.sw_f107.**clean**(*self*)

Clean the F10.7 data, empty function as this is not necessary.

pysatSpaceWeather.instruments.sw_f107.**download**(*date_array*, *tag*, *inst_id*, *data_path*, *update_files=False*)

Download F107 index data from the appropriate repository.

Parameters

date_array
[array-like] Sequence of dates for which files will be downloaded.

tag
[str] Denotes type of file to load.

inst_id
[str] Specifies the satellite ID for a constellation.

data_path
[str] Path to data directory.

update_files
[bool] Re-download data for files that already exist if True (default=False)

Raises

IOError
If a problem is encountered connecting to the gateway or retrieving data from the repository.

Warning: Only able to download current forecast data, not archived forecasts.

pysatSpaceWeather.instruments.sw_f107.**init**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.sw_f107.**list_files**(*tag=''*, *inst_id=''*, *data_path=''*, *format_str=None*)

List local F10.7 data files.

Parameters

tag
[str] Instrument tag, accepts any value from *tags*. (default='')

inst_id
[str] Instrument ID, not used. (default='')

data_path
[str] Path to data directory. (default='')

format_str
[str or NoneType] User specified file format. If None is specified, the default formats associated with the supplied tags are used. (default=None)

Returns

out_files
[pysat._files.Files] A class containing the verified available files

pysatSpaceWeather.instruments.sw_f107.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load F10.7 index files.

Parameters

fnames
[pandas.Series] Series of filenames.

tag
[str] Instrument tag. (default='')

inst_id
[str] Instrument ID, not used. (default='')

Returns

data
[pandas.DataFrame] Object containing satellite data.

meta
[pysat.Meta] Object containing metadata such as column names and units.

See also:

`pysat.instruments.methods.general.load_csv_data`

4.4 Kp

Kp is a geomagnetic index that reflects the magnitude of geomagnetic disturbances at Earth. Historic, recent (last 30 days), and forecasted values are available from the German Research Centre for Geosciences at Potsdam, GFZ, and the SWPC Kp page.

Supports Kp index values.

4.4.1 Properties

platform
'sw'

name
'kp'

tag

- '' Standard Kp data
- 'forecast' Grab forecast data from SWPC (next 3 days)
- 'recent' Grab last 30 days of Kp data from SWPC

inst_id
''

4.4.2 Note

Downloads data from ftp.gfz-potsdam.de or SWPC.

Standard Kp files are stored by the first day of each month. When downloading use kp.download(start, stop, freq='MS') to only download days that could possibly have data. 'MS' gives a monthly start frequency.

The forecast data is stored by generation date, where each file contains the forecast for the next three days. Forecast data downloads are only supported for the current day. When loading forecast data, the date specified with the load command is the date the forecast was generated. The data loaded will span three days. To always ensure you are loading the most recent data, load the data with tomorrow's date.

Recent data is also stored by the generation date from the SWPC. Each file contains 30 days of Kp measurements. The load date issued to pysat corresponds to the generation date.

4.4.3 Examples

```
kp = pysat.Instrument('sw', 'kp', tag='recent')
kp.download()
kp.load(date=kp.tomorrow())
```

4.4.4 Warnings

The 'forecast' and 'recent' tags load Kp data for a specific period of time. Loading multiple files, loading multiple days, the data padding feature, and multi_file_day feature available from the pyast.Instrument object is not appropriate for these tags data.

This material is based upon work supported by the National Science Foundation under Grant Number 1259508.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

pysatSpaceWeather.instruments.sw_kp.**clean**(*self*)

Clean the Kp, not required for this index (empty function).

pysatSpaceWeather.instruments.sw_kp.**download**(*date_array*, *tag*, *inst_id*, *data_path*)

Download the Kp index data from the appropriate repository.

Parameters

date_array
[array-like or pandas.DatetimeIndex] Array-like or index of datetimes to be downloaded.

tag
[str] Denotes type of file to load.

inst_id
[str] Specifies the instrument identification, not used.

data_path
[str] Path to data directory.

Raises

Exception
Bare raise upon FTP failure, facilitating useful error messages.

Warning: Only able to download current forecast data, not archived forecasts.

pysatSpaceWeather.instruments.sw_kp.**init**(*self*)

Initialize the Instrument object with instrument specific values.

pysatSpaceWeather.instruments.sw_kp.**list_files**(*tag=''*, *inst_id=''*, *data_path=''*, *format_str=None*)

List local files for the requested Instrument tag.

Parameters

tag
[str] Instrument tag, accepts any value from *tags*. (default='')

inst_id
[str] Instrument ID, not used. (default='')

data_path
[str] Path to data directory. (default=’’)

format_str
[str or NoneType] User specified file format. If None is specified, the default formats associated with the supplied tags are used. (default=None)

Returns

files
[pysat._files.Files] A class containing the verified available files

pysatSpaceWeather.instruments.sw_kp.**load**(*fnames*, *tag=''*, *inst_id=''*)

Load Kp index files.

Parameters

fnames
[pandas.Series] Series of filenames

tag
[str] Instrument tag (default=’’)

inst_id
[str] Instrument ID, not used. (default=’’)

Returns

data
[pandas.DataFrame] Object containing satellite data

meta
[pysat.Meta] Object containing metadata such as column names and units

CHAPTER

FIVE

METHODS

Several methods exist to help combine multiple data sets and convert between equivalent indices.

5.1 ACE

Supports the ACE instrument by providing reference and acknowledgement information.

Provides general routines for the ACE space weather instruments.

pysatSpaceWeather.instruments.methods.ace.**ace_swepam_hourly_omni_norm**(*as_inst*, *speed_key='sw_bulk_speed'*, *dens_key='sw_proton_dens'*, *temp_key='sw_ion_temp'*)

Normalize ACE SWEPAM variables as described in the OMNI processing _[1].

Parameters

as_inst
[pysat.Instrument] pysat Instrument object with ACE SWEPAM data.

speed_key
[str] Data key for bulk solar wind speed data in km/s (default='sw_bulk_speed')

dens_key
[str] Data key for solar wind proton density data in P/cm^3 (default='sw_proton_dens')

temp_key
[str] Data key for solar wind ion temperature data in K (default='sw_ion_temp')

References

[1] https://omniweb.gsfc.nasa.gov/html/omni_min_data.html

pysatSpaceWeather.instruments.methods.ace.**acknowledgements**()

Define the acknowledgements for the specified ACE instrument.

Returns

ackn
[str] Acknowledgements for the ACE instrument

pysatSpaceWeather.instruments.methods.ace.**clean**(*inst*)

Clean the common aspects of the ACE space weather data.

Parameters

inst
[pysat.Instrument] ACE pysat.Instrument object

Returns

max_status
[int] Maximum allowed status

pysatSpaceWeather.instruments.methods.ace.**common_metadata**()

Define the common metadata information for all ACE instruments.

Returns

meta
[pysat.Meta] pysat Meta class with 'jd' and 'sec' initiated

status_desc
[str] Description of the status flags

pysatSpaceWeather.instruments.methods.ace.**download**(*date_array*, *name*, *tag=''*, *inst_id=''*, *data_path=''*, *now=None*)

Download the requested ACE Space Weather data.

Parameters

date_array
[array-like] Array of datetime values

name
[str] ACE Instrument name.

tag
[str] ACE Instrument tag. (default='')

inst_id
[str] ACE instrument ID. (default='')

data_path
[str] Path to data directory. (default='')

now
[dt.datetime or NoneType] Current universal time, if None this is determined for each download. (default=None)

Warning:

- Only able to download current real-time data
- File requested not available on server

pysatSpaceWeather.instruments.methods.ace.**list_files**(*name*, *tag=''*, *inst_id=''*, *data_path=''*, *format_str=None*)

List the local ACE data files.

Parameters

name
[str] ACE Instrument name.

tag
[str] ACE Instrument tag. (default='')

inst_id
[str] Specifies the ACE instrument ID. (default=”)

data_path
[str] Path to data directory. (default=”)

format_str
[str or NoneType] User specified file format. If None is specified, the default formats associated with the supplied tags are used. (default=None)

Returns

files
[pysat.Files] A class containing the verified available files

pysatSpaceWeather.instruments.methods.ace.**load_csv_data**(*fnames*, *read_csv_kwargs=None*)

Load CSV data from a list of files into a single DataFrame.

Deprecated since version 0.0.5: *load_csv_data* will be removed in pysatSpaceWeather 0.0.6+, as it has been moved to *pysat.instruments.methods.general* as of pysat 3.0.1.

Parameters

fnames
[array-like] Series, list, or array of filenames

read_csv_kwargs
[dict or NoneType] Dict of kwargs to apply to *pds.read_csv*. (default=None)

Returns

data
[pds.DataFrame] Data frame with data from all files in the fnames list

See also:

pds.read_csv, pysat.instruments.methods.general.load_csv_data

pysatSpaceWeather.instruments.methods.ace.**references**(*name*)

Define the references for the specified ACE instrument.

Parameters

name
[str] Instrument name of the ACE instrument

Returns

ref
[str] Reference for the ACE instrument paper

5.2 Dst

Supports the Dst ring current index by providing reference and acknowledgement information.

Provides default routines for Dst.

pysatSpaceWeather.instruments.methods.dst.**acknowledgements**(*tag*)

Define the acknowledgements for the Dst data.

Parameters

tag
[str] Tag of the space weather index

Returns

ackn
[str] Acknowledgements string associated with the appropriate Dst tag.

pysatSpaceWeather.instruments.methods.dst.**references**(*tag*)

Define the references for the Dst data.

Parameters

tag
[string] Tag of the space weather index

Returns

refs
[str] Reference string associated with the appropriate Dst tag.

5.3 F $_{10.7}$

Supports the F $_{10.7}$ radio flux by providing reference and acknowledgement information as well as a routine to combine F $_{10.7}$ data obtained from multiple sources.

Routines for the F10.7 solar index.

pysatSpaceWeather.instruments.methods.f107.**acknowledgements**(*tag*)

Define the acknowledgements for the F10.7 data.

Parameters

tag
[str] Tag of the space waether index

Returns

ackn
[str] Acknowledgements string associated with the appropriate F10.7 tag.

pysatSpaceWeather.instruments.methods.f107.**calc_f107a**(*f107_inst*, *f107_name='f107'*, *f107a_name='f107a'*, *min_pnts=41*)

Calculate the 81 day mean F10.7.

Parameters

f107_inst
[pysat.Instrument] pysat Instrument holding the F10.7 data

f107_name
[str] Data column name for the F10.7 data (default='f107')

f107a_name
[str] Data column name for the F10.7a data (default='f107a')

min_pnts
[int] Minimum number of points required to calculate an average (default=41)

pysatSpaceWeather.instruments.methods.f107.**combine_f107**(*standard_inst*, *forecast_inst*, *start=None*, *stop=None*)

Combine the output from the measured and forecasted F10.7 sources.

Parameters

standard_inst
[pysat.Instrument or NoneType] Instrument object containing data for the 'sw' platform, 'f107' name, and 'historic', 'prelim', or 'daily' tag

forecast_inst
[pysat.Instrument or NoneType] Instrument object containing data for the 'sw' platform, 'f107' name, and 'prelim', '45day' or 'forecast' tag

start
[dt.datetime or NoneType] Starting time for combining data, or None to use earliest loaded date from the pysat Instruments (default=None)

stop
[dt.datetime or NoneType] Ending time for combining data, or None to use the latest loaded date from the pysat Instruments (default=None)

Returns

f107_inst
[pysat.Instrument] Instrument object containing F10.7 observations for the desired period of time, merging the standard, 45day, and forecasted values based on their reliability

Raises

ValueError
If appropriate time data is not supplied, or if the date range is badly formed.

Notes

Merging prioritizes the standard data, then the 45day data, and finally the forecast data

Will not attempt to download any missing data, but will load data

pysatSpaceWeather.instruments.methods.f107.**parse_45day_block**(*block_lines*)

Parse the data blocks used in the 45-day Ap and F10.7 Flux Forecast file.

Parameters

block_lines
[list] List of lines containing data in this data block

Returns

dates
[list] List of dates for each date/data pair in this block

values
[list] List of values for each date/data pair in this block

pysatSpaceWeather.instruments.methods.f107.**parse_daily_solar_data**(*data_lines*, *year*, *optical*)

Parse the data in the SWPC daily solar index file.

Parameters

data_lines
[list] List of lines containing data

year
[list] Year of file

optical
[boolean] Flag denoting whether or not optical data is available

Returns

dates
[list] List of dates for each date/data pair in this block

values
[dict] Dict of lists of values, where each key is the value name

pysatSpaceWeather.instruments.methods.f107.**references**(*tag*)

Define the references for the F10.7 data.

Parameters

tag
[str] Instrument tag for the F10.7 data.

Returns

refs
[str] Reference string associated with the appropriate F10.7 tag.

pysatSpaceWeather.instruments.methods.f107.**rewrite_daily_file**(*year*, *outfile*, *lines*)

Rewrite the SWPC Daily Solar Data files.

Parameters

year
[int] Year of data file (format changes based on date)

outfile
[str] Output filename

lines
[str] String containing all output data (result of 'read')

5.4 Kp and Ap

Supports the Kp instrument by providing reference and acknowledgement information, a routine to combine Kp from multiple sources, routines to convert between Kp and Ap, and a routine that uses Kp data as a geomagnetic activity filter for other data sets.

Provides routines to support the geomagnetic indeces, Kp and Ap.

pysatSpaceWeather.instruments.methods.kp_ap.**acknowledgements**(*name*, *tag*)

Define the acknowledgements for the geomagnetic data sets.

Parameters

name
[str] Instrument name of space weather index, accepts 'kp' or 'ap'.

tag
[str] Instrument tag.

pysatSpaceWeather.instruments.methods.kp_ap.**calc_daily_Ap**(*ap_inst*, *ap_name='3hr_ap'*, *daily_name='Ap'*, *running_name=None*)

Calculate the daily Ap index from the 3hr ap index.

Parameters

ap_inst
[pysat.Instrument] pysat instrument containing 3-hourly ap data

ap_name
[str] Column name for 3-hourly ap data (default='3hr_ap')

daily_name
[str] Column name for daily Ap data (default='Ap')

running_name
[str or NoneType] Column name for daily running average of ap, not output if None (default=None)

Raises

ValueError
If *ap_name* or *daily_name* aren't present in *ap_inst*

pysatSpaceWeather.instruments.methods.kp_ap.**combine_kp**(*standard_inst=None*, *recent_inst=None*, *forecast_inst=None*, *start=None*, *stop=None*, *fill_val=nan*)

Combine the output from the different Kp sources for a range of dates.

Parameters

standard_inst
[pysat.Instrument or NoneType] Instrument object containing data for the 'sw' platform, 'kp' name, and '' tag or None to exclude (default=None)

recent_inst
[pysat.Instrument or NoneType] Instrument object containing data for the 'sw' platform, 'kp' name, and 'recent' tag or None to exclude (default=None)

forecast_inst
[pysat.Instrument or NoneType] Instrument object containing data for the 'sw' platform, 'kp' name, and 'forecast' tag or None to exclude (default=None)

start
[dt.datetime or NoneType] Starting time for combining data, or None to use earliest loaded date from the pysat Instruments (default=None)

stop
[dt.datetime] Ending time for combining data, or None to use the latest loaded date from the pysat Instruments (default=None)

fill_val
[int or float] Desired fill value (since the standard instrument fill value differs from the other sources) (default=np.nan)

Returns

kp_inst
[pysat.Instrument] Instrument object containing Kp observations for the desired period of time, merging the standard, recent, and forecasted values based on their reliability

Raises

ValueError
If only one Kp instrument or bad times are provided

pysatSpaceWeather.instruments.methods.kp_ap.**convert_3hr_kp_to_ap**(*kp_inst*, *var_name='Kp'*)

Calculate 3 hour ap from 3 hour Kp index.

Parameters

kp_inst
[pysat.Instrument] Instrument containing Kp data

var_name
[str] Variable name for the Kp data (default='Kp')

Raises

ValueError
If *var_name* is not present in *kp_inst*.

pysatSpaceWeather.instruments.methods.kp_ap.**convert_ap_to_kp**(*ap_data*, *fill_val=- 1*, *ap_name='ap'*, *kp_name='Kp'*)

Convert Ap into Kp.

Parameters

ap_data
[array-like] Array-like object containing Ap data

fill_val
[int, float, NoneType] Fill value for the data set (default=-1)

ap_name
[str] Name of the input ap (default='ap')

kp_name
[str] Name of the output Kp (default='Kp')

Returns

kp_data
[array-like] Array-like object containing Kp data

meta
[Metadata] Metadata object containing information about transformed data

pysatSpaceWeather.instruments.methods.kp_ap.**filter_geomag**(*inst*, *min_kp=0*, *max_kp=9*, *filter_time=24*, *kp_inst=None*, *var_name='Kp'*)

Filter pysat.Instrument data for given time after Kp drops below gate.

Parameters

inst
[pysat.Instrument] Instrument with non-Kp data to be filtered by geomagnetic activity

min_kp
[float] Minimum Kp value allowed. Kp values below this filter the data in inst (default=0)

max_kp
[float] Maximum Kp value allowed. Kp values above this filter the data in inst (default=9)

filter_time
[int] Number of hours to filter data after Kp drops below max_kp (default=24)

kp_inst
[pysat.Instrument or NoneType] Kp pysat.Instrument object with or without data already loaded. If None, will load GFZ historic kp data for the instrument date (default=None)

var_name
[str] String providing the variable name for the Kp data (default='Kp')

pysatSpaceWeather.instruments.methods.kp_ap.**initialize_kp_metadata**(*meta*, *data_key*, *fill_val=- 1*)

Initialize the Kp meta data using our knowledge of the index.

Parameters

meta
[pysat._meta.Meta] Pysat Metadata

data_key
[str] String denoting the data key

fill_val
[int or float] File-specific fill value (default=-1)

pysatSpaceWeather.instruments.methods.kp_ap.**references**(*name*, *tag*)

Define the references for the geomagnetic data sets.

Parameters

name
[str] Instrument name of space weather index, accepts kp or ap.

tag
[str] Instrument tag.

pysatSpaceWeather.instruments.methods.kp_ap.**round_ap**(*ap_in*, *fill_val=nan*)

Round an ap value to the nearest Kp value.

Parameters

ap_in
[float] Ap value as a floating point number.

fill_val
[float] Value for unassigned or bad indices. (default=np.nan)

Returns

float
Fill value for infinite or out-of-range data, otherwise the best kp index is provided.

CHAPTER

SIX

EXAMPLES

Here are some examples that demonstrate how to use various pysatSpaceWeather tools

6.1 Loading $F_{10.7}$

pysatSpaceWeather uses pysat to load different historical, real-time, and forecasted solar, geomagnetic, and other space weather indices. As specified in the pysat tutorial, data may be loaded using the following commands. Historic $F_{10.7}$ is used as an example.

```
import pysat
import pysatSpaceWeather as py_sw

f107 = pysat.Instrument(inst_module=py_sw.instruments.sw_f107,
                        tag='historic', update_files=True)
f107.download(start=f107.lasp_stime, stop=f107.today(), freq='MS')
f107.load()
print(f107)
```

The output includes all available historic data (as implied by the tag name), including the specified date. This data set starts on 14 February 1947, as indicated by the special instrument attribute *f107.lasp_stime*, and will not reach up to the present day. At the time of publication this produces the output shown below (the index header has been added here for clarity).

```
<Index>      f107
1947-02-14  253.9
1947-02-17  228.5
1947-02-19  178.8
1947-02-20  163.7
1947-02-24  164.1
...           ...
2018-04-27   69.6
2018-04-28   71.1
2018-04-29   72.2
2018-04-30   71.3

[25367 rows x 1 columns]
```

CHAPTER

SEVEN

GUIDE FOR DEVELOPERS

7.1 Contributor Covenant Code of Conduct

7.1.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

7.1.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

7.1.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

7.1.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

7.1.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at pysat.developers@gmail.com. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

7.1.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://contributor-covenant.org/version/1/4

7.2 Contributing

Bug reports, feature suggestions and other contributions are greatly appreciated! pysat and pysatSpaceWeather are community-driven projects and welcomes both feedback and contributions.

Come join us on Slack! An invitation to the pysat workspace is available in the 'About' section of the pysat GitHub Repository. Development meetings are generally held fortnightly.

7.2.1 Short version

- Submit bug reports, feature requests, and questions at GitHub
- Make pull requests to the `develop` branch

7.2.2 Bug reports

When reporting a bug please include:

- Your operating system name and version
- Any details about your local setup that might be helpful in troubleshooting
- Detailed steps to reproduce the bug

7.2.3 Feature requests and feedback

The best way to send feedback is to file an issue at GitHub.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

7.2.4 Development

To set up `pysatSpaceWeather` for local development:

1. **`Fork pysatSpaceWeather on GitHub <https://github.com/pysat/pysatSpaceWeather/fork>>`_.**
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/pysatSpaceWeather.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally. Tests for new instruments are performed automatically. See discussion here for more information on triggering these standard tests.

   Tests for custom functions should be added to the appropriately named file in `pysatSpaceWeather/tests`. For example, space weather methods should be named `pysatSpaceWeather/tests/test_methods_sw.py`. If no test file exists, then you should create one. This testing uses pytest, which will run tests on any python file in the test directory that starts with `test`. Classes must begin with `Test`, and test methods must also begin with `test`.

4. When you're done making changes, run all the checks to ensure that nothing is broken on your local system:

```
pytest -vs pysatSpaceWeather
```

5. Update/add documentation (in `docs`). Even if you don't think it's relevant, check to see if any existing examples have changed.
6. Add your name to the .zenodo.json file as an author
7. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "CODE: Brief description of your changes"
git push origin name-of-your-bugfix-or-feature
```

Where CODE is a standard shorthand for the type of change (eg, BUG or DOC). `pysat` follows the numpy development workflow, see the discussion there for a full list of this shorthand notation.

8. Submit a pull request through the GitHub website. Pull requests should be made to the `develop` branch.

7.2.5 Pull Request Guidelines

If you need some code review or feedback while you're developing the code, just make a pull request. Pull requests should be made to the `develop` branch.

For merging, you should:

1. Include an example for use
2. Add a note to `CHANGELOG.md` about the changes
3. Ensure that all checks passed (current checks include GitHub Actions, Coveralls, and ReadTheDocs)[1]

7.2.6 Project Style Guidelines

In general, pysat follows PEP8 and numpydoc guidelines. Pytest runs the unit and integration tests, flake8 checks for style, and sphinx-build performs documentation tests. However, there are certain additional style elements that have been settled on to ensure the project maintains a consistent coding style. These include:

- Line breaks should occur before a binary operator (ignoring flake8 W503)
- Combine long strings using `join`
- Preferably break long lines on open parentheses rather than using \
- Use no more than 80 characters per line
- Avoid using Instrument class key attribute names as unrelated variable names: `platform`, `name`, `tag`, and `inst_id`
- The pysat logger is imported into each sub-module and provides status updates at the info and warning levels (as appropriate)
- Several dependent packages have common nicknames, including:
  - `import datetime as dt`
  - `import numpy as np`
  - `import pandas as pds`
  - `import xarray as xr`
- When incrementing a timestamp, use `dt.timedelta` instead of `pds.DateOffset` when possible to reduce program runtime
- All classes should have `__repr__` and `__str__` functions
- Docstrings use `Note` instead of `Notes`
- Try to avoid creating a try/except statement where except passes
- Use setup and teardown in test classes
- Use pytest parametrize in test classes when appropriate

[1] If you don't have all the necessary Python versions available locally or have trouble building all the testing environments, you can rely on Travis to run the tests for each change you add in the pull request. Because testing here will delay tests by other developers, please ensure that the code passes all tests on your local system first.

- Provide testing class methods with informative failure statements and descriptive, one-line docstrings
- Block and inline comments should use proper English grammar and punctuation with the exception of single sentences in a block, which may then ommit the final period

CHAPTER

EIGHT

CHANGE LOG

All notable changes to this project will be documented in this file. This project adheres to Semantic Versioning.

8.1 [0.0.6] - 2022-07-08

- Updated `sw_f107` to reflect changes in the `historic` data file format

8.2 [0.0.5] - 2022-06-10

- Updated the docstrings to conform to pysat standards
- Added docstring tests to Flake8 portion of CI testing
- Fixed bug in `combine_kp` that occurs if no times are provided
- Improved unit test style and expanded unit test coverage
- Updated package organization documentation
- Added a function to normalize ACE SWEPAM variables as described in the OMNI processing guide
- Deprecated `load_csv_data` method, which was moved to pysat
- Added the LASP predicted Dst to the Dst Instrument
- Updated pandas usage to remove existing deprecation warnings
- Updated `pysat.Instrument.load` calls to remove `use_header` deprecation warning

8.3 [0.0.4] - 2021-05-19

- New Logo
- Implements GitHub Actions for primary CI testing
- Updated tested python versions
- Removed non-document testing from Travis-CI and updated installation method
- Updated redirected links
- Improved PEP8 compliance
- Separated ACE instrument into four, following standard pysat practice of grouping satellite instruments using the satellite mission as the platform.

- Made standard Kp loading more robust
- Fixed bugs where current time used local time zone instead of UTC
- Added PyPi links to README and documentation
- Deprecated F10.7 instrument tag 'all' and '', replacing them with 'historic'
- Improved F10.7 instrument routines by combining similar code blocks
- Fixed F10.7 load/list bugs that lead to duplicate data entries
- Fixed Dst load bugs when loading multiple days of data
- Replaced Dst tag '' with 'noaa', making it easy to add other sources
- Moved all instrument support routines to appropriate `methods` sub-module

8.4 [0.0.3] - 2021-01-15

- Fixes bugs in configuration and zenodo files
- Added pysat version restriction to requirements.txt

8.5 [0.0.2] - 2021-01-11

- Added real-time ACE instrument
- Fixed bugs and restructured code to comply with changes in pysat-3.0
- Added user documentation
- Simplified setup and testing environments
- Separated space weather methods into sub-modules by instrument

8.6 [0.0.1] - 2020-08-13

- Initial port of existing routines from pysat

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

L

M

P

R