



A Python package for InSAR time series analysis

PySAR Documentation

Version 1.3.0

Zhang Yunjun, Heresh Fattahi

2018 March

0 Introduction

PySAR is an open-source Python package for InSAR (Interferometric Synthetic Aperture Radar) time series analysis. It reads stack of interferograms (coregistered and unwrapped) and produces three-dimensional (2D in space and 1D in time) ground displacement. It includes a routine time series analysis (`pysarApp.py`) and some independent toolboxes. PySAR is built on the initial work done by Scott Baker. Alfredo Terrero developed the code to prepare HDF-EOS5 time-series product for web viewer.

PySAR is available on GitHub: <https://yunjunz.github.io/PySAR/>

When using this software please reference Yunjun et al. [2018]:
Yunjun Z., H. Fattahi, F. Amelung, (2018), PySAR – A Python Package for Generic InSAR Time Series Analysis based on Full Rank Network. (In prep.)

We also encourage users to cite the individual paper for the steps used in your processing. Details are included in the help of each individual script and summed up in the Bibliography chapter.

This manual provides a brief description to running PySAR, but does not explain all the processing. For technical details, please regards to the cited paper, or code. A detailed API description is attached at the end with index. This is for the developers who would like to develop your own processing routine, or other code where PySAR serves as a platform or toolbox. Contributions are welcomed and can be submitted to our Github repository.

Contents

1	Welcome to PySAR!	2
2	_Sidebar	4
3	Attributes	5
4	Bibliography	6
5	Coordinate	6
6	Corrected DEM	7
7	Documentation-Generation	7
8	Example	7
9	File-Descriptions	8
10	Gamma-File-Decription	9
11	Google-Earth	10
12	Home	10
13	SAR-Sensor-Parameter	11
14	UNAVCO-InSAR-Archive	11
15	Web-Viewer	12
16	Namespace Index	12
17	Hierarchical Index	15
18	Class Index	15
19	File Index	16
20	Namespace Documentation	19

21 Class Documentation	234
22 File Documentation	260
Index	305

1 Welcome to PySAR!

PySAR is a open-source Python package for InSAR (Interferometric Synthetic Aperture Radar) time series analysis. It reads stack of interferograms (coregistered and unwrapped) in ROI_PAC, Gamma and ISCE format, and produces three dimensional (2D in space and 1D in time) ground displacement. It includes a routine time series analysis ([pysarApp.py](#)) and some independent toolboxes. PySAR is built on the initial work done by [Scott Baker](#). [Alfredo Terrero](#) developed the code to prepare UNAVCO InSAR product for [time series web viewer](#).

We recommend using [Anaconda](#) to install the python environment and the prerequisite packages. You will need:

- [Python2.7](#)

- Numpy
- Scipy
- h5py
- Matplotlib
- Basemap (optional, for plotting in geo coordinate)
- pykml (optional, for Google Earth KMZ file output)
- joblib (optional, for parallel processing)
- [PyAPS](#) (optional, for tropospheric correction using weather re-analysis models, i.e. ERA-Interim, NARR, MERRA)

Here is a example on Mac OSX using csh/tcsh:
Add the following in `~/.cshrc` file and source it.

```
##### Python #####
setenv PYTHON2DIR ~/python/anaconda2
setenv PATH ${PYTHON2DIR}/bin:${PATH}
```

Install Xcode with command line tools, follow instructions [here](#).

Then run the following in your terminal:

```
cd ~/python
wget https://repo.continuum.io/archive/Anaconda2-4.4.0-MacOSX-x86_64.sh
chmod +x Anaconda2-4.4.0-MacOSX-x86_64.sh
./Anaconda2-4.2.0-MacOSX-x86_64.sh -b -p $PYTHON2DIR
$PYTHON2DIR/bin/conda config --add channels conda-forge
$PYTHON2DIR/bin/conda install basemap joblib pykml --yes
```

For PyAPS installation, please refer to [PyAPS's Wiki at Caltech](#)

1.2 PySAR

Download the latest released version at <https://github.com/yunjunz/PySAR/releases>, or use the command below.

```
cd ~/python
wget https://github.com/yunjunz/PySAR/archive/v1.2.0.tar.gz
tar -zxvf v1.2.0.tar.gz
```

or download the development version using git:

```
cd ~/python
git clone https://github.com/yunjunz/PySAR.git
```

To use the package, you need to setup the environment. Depending on your shell, you may use commands below to setup pysar, by adding the following to your source file. They are for:

1. To make pysar importable in python, by adding the path to PySAR directory to your \$PYTHONPATH
2. To make utility scripts available in command line, by adding \${PYSAR_HOME}/pysar and \${PYSAR_HOME}/shellscrip↵ts to your \$path.

For bash user, add to your .bashrc file:

```
if [ -z ${PYTHONPATH+x} ]; then export PYTHONPATH=""; fi
export PYSAR_HOME=~/.python/PySAR          #for released version, "~/.python/PySAR-1.2.0"
export PYTHONPATH=${PYSAR_HOME}:${PYTHONPATH}
export PATH=${PYSAR_HOME}/pysar:${PYSAR_HOME}/shellscrip↵ts:${PATH}
```

For csh/tcsh user, add to your .cshrc file:

```
if ( ! ${PYTHONPATH} ) then
    setenv PYTHONPATH ""
endif
setenv PYSAR_HOME ~/.python/PySAR          #for released version, "~/.python/PySAR-1.2.0"
setenv PYTHONPATH ${PYSAR_HOME}:${PYTHONPATH}
setenv PATH       ${PYSAR_HOME}/pysar:${PYSAR_HOME}/shellscrip↵ts:${PATH}
```

The current version is compatible with ROI_PAC and Gamma products. PySAR reads unwrapped interefrograms (at the same coordinate system: radar or geo) and the baseline files for each interefrogram. You need to give the path to where the interferograms are and PySAR takes care of the rest!

Run [pysarApp.py -h](#) see the processing options.

Run [pysarApp.py -g](#) to generate a default template file and see the detailed settings.

Download the test data: [Download Link](#) and unzip it.
Create a custom template file:

```
cd ~/KujuAlosAT422F650/PYSAR
vi KujuAlosAT422F650_template.txt
```

Include the following pysar options in your template:

```
##----- Data Loading -----
# RADAR COORD ROIPAC PRODUCTS
pysar.unwrapFiles      = ~/KujuAlosAT422F650/ROIPAC/RADAR/filt_*.unw
pysar.corFiles          = ~/KujuAlosAT422F650/ROIPAC/RADAR/filt_*.cor
pysar.geomap            = ~/KujuAlosAT422F650/ROIPAC/RADAR/geomap*.trans
pysar.dem.radarCoord    = ~/KujuAlosAT422F650/ROIPAC/RADAR/radar*.hgt
pysar.dem.geoCoord      = ~/KujuAlosAT422F650/ROIPAC/RADAR/*.dem
```

```
pysar.reference.lalo    = 33.0655, 131.2076
pysar.deramp            = plane
```

Save your template file and run PySAR as:

```
pysarApp.py KujuAlosAT422F650_template.txt
```

Inside [pysarApp.py](#), it reads the unwrapped interferograms, references all of them to the same coherent pixel (a seed point point), calculates the phase closure and estimates the unwrapping errors (if it has been asked for), inverts the interferograms, calculates a parameter called "temporal_coherence" which can be used to evaluate the quality of inversion, removes ramps or surface from time-series epochs, corrects dem errors, corrects local oscillator drift (for Envisat only), corrects stratified tropospheric delay (using pyaps and using phase-elevation approach), ... and finally estimates the velocity.

Use [view.py](#) to view any pysar output.

Use [tsviewer.py](#) to plot the time-series for each point (relative to the reference point and epoch!).

PySAR is a toolbox with a lot of individual utility scripts, highly modulized in python. Check its documentaion or simple run it with -h to see its usage, you could build your own customized processing recipe!

- Manual: [PDF](#), [HTML](#)
- Wiki: Check our [Github Wiki](#) to see the example data, paper references, file naming convention and more.

Join our google group <https://groups.google.com/forum/#!forum/py-sar> to ask questions, get notice of latest features pushed to you!

2 _Sidebar

Wiki

- [Home](#)
- [pysarApp](#)
- [Example](#)
- [File Description](#)
- [Attributes](#)
- [Coordinate](#)
- [DEM](#)
- [Bibliography](#)

Output

- [Google Earth](#)
- [UNAVCO](#)
- [Web Viewer](#)

3 Attributes

PySAR mainly use attribute name from ROI_PAC, with some additional attributes generated by PySAR itself.

If using ROI_PAC as InSAR processor, both "baseline parameter RSC" file (i.e. *100416-100901_baseline.rsc*) and basic metadata file (i.e. *filt_100416-100901-sim_HDR_4rlks_c10.unw.rsc*) will be imported into PySAR. The following attributes for each interferogram are required in order to run PySAR:

- + FILE_LENGTH = number of rows
- + WIDTH = number of columns
- + X/Y_STEP = Ground resolution in degree in Longitude/latitude direction, for geocoded product
- + X/Y_FIRST = Longitude/latitude in degree of the first pixel - Upper left corner, for geocoded product

- LAT/LON_REF1/2/3/4 = Latitude/longitude at corner 1/2/3/4 (degree), used in save_unavco, PyAPS (DEM file in radar coord), not accurate; number named in order of first line near/far range, last line near/far range
 - + WAVELENGTH = Radar wavelength (m)
 - + RANGE_PIXEL_SIZE = Slant range pixel size (search for pixel_ratio to convert to ground size, in m), used in dem_error, incidence_angle, multilook, transect.
 - + EARTH_RADIUS = Best fitting spheroid radius (m), used in dem_error, incidence_angle, convert2mat
 - + CENTER_LINE.UTC = Time at middle of interferogram (seconds), used in tropo correction using PyAPS
 - + HEIGHT = Height of satellite (m), used in dem_error, incidence_angle, convert2mat
 - + STARTING_RANGE = Distance from satellite to first ground pixel (m), used in incidence_angle calculation
 - + DATE12 = (date1)-(date2), master - slave date of interferogram in 6 digit number
 - + PLATFORM = satellite/sensor name, used in Local Oscillator Drift correction for Envisat
 - + ORBIT_DIRECTION = ascending, or descending
 - + P_BASELINE_TOP_HDR = Perpendicular baseline at top (first line) of interferogram (m), used in _network, _pysar_utilities
 - + P_BASELINE_BOTTOM_HDR = Perpendicular baseline at bottom (last line) of interferogram (m), used in _network, _pysar_utilities
 - + ALOOKS/LOOKS = multilook number in azimuth/range direction, used in weighted network inversion.

- ANTENNA_SIDE = -1 for right looking radar, used in save_unavco

- AZIMUTH_PIXEL_SIZE = Azimuth pixel size at orbital altitude (multiply by $R_e/(R_e+h)$ for ground size (m), where R_e is the local earth radius), used in baseline_error/trop and multilook.
 - + HEADING = Spacecraft heading at peg point (degree), used in asc_desc, los2enu
 - + LOOK_REF1/2 = Look angle at corner 1/2 (degree), not accurate (optional)
 - + PRF = Pulse repetition frequency (Hz), used in save_unavco
 - + H_BASELINE_RATE_HDR = Rate of change of horizontal baseline as a function of line number (linear term), used in _pysar_utilities
 - + H_BASELINE_TOP_HDR = Horizontal baseline separation at the top of the interferogram calculated from orbital parameters, used in _pysar_utilities
 - + V_BASELINE_RATE_HDR = Linear term for vertical baseline change, used in _pysar_utilities
 - + V_BASELINE_TOP_HDR = Vertical baseline separation at top of the interferogram, used in _pysar_utilities

- FILE_TYPE = file type, velocity, timeseries, interferograms, etc.; for non-HDF5 file, it's the file extension name.
 - + FILE_PATH = absolute file path
 - + INSAR_PROCESSOR = InSAR processor, roipac, gamma, isce, etc.
 - + PROCESSOR = processing software, i.e. isce, roipac, gamma

- DATA_TYPE = data type, i.e. float32, int16, etc., for isce product read using GDAL

- P_BASELINE_TIMESERIES = timeseries of perpendicular baseline

- P_BASELINE_TOP_TIMESERIES = timeseries of perpendicular baseline at top of interferogram

- `P_BASELINE_BOTTOM_TIMESERIES` = timeseries of perpendicular baseline at bottom of interferogram
- `UNIT` = data unit, i.e. m, m/yr, radian, and 1 for file without unit, such as coherence
- `date1` = start time of dataset
- `date2` = end time of dataset
- `drop_date` =
- `drop_ifgram` = yes or no, drop this interferogram or not for `unwrapIfgram.h5`, `coherence.h5` etc.
- `ref_date` = reference date
- `ref_x/y/lat/lon` = column/row/latitude/longitude of reference point
- `subest_x0/y0/x1/y1` = start/end column/row number of subset in the original coverage

Pritchard et al., (2014), Open-source software for geodetic imaging: ROI_PAC for InSAR and pixel tracking, pp 44-48. [PDF](#)

4 Bibliography

Berardino, P., G. Fornaro, R. Lanari, and E. Sansosti (2002), A new algorithm for surface deformation monitoring based on small baseline differential SAR interferograms, *Geoscience and Remote Sensing, IEEE Transactions on*, 40(11), 2375-2383, doi:[10.1109/TGRS.2002.803792](#).

Doin, M. P., C. Lasserre, G. Peltzer, O. Cavalié, and C. Doubre (2009), Corrections of stratified tropospheric delays in SAR interferometry: Validation with global atmospheric models, *Journal of Applied Geophysics*, 69(1), 35-50, doi:[10.1016/j.jappgeo.2009.03.010](#).

Fattahi, H., and F. Amelung (2013), DEM Error Correction in InSAR Time Series, *Geoscience and Remote Sensing, IEEE Transactions on*, 51(7), 4249-4259, doi:[10.1109/TGRS.2012.2227761](#).

Fattahi, H., and F. Amelung (2014), InSAR uncertainty due to orbital errors, *Geophysical Journal International*, 199(1), 549-560, doi:[10.1093/gji/ggu276](#).

Fattahi, H., and F. Amelung (2015), InSAR bias and uncertainty due to the systematic and stochastic tropospheric delay, *Journal of Geophysical Research: Solid Earth* (120), doi:[10.1002/2015JB012419](#).

Jolivet, R., R. Grandin, C. Lasserre, M. P. Doin, and G. Peltzer (2011), Systematic InSAR tropospheric phase delay corrections from global meteorological reanalysis data, *Geophysical Research Letters*, 38(17), L17311, doi:[10.1029/2011GL048757](#).

Tizzani, P., P. Berardino, F. Casu, P. Euillades, M. Manzo, G. P. Ricciardi, G. Zeni, and R. Lanari (2007), Surface deformation of Long Valley caldera and Mono Basin, California, investigated with the SBAS-InSAR approach, *Remote Sensing of Environment*, 108(3), 277-289, doi:[10.1016/j.rse.2006.11.015](#).

5 Coordinate

There are two coordination systems in PySAR: **radar coordinate** and **geo coordinate**. Geo coordinate is defined in WGS84 coordination for horizontal direction, and determined by the following `ROI_PAC` `attributes` in latitude and longitude. The following shows examples from `AlosAT422F650/geo_velocity.h5`:

```
X_FIRST    131.02409876
Y_FIRST    33.63756779
X_STEP     0.00033333
Y_STEP     -0.00033333
X_UNIT     degrees
Y_UNIT     degrees
```

X/Y_FIRST are the longitude/latitude value of the first (upper left corner) pixel's upper left corner, as shown below:

6 Corrected DEM

PySAR estimates DEM residual in time series domain using Fattahi and Amelung's method (2013, TGRS), and output a estimated DEM residual value for each pixel into file demRadar/demGeo_error.h5. It can be used to generate a new, corrected DEM after masking and proper decamping, using the command below.

```
mask.py demRadar_error.h5 -m maskTempCoh.h5
add.py demRadar_error_masked.h5 demRadar.h5 demRadar_cor.h5
```

To better under this correction approach, please keep in mind:

1. InSAR measures relative range distance, so does this step. Thus, this estimated DEM residual is with respect to the reference pixel, which has zero value.

1. Fattahi and Amelung (2013, TGRS) method estimate phase components correlated with perpendicular baseline history, which should mainly be DEM residual; it also contains the correlated part from temporal deformation or orbit error, if they are correlated, or partial correlated with perpendicular baseline history.

Fattahi, H., and F. Amelung (2013), DEM Error Correction in InSAR Time Series, *Geoscience and Remote Sensing, IEEE Transactions on*, 51(7), 4249-4259, doi:[10.1109/TGRS.2012.2227761](https://doi.org/10.1109/TGRS.2012.2227761).

7 Documentation-Generation

We use Doxygen to generate the API documentation automatically.

Install Doxygen following [link](#).

8 Example

Here is some demo dataset for testing, just download and unzip it, and run the command below:

```
cd $PROJECT_NAME/PYSAR
pysarApp.py $PROJECT_NAME_template.txt
```

- **Kuju Volcano with ALOS Asc Track 422 Frame 650** - [Link to Download Data](#)

```
cd KujuAlosAT422F650/PYSAR pysarApp.py KujuAlosAT422F650_template.txt
```

[pysarApp.py](#) reads this custom template and update the corresponding options in the default full template file `**_pysarApp_template.txt**` (generated automatically by PySAR whenever you run [pysarApp.py](#); if it already exists, the existing one will be used). Then it runs through the whole processing chain step by step. You can modify the parameters, method for tropospheric delay correction etc. by changing the option value in `pysarApp_template.txt`, and re-run [pysarApp.py](#) to update your processing; and re-do this on and on until you are happy with your result. If no `pysarApp_template.txt` found, you can generate it with default values using the command below, all options are documented inside the file:

```
pysarApp.py -g
```

To re-run the program:

```
pysarApp.py
```

Check the auto plotted figures in **PYSAR/PIC** folder, and modify the plotting parameters to adjust your plotting, and re-run the plotting script to update your plotting result, a modified version is attached in the example data link above:

```
./plot_pysarApp.sh
```

By default, it's using ROI_PAC product in radar coordinate in ROIPAC/RADAR folder; if you want to try PySAR with geo coordinate ROI_PAC product in ROIPAC/GEO folder, edit "Data Loading" part in template file: comment out the "RADAR COORD ROIPAC PRODUCTS" part and un-comment "GEO COORD ROIPAC PRODUCTS" part, and re-run [pysarApp.py](#)

Yunjun, Z., Amelung F., Aoki Y., (2016). Poster: A time series InSAR survey of volcanic deformation in Kyushu, SW Japan with JERS and ALOS data (G51B-1113). AGU Fall Meeting 2016, Dec 12-16, 2016, San Francisco, CA, USA. [PDF](#)

9 File-Descriptions

PySAR use HDF5 file internally. It loads ROI_PAC file into .h5 file in the beginning and has the capability to output to UNAVCO hdf5 file, .grd file, ROI_PAC file and Google Earth KMZ file.

There are 3 types of HDF5 file structures used in PySAR:

- multi_group (**Ngroup-1dset-1atr**) = multiple groups with one dataset and one attribute dict per group
i.e. interferograms, coherence, wrapped, snaphu_connect_component

- multi_dataset (**1group-Ndset-1atr**) = one group with multiple dataset and one attribute dict per group
i.e. timeseries, geometry
- single_dataset (**1group-1dset-1atr**) = one group with one dataset and one attribute dict per group
i.e. velocity, dem, rmse, temporal_coherence, mask
- coherence.h5 = spatial coherence files loaded from ROI_PAC, generated in load_data step
- snaphuConnectComponent.h5 = multi_group type, mask of connect component files from SNAPHU phase unwrapping, loaded from ROI_PAC, generated in load_data step
- wrapIfgram.h5 = wrapped interferograms loaded from ROI_PAC, generated in load_data step
- unwrapIfgram.h5 = unwrapped interferograms loaded from ROI_PAC, generated in load_data step
- timeseries.h5 = multi_dataset type, time series displacement, generated in network inversion step
 - geometryRadar/Geo.h5 = multi_dataset type, geometry file for dataset, including the following info:
 - /geometry/latitude/ #for geometryRadar.h5 only, from ISCE/Doris lookup table
 - /geometry/longitude/ #for geometryRadar.h5 only, from ISCE/Doris lookup table
 - /geometry/rangeCoord/ #for geometryGeo.h5 only, from ROI_PAC/Gamma lookup table
 - /geometry/azimuthCoord/ #for geometryGeo.h5 only, from ROI_PAC/Gamma lookup table
 - /geometry/height/
 - /geometry/incidenceAngle/
 - /geometry/headingAngle/
 - /geometry/slantRangeDistance/

- averageSpatialCoherence.h5 = temporal mean of all spatial coherence, generated from coherence.h5 in data loading step
- demGeo.h5 = DEM in geo coordinate, loaded from pysar.dem.geoCoord
- demRadar.h5 = DEM in radar coordinate, loaded from pysar.dem.radarCoord
- mask.h5 = mask of non-zero amplitude pixels, generated from .unw file list in data loading step
- maskTempCoh.h5 = mask of high temporal coherent pixels, generated from temporalCoherence.h5 with threshold (default=0.7)
- temporalCoherence.h5 = temporal coherence, generated from the inversion of network of interferograms to timeseries
- velocity.h5 = Line-Of-Sight (LOS) velocity, generated in time series inversion step
 - velocityRmse.h5 = root-mean-square deviation of Mean LOS velocity estimation
- velocityStd.h5 = standard deviation of Mean LOS velocity estimation
- geomap_*rlks.trans = ROI_PAC file, with inverse mapping transformation from radar to geo coordinates, check more [ROI_PAC File Descriptions](#), copied in load_data step
 - radar_*rlks.hgt = ROI_PAC DEM file in radar coordinate, check more [ROI_PAC File Descriptions](#), copied in load_data step
- geo_* = transformed from radar coord to geo coord using [geocode.py](#)
 - Modified_* = network modification using [modify_network.py](#)
 - subset_* = subset/crop in space using [subset.py](#)
 - Seeded_* = referencing/seeding in space using [seed_data.py](#)
- *_demErr = DEM error correction in time series domain
 - *Ex = processed with some date(s) dropped
 - *_ECMWF/MERRA/NARR = tropospheric correction using PyAPS, name is the weather re-analysis data used to estimate the tropospheric phase delay
 - *_plane/quadratic/... = phase ramp removal
 - *_refDate = referencing in time

10 Gamma-File-Decription

Basically, in GAMMA, we can name the file in any "nickname" if we want. But, there are also some common habits to name different type of files to make non-GAMMA guys readable, which is very similar like other softwares but not absolutely same. Here will introduce some common names of GAMMA-based files from SLC step to Unwrapping step.

ps: GAMMA software has several modules: MSP, ISP, DIFF&GEO, IPTA. MSP for focusing, ISP for interferometry, DIFF&GEO for DInSAR and gecoding, IPTA mainly for TS-InSAR (conventional PS and SBAS).

*****MSP*****
(skipped here)

*****ISP*****
*.slc (same thing as roi_pac)

*.slc.par (parameters' file about orbit, width, length, time, ... But parameters in *.par file is far less than *.rsc file)

*.mli (magnitude image of SLC after doing multilook)

*.mli.par (same thing like *.slc.par, but width and length are changed due to multi-looking)

*.rslc (co-registered SLC, for TS-InSAR, usually coregistrated to one master image)

*.rslc.par (parameter file of *.rslc, absolutely same as *.slc.par)

*.rmli (co-registered magnitude images of multi-looked SLC)
 *.rmli.par (parameter file of ...)
 *.off (offset file of co-registration, include fitted polynomial parameters, length, width, ...)
 *.offs (COMPLEX file, offset value in each chosen points, real and imaginary parts for Range and Azimuth offset)
 *.snr (std of co-registration in each point, which will be used to mask some points based on a threshold)
 *.offset (text file of *.offs)
 *.coffs (COMPLEX file, culled offset of *.offs)
 *.coffsets (text type of *.coffs)
 *.base (baseline file)
 *.base.perp (perpendicular baseline file)
 *.cc (coherence map)
 *.int (original interferometry file, include every signal, flatten phase, DEM, Def, APS,...)
 *.flt ("flatten" interferogram, after removing flatten signals from *.int)
 *.smcc (coherence map based on filtered interferogram)
 *.sm_flt (filtered *.flt interferogram)
 *****DIFF & GEOCODE*****
 *.diff (interferogram that has removed flatten signals and topography signals)
 *.flag (masked file based on coherence map, 0 and 1, only used for Branch-cut unwrapping)
 *.mask.ras (masked file for MCF unwrapping, also masked based on coherence)
 *.unw (unwrapped interferogram, usually unwrapped from *.diff, data type order is different from that of ROI_PAC's .unw file)
 The same thing as ISP, all files based on filtering will include "sm", e.g., *sm.diff*, **.sm.unw*, but the final part of suffix will not change.
 *.htg (digital elevation model in radar coordinates)
 *.dem (..... in UTM coordinates)
 *.dem.par (parameters of *.dem file, which is in UTM coordinates, same as *.dem.rsc in ROI_PAC)
 *.utm_to_rdc (lookup table: from utm to radar coordinates)

11 Google-Earth

PySAR use [pyKML](#) module to output files into [Google Earth](#) .kmz format using script [save_kml.py](#). Check its usage by typing "save_kml.py -h" in your terminal. Below is an screenshot of the velocity of [Kuju example](#) using the command:

```
save_kml.py geo_velocity_masked.h5 -c jet_r -u cm --ylim -2.5 0.5 --cbar-height 2000
```

- [Download KMZ file](#)

12 Home

Github Page: <https://yunjunz.github.io/PySAR/>
 Google Group: <https://groups.google.com/forum/#!forum/py-sar>
 Workshop 2017: [PDF](#)

Documentation: [PDF](#)

13 SAR-Sensor-Parameter

Here is summary of SAR sensor parameters commonly used in InSAR.

JERS-1 (Japan Earth Resources Satellite, nickname of Fuyo-1). Data available from 1992 - 1998.

Information from: [ESA EO portal](#)

Center frequency	1.275 GHz (L-band, 23.5 cm wavelength)
Bandwidth	15 MHz
Observation Mode	StripMap
Spatial resolution	18 m (range) x 18 m (azimuth, 3 looks)
Swath width	75 km
Pulse width	35 μ s
PRF	1505.8 - 1606.0 Hz
Antenna	Array of 1024 microstrip radiation elements
- Polarization	HH
- Look angle	35.21°
- Antenna gain	>33.5 dB
- Signal to ambiguity ratio	>14 dB

14 UNAVCO-InSAR-Archive

Use the following commands to convert PySAR product into [UNAVCO InSAR Archive](#) format. All files should be geocoded in the same coordinations and resolution.

```
add_attribute.py timeseries.h5 add_attribute.txt
save_unavco.py timeseries.h5 -i incidence_angle.h5 -d dem.h5 -c temporal_coherence.h5 -m mask.h5
```

Create an text file (i.e. *add_attribute.txt*) with the following attributes and manual modify them for your dataset.

```
##### UNAVCO Required Metadata
mission           = ALOS                      # ERS, ENV, S1, RS1, RS2, CSK, TSX, JERS, ALOS, ALOS2
beam_mode         = SM                      # S2, IW
beam_swath        = 7
relative_orbit     = 422
processing_software = ROI_PAC
processing_type    = LOS_TIMESERIES
#first_date       =                        # grab by script
#last_date        =                        # grab by script
#scene_footprint  =                        # grab by script
#history          =                        # grab by script
frame             = 650                     # first frame number, need in file name

##### UNAVCO Recommended Metadata
atmos_correct_method = ERA-Interim
post_processing_method = PySAR
processing_dem       = GSI_DEHM_10m
unwrap_method       = SNAPHU
#flight_direction    =                        # grab by script
#look_direction      =                        # grab by script
#polarization        =
#prf                 =                        # grab by script
#wavelength          =                        # grab by script
#master_platform     =                        #For INTERFEROGRAM products
#master_absolute_orbit =                    #For INTERFEROGRAM products
#master_doppler       =                        #For INTERFEROGRAM products
#slave_platform      =                        #For INTERFEROGRAM products
#slave_absolute_orbit =                    #For INTERFEROGRAM products
#slave_doppler        =                        #For INTERFEROGRAM products
#percent_unwrapped    =
#average_coherence    =
#max_coherence        =
```

```
#percent_atmos_corrected =
#baseline_perp          =

##### INSARMAPS Metadata
reference      = 'Yunjun, Z., Amelung F., Aoki Y., (2016). Poster: A time series InSAR survey of volcanic deform
referencePdf   = 'https://yunjunzhang.files.wordpress.com/2015/01/yunjun_2016_agu.pdf'
unavcoUrl      = ''
```

Baker, S., (2015), Product Format Specification of UNAVCO InSAR Product Archive [DOC](#)

15 Web-Viewer

You could check the InSAR time-series products processed by University of Miami Geodesy Lab through its web viewer below:

Time series displacement of Kuju volcano from ALOS dataset (Track 422, Frame 650)

16 Namespace Index

16.1 Packages

Here are the packages with brief descriptions (if available):

animation	19
delayTimeseries	22
dloadUtil	22
get_modis_v3	23
plot_tropcor_phase_elevation	25
pysar	29
pysar._datetime	32
pysar._gmt	35
pysar._network	36
pysar._plot	47
pysar._pysar_utilities	48
pysar._readfile	62
pysar._remove_surface	69
pysar._sensor	70

pysar._variance	70
pysar._writefile	71
pysar.add	73
pysar.add_attribute	75
pysar.add_attribute_insarmaps	75
pysar.asc_desc	76
pysar.baseline_error	77
pysar.baseline_trop	78
pysar.coord_glob2radar	78
pysar.coord_radar2glob	79
pysar.correct_dem	79
pysar.correlation_with_dem	80
pysar.dem_error	80
pysar.diff	82
pysar.gamma_view	83
pysar.generate_mask	83
pysar.geocode	84
pysar.hdfeos5_2insarmaps	88
pysar.hdfeos5_2json_mbtiles	89
pysar.ifgram_closure	91
pysar.ifgram_inversion	91
pysar.ifgram_reconstruction	97
pysar.ifgram_simulation	97
pysar.image_math	98
pysar.incidence_angle	99
pysar.info	100
pysar.insar_vs_gps	101
pysar.insarmaps_query	102
pysar.json_mbtiles2insarmaps	103
pysar.l1	104
pysar.load_data	106
pysar.load_dem	110

pysar.lod	112
pysar.look_angle	113
pysar.los2enu	113
pysar.mask	114
pysar.match	115
pysar.modify_network	117
pysar.multi_transect	119
pysar.multilook	140
pysar.perp_baseline	142
pysar.plot_network	142
pysar.prep4timeseries	144
pysar.prep_gamma	147
pysar.prep_giant_ifg_list	149
pysar.prep_isce	150
pysar.prep_roipac	152
pysar.pysarApp	153
pysar.quality_map	155
pysar.range_distance	155
pysar.reference_epoch	156
pysar.remove_plane	157
pysar.rewrap	158
pysar.save_gmt	159
pysar.save_hdfeos5	160
pysar.save_kml	162
pysar.save_mat	163
pysar.save_roipac	164
pysar.seed_data	165
pysar.select_network	168
pysar.spatial_average	170
pysar.spatial_filter	171
pysar.stacking	172
pysar.subset	173

pysar.sum_epochs	178
pysar.temporal_average	179
pysar.temporal_coherence	179
pysar.temporal_derivative	181
pysar.temporal_filter	181
pysar.timeseries2velocity	182
pysar.timeseries_rms	184
pysar.transect	185
pysar.transect_legacy	187
pysar.tropcor_phase_elevation	207
pysar.tropcor_pyaps	207
pysar.tsviewer	210
pysar.unwrap_error	223
pysar.view	226
troposphere_uncertainty	233

17 Hierarchical Index

17.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

BasicHTTP	237
object	
timeseries	254
JERS	248
InsarDatabaseController	238
InsarDatasetController	244
progress_bar	250
Basemap	
Basemap2	234

18 Class Index

18.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Basemap2	
Class #####	234
BasicHTTP	237
InsarDatabaseController	238
InsarDatasetController	244
JERS	
Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang #	248
progress_bar	
Simple progress bar#####	250
timeseries	254

19 File Index

19.1 File List

Here is a list of all files with brief descriptions:

__init__.py	260
_datetime.py	260
_gmt.py	261
_network.py	261
_plot.py	262
_pysar_utilities.py	262
_readfile.py	263
_remove_surface.py	264
_sensor.py	264
_variance.py	264
_writefile.py	265
add.py	265
add_attribute.py	265
add_attribute_insarmaps.py	266
animation.py	266
asc_desc.py	266
baseline_error.py	267
baseline_trop.py	267

coord_glob2radar.py	267
coord_radar2glob.py	268
correct_dem.py	268
correlation_with_dem.py	268
delayTimeseries.py	268
dem_error.py	269
diff.py	269
dloadUtil.py	270
gamma_view.py	270
generate_mask.py	270
geocode.py	271
get_modis_v3.py	271
hdfeos5_2insarmaps.py	272
hdfeos5_2json_mbtiles.py	272
ifgram_closure.py	272
ifgram_inversion.py	273
ifgram_reconstruction.py	273
ifgram_simulation.py	273
image_math.py	274
incidence_angle.py	274
info.py	274
insar_vs_gps.py	275
insarmaps_query.py	275
json_mbtiles2insarmaps.py	275
l1.py	276
load_data.py	276
load_dem.py	277
lod.py	277
look_angle.py	277
los2enu.py	278
mask.py	278
match.py	278

modify_network.py	279
multi_transect.py	279
multilook.py	282
perp_baseline.py	282
plot_network.py	283
plot_tropcor_phase_elevation.py	283
prep4timeseries.py	284
prep_gamma.py	285
prep_giant_ifg_list.py	285
prep_isce.py	285
prep_roipac.py	286
pysarApp.py	286
quality_map.py	287
range_distance.py	287
reference_epoch.py	287
remove_plane.py	288
rewrap.py	288
save_gmt.py	288
save_hdfeos5.py	289
save_kml.py	289
save_mat.py	289
save_roipac.py	290
seed_data.py	290
select_network.py	291
spatial_average.py	291
spatial_filter.py	292
stacking.py	292
subset.py	292
sum_epochs.py	293
temporal_average.py	293
temporal_coherence.py	294
temporal_derivative.py	294

temporal_filter.py	294
timeseries2velocity.py	295
timeseries_rms.py	295
transect.py	295
transect_legacy.py	296
tropcor_phase_elevation.py	299
tropcor_pyaps.py	299
troposphere_uncertainty.py	300
tsviewer.py	300
unwrap_error.py	302
view.py	303

20 Namespace Documentation

20.1 animation Namespace Reference

Functions

- def [updatefig](#) (args)

Variables

- string [work_dir](#) = '/Users/yunjunz/insarlab/Galapagos/AlcedoEnvA2T061/PIC'
- list [fileList](#) = []
- list [titleList](#) = []
- list [imgs](#) = []
- [img](#) = mpimg.imread(fname)
- [fig](#) = plt.figure(figsize=[10, 5.4])
- [ax](#) = fig.add_axes([0.05, 0.05, 0.9, 0.8])
- int [i](#) = -2
- [im](#) = ax.imshow([imgs](#)[i], animated=True)
- [ttl](#) = ax.text(200, -150, [titleList](#)[i], ha='left', fontsize=32)
- [ani](#) = animation.FuncAnimation([fig](#), [updatefig](#), interval=1000, blit=True)
- [savefigDict](#) = dict()
- string [outName](#) = 'timeseries_animation.gif'
- [writer](#)
- [dpi](#)
- [savefig_kwargs](#)

20.1.1 Function Documentation

20.1.1.1 updatefig()

```
def animation.updatefig (  
    args )
```

20.1.2 Variable Documentation

20.1.2.1 ani

```
ani = animation.FuncAnimation(fig, updatefig, interval=1000, blit=True)
```

20.1.2.2 ax

```
ax = fig.add_axes([0.05, 0.05, 0.9, 0.8])
```

20.1.2.3 dpi

```
dpi
```

20.1.2.4 fig

```
fig = plt.figure(figsize=[10, 5.4])
```

20.1.2.5 fileList

```
list fileList = []
```

20.1.2.6 i

```
int i = -2
```

20.1.2.7 im

```
im = ax.imshow(imgs[i], animated=True)
```

20.1.2.8 img

```
img = mpimg.imread(fname)
```

20.1.2.9 imgs

```
list imgs = []
```

20.1.2.10 outName

```
outName = 'timeseries_animation.gif'
```

20.1.2.11 savefig_kwargs

```
savefig_kwargs
```

20.1.2.12 savefigDict

```
savefigDict = dict()
```

20.1.2.13 titleList

```
list titleList = []
```

20.1.2.14 ttl

```
ttl = ax.text(200, -150, titleList[i], ha='left', fontsize=32)
```

20.1.2.15 work_dir

```
string work_dir = '/Users/yunjunz/insarlab/Galapagos/AlcedoEnvA2T061/PIC'
```

20.1.2.16 writer

```
writer
```

20.2 delayTimeseries Namespace Reference

Classes

- class [timeseries](#)

Functions

- def [write_to_h5](#) (dataset, outName, groupName, h5withAttributes)
- def [nearest_valid](#) (xr, yr, data_flat, rows, cols)

20.2.1 Function Documentation

20.2.1.1 nearest_valid()

```
def delayTimeseries.nearest_valid (
    xr,
    yr,
    data_flat,
    rows,
    cols )
```

20.2.1.2 write_to_h5()

```
def delayTimeseries.write_to_h5 (
    dataset,
    outName,
    groupName,
    h5withAttributes )
```

20.3 dloadUtil Namespace Reference

Functions

- def [download_modis](#) (inps)
- def [download_atmosphereModel](#) (inps)
- def [daterange](#) (start_date, end_date)
- def [get_date](#) (f)
- def [pwv2zwd](#) (pwv)
- def [zwd2swd](#) (zwd, theta)
- def [read_modis](#) (file)

20.3.1 Function Documentation

20.3.1.1 daterange()

```
def dloadUtil.daterange (
    start_date,
    end_date )
```

20.3.1.2 download_atmosphereModel()

```
def dloadUtil.download_atmosphereModel (
    inps )
```

20.3.1.3 download_modis()

```
def dloadUtil.download_modis (
    inps )
```

20.3.1.4 get_date()

```
def dloadUtil.get_date (
    f )
```

20.3.1.5 pwv2zwd()

```
def dloadUtil.pwv2zwd (
    pwv )
```

20.3.1.6 read_modis()

```
def dloadUtil.read_modis (
    file )
```

20.3.1.7 zwd2swd()

```
def dloadUtil.zwd2swd (
    zwd,
    theta )
```

20.4 get_modis_v3 Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) ()

Variables

- [out](#)
- [start_time_main](#)
- [time_elapsed](#)

20.4.1 Function Documentation

20.4.1.1 `main()`

```
def get_modis_v3.main ( )
```

20.4.1.2 `usage()`

```
def get_modis_v3.usage ( )
```

20.4.2 Variable Documentation

20.4.2.1 `out`

`out`

20.4.2.2 `start_time_main`

`start_time_main`

20.4.2.3 `time_elapsed`

`time_elapsed`

20.5 plot_tropcor_phase_elevation Namespace Reference

Variables

- [workDir](#)
- [demFile](#)
- [timeseriesFile](#)
- [timeseriesFile2](#)
- [maskFile](#)
- [tropHgtFile](#)
- [ecmwfFile](#)
- [epoch](#)
- [dem](#)
- [dem_atr](#)
- [data](#)
- [atr](#)
- [data2](#)
- [atr2](#)
- [tropHgt](#)
- [atr3](#)
- [ecmwf](#)
- [atr4](#)
- [mask](#)
- [msk_atr](#)
- [ndx](#)
- [dataList](#)
- [fig](#)
- [axes](#)
- [nrows](#)
- [ncols](#)
- [sharex](#)
- [True](#)
- [sharey](#)
- [figsize](#)
- [i](#)
- [ms](#)
- [bbox_inches](#)
- [dpi](#)

20.5.1 Variable Documentation

20.5.1.1 atr

`atr`

20.5.1.2 atr2

`atr2`

20.5.1.3 atr3

atr3

20.5.1.4 atr4

atr4

20.5.1.5 axes

axes

20.5.1.6 bbox_inches

bbox_inches

20.5.1.7 data

data

20.5.1.8 data2

data2

20.5.1.9 dataList

dataList

20.5.1.10 dem

dem

20.5.1.11 dem_atr

dem_atr

20.5.1.12 demFile

demFile

20.5.1.13 dpi

dpi

20.5.1.14 ecmwf

ecmwf

20.5.1.15 ecmwfFile

ecmwfFile

20.5.1.16 epoch

epoch

20.5.1.17 fig

fig

20.5.1.18 figsize

figsize

20.5.1.19 i

i

20.5.1.20 mask

mask

20.5.1.21 maskFile

maskFile

20.5.1.22 ms

ms

20.5.1.23 msk_atr

msk_atr

20.5.1.24 ncols

ncols

20.5.1.25 ndx

ndx

20.5.1.26 nrows

nrows

20.5.1.27 sharex

sharex

20.5.1.28 sharey

sharey

20.5.1.29 timeseriesFile

timeseriesFile

20.5.1.30 timeseriesFile2

timeseriesFile2

20.5.1.31 tropHgt

tropHgt

20.5.1.32 tropHgtFile

tropHgtFile

20.5.1.33 True

True

20.5.1.34 workDir

workDir

20.6 pysar Namespace Reference

Namespaces

- [_datetime](#)
- [_gmt](#)
- [_network](#)
- [_plot](#)
- [_pysar_utilities](#)
- [_readfile](#)
- [_remove_surface](#)
- [_sensor](#)
- [_variance](#)
- [_writefile](#)
- [add](#)
- [add_attribute](#)
- [add_attribute_insarmaps](#)
- [asc_desc](#)
- [baseline_error](#)
- [baseline_trop](#)
- [coord_glob2radar](#)
- [coord_radar2glob](#)
- [correct_dem](#)
- [correlation_with_dem](#)

- [dem_error](#)
- [diff](#)
- [gamma_view](#)
- [generate_mask](#)
- [geocode](#)
- [hdfeos5_2insarmaps](#)
- [hdfeos5_2json_mbtiles](#)
- [ifgram_closure](#)
- [ifgram_inversion](#)
- [ifgram_reconstruction](#)
- [ifgram_simulation](#)
- [image_math](#)
- [incidence_angle](#)
- [info](#)
- [insar_vs_gps](#)
- [insarmaps_query](#)
- [json_mbtiles2insarmaps](#)
- [l1](#)
- [load_data](#)
- [load_dem](#)
- [lod](#)
- [look_angle](#)
- [los2enu](#)
- [mask](#)
- [match](#)
- [modify_network](#)
- [multi_transect](#)
- [multilook](#)
- [perp_baseline](#)
- [plot_network](#)
- [prep4timeseries](#)
- [prep_gamma](#)
- [prep_giant_ifg_list](#)
- [prep_isce](#)
- [prep_roipac](#)
- [pysarApp](#)
- [quality_map](#)
- [range_distance](#)
- [reference_epoch](#)
- [remove_plane](#)
- [rewrap](#)
- [save_gmt](#)
- [save_hdfeos5](#)
- [save_kml](#)
- [save_mat](#)
- [save_roipac](#)
- [seed_data](#)
- [select_network](#)
- [spatial_average](#)
- [spatial_filter](#)
- [stacking](#)
- [subset](#)
- [sum_epochs](#)
- [temporal_average](#)
- [temporal_coherence](#)

- [temporal_derivative](#)
- [temporal_filter](#)
- [timeseries2velocity](#)
- [timeseries_rms](#)
- [transect](#)
- [transect_legacy](#)
- [tropcor_phase_elevation](#)
- [tropcor_pyaps](#)
- [tsviewer](#)
- [unwrap_error](#)
- [view](#)

Variables

- bool [miami_path](#) = True
- int [parallel_num](#) = 8
- float [figsize_single_min](#) = 6.0
- float [figsize_single_max](#) = 12.0
- list [figsize_multi](#) = [15.0, 8.0]

20.6.1 Variable Documentation

20.6.1.1 [figsize_multi](#)

```
list figsize_multi = [15.0, 8.0]
```

20.6.1.2 [figsize_single_max](#)

```
float figsize_single_max = 12.0
```

20.6.1.3 [figsize_single_min](#)

```
float figsize_single_min = 6.0
```

20.6.1.4 [miami_path](#)

```
bool miami_path = True
```

20.6.1.5 [parallel_num](#)

```
int parallel_num = 8
```

20.7 pysar._datetime Namespace Reference

Classes

- class [progress_bar](#)
Simple progress bar#####.

Functions

- def [yyyymmdd2years](#) (dates)
- def [yymmdd2yyyymmdd](#) (date)
- def [yyyymmdd](#) (dates)
- def [yymmdd](#) (dates)
- def [ifgram_date_list](#) (ifgramFile, fmt='YYYYMMDD')
- def [read_date_list](#) (date_list_file)
- def [date_index](#) (dateList)
- def [date_list2tbase](#) (dateList)
- def [date_list2vector](#) (dateList)
- def [auto_adjust_xaxis_date](#) (ax, datevector, fontSize=12, every_year=1)
- def [list_ifgram2date12](#) (ifgram_list)
- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')

20.7.1 Function Documentation

20.7.1.1 auto_adjust_xaxis_date()

```
def pysar._datetime.auto_adjust_xaxis_date (
    ax,
    datevector,
    fontSize = 12,
    every_year = 1 )
```

Adjust X axis

Input:

```
ax : matplotlib figure axes object
datevector : list of float, date in years
             i.e. [2007.013698630137, 2007.521917808219, 2007.6463470319634]
```

Output:

```
ax - matplotlib figure axes object
dss - datetime.date object, xmin
dee - datetime.date object, xmax
```

20.7.1.2 closest_weather_product_time()

```
def pysar._datetime.closest_weather_product_time (
    sar_acquisition_time,
    grib_source = 'ECMWF' )
```

Find closest available time of weather product from SAR acquisition time

Inputs:

 sar_acquisition_time - string, SAR data acquisition time in seconds
 grib_source - string, Grib Source of weather reanalysis product

Output:

 grib_hr - string, time of closest available weather product

Example:

```
'06:00' = closest_weather_product_time(atr['CENTER_LINE_UTC'], 'ECMWF')
'12'    = closest_weather_product_time(atr['CENTER_LINE_UTC'], 'NARR')
```

20.7.1.3 date_index()

```
def pysar._datetime.date_index (
    dateList )
```

20.7.1.4 date_list2tbase()

```
def pysar._datetime.date_list2tbase (
    dateList )
```

Get temporal Baseline in days with respect to the 1st date

Input: dateList - list of string, date in YYYYMMDD or YYMMDD format

Output:

 tbase - list of int, temporal baseline in days
 dateDict - dict with key - string, date in YYYYMMDD format
 value - int, temporal baseline in days

20.7.1.5 date_list2vector()

```
def pysar._datetime.date_list2vector (
    dateList )
```

Get time in datetime format: datetime.datetime(2006, 5, 26, 0, 0)

Input: dateList - list of string, date in YYYYMMDD or YYMMDD format

Outputs:

 dates - list of datetime.datetime objects, i.e. datetime.datetime(2010, 10, 20, 0, 0)
 datevector - list of float, years, i.e. 2010.8020547945205

20.7.1.6 ifgram_date_list()

```
def pysar._datetime.ifgram_date_list (
    ifgramFile,
    fmt = 'YYYYMMDD' )
```

Read Date List from Interferogram file

for timeseries file, use h5file['timeseries'].keys() directly

Inputs:

ifgramFile - string, name/path of interferograms file

fmt - string, output date format, choices=['YYYYMMDD', 'YYMMDD']

Output:

date_list - list of string, date included in ifgramFile in YYYYMMDD or YYMMDD format

20.7.1.7 list_ifgram2date12()

```
def pysar._datetime.list_ifgram2date12 (
    ifgram_list )
```

Convert ifgram list into date12 list

Input:

ifgram_list - list of string in *YYMMDD-YYMMDD* or *YYMMDD_YYMMDD* format

Output:

date12_list - list of string in YYMMDD-YYMMDD format

Example:

```
h5 = h5py.File('unwrapIfgram.h5', 'r')
```

```
ifgram_list = sorted(h5['interferograms'].keys())
```

```
date12_list = ptime.list_ifgram2date12(ifgram_list)
```

20.7.1.8 read_date_list()

```
def pysar._datetime.read_date_list (
    date_list_file )
```

Read Date List from txt file

20.7.1.9 yymmdd()

```
def pysar._datetime.yymmdd (
    dates )
```

20.7.1.10 yymmdd2yyyymmdd()

```
def pysar._datetime.yymmdd2yyyymmdd (
    date )
```

20.7.1.11 `yyyymmdd()`

```
def pysar._datetime.yyyymmdd (
    dates )
```

20.7.1.12 `yyyymmdd2years()`

```
def pysar._datetime.yyyymmdd2years (
    dates )
```

20.8 `pysar._gmt` Namespace Reference

Functions

- def [write_gmt_simple](#) (lons, lats, z, fname, title='default', name='z', scale=1.0, offset=0, units='meters')

20.8.1 Function Documentation

20.8.1.1 `write_gmt_simple()`

```
def pysar._gmt.write_gmt_simple (
    lons,
    lats,
    z,
    fname,
    title = 'default',
    name = 'z',
    scale = 1.0,
    offset = 0,
    units = 'meters' )
```

Writes a simple GMT grd file with one array.

.. Args:

```
* lons    -> 1D Array of lon values
* lats    -> 1D Array of lat values
* z       -> 2D slice to be saved
* fname   -> Output file name
```

.. Kwargs:

```
* title   -> Title for the grd file
* name    -> Name of the field in the grd file
* scale   -> Scale value in the grd file
* offset  -> Offset value in the grd file
```

.. Returns:

```
* None
```

20.9 pysar._network Namespace Reference

Functions

- def [read_pairs_list](#) (date12ListFile, dateList=[])
- def [write_pairs_list](#) (pairs, dateList, outName)
- def [read_igram_pairs](#) (igramFile)
- def [read_baseline_file](#) (baselineFile, exDateList=[])
- def [date12_list2index](#) (date12_list, date_list=[])
- def [get_date12_list](#) (File, check_drop_ifgram=False)
- def [igram_perp_baseline_list](#) (File)
- def [azimuth_bandwidth](#) (sensor)
- def [range_bandwidth](#) (sensor)
- def [wavelength](#) (sensor)
- def [incidence_angle](#) (sensor, inc_angle=None)
- def [signal2noise_ratio](#) (sensor)
- def [critical_perp_baseline](#) (sensor, inc_angle=None, print_msg=False)
- def [calculate_doppler_overlap](#) (dop_a, dop_b, bandwidth_az)
- def [simulate_coherence](#) (date12_list, baselineFile='bl_list.txt', sensor='Env', inc_angle=22.8, decor_↵
time=200.0, coh_resid=0.2, display=False)
- def [threshold_doppler_overlap](#) (date12_list, date_list, dop_list, bandwidth_az, dop_overlap_min=0.15)
- def [threshold_perp_baseline](#) (date12_list, date_list, pbase_list, pbase_max, pbase_min=0.0)
- def [threshold_temporal_baseline](#) (date12_list, btemp_max, keep_seasonal=True, btemp_min=0.0)
- def [coherence_matrix](#) (date12_list, coh_list, diagValue=np.nan)
- def [threshold_coherence_based_mst](#) (date12_list, coh_list)
- def [pair_sort](#) (pairs)
- def [pair_merge](#) (pairs1, pairs2)
- def [select_pairs_all](#) (date_list)
- def [select_pairs_sequential](#) (date_list, increment_num=2)
- def [select_pairs_hierarchical](#) (date_list, pbase_list, temp_perp_list)
- def [select_pairs_delaunay](#) (date_list, pbase_list, norm=True)
- def [select_pairs_mst](#) (date_list, pbase_list)
- def [select_pairs_star](#) (date_list, m_date=None, pbase_list=[])
- def [select_master_date](#) (date_list, pbase_list=[])
- def [select_master_interferogram](#) (date12_list, date_list, pbase_list, m_date=None)
- def [plot_network](#) (ax, date12_list, date_list, pbase_list, plot_dict={}, date12_list_drop=[], print_msg=True)
- def [plot_perp_baseline_hist](#) (ax, date8_list, pbase_list, plot_dict={}, date8_list_drop=[])
- def [plot_coherence_matrix](#) (ax, date12_list, coherence_list, date12_list_drop=[], plot_dict={})
- def [mode](#) (thelist)
- def [plot_coherence_history](#) (ax, date12_list, coherence_list, plot_dict={})
- def [auto_adjust_yaxis](#) (ax, dataList, fontSize=12, ymin=None, ymax=None)

Variables

- string [BASELINE_LIST_FILE](#)
- string [IFGRAM_LIST_FILE](#)

20.9.1 Function Documentation

20.9.1.1 auto_adjust_yaxis()

```
def pysar._network.auto_adjust_yaxis (
    ax,
    dataList,
    fontSize = 12,
    ymin = None,
    ymax = None )
```

Adjust Y axis

Input:

ax : matplotlib figure axes object
 dataList : list of float, value in y axis
 fontSize : float, font size
 ymin : float, lower y axis limit
 ymax : float, upper y axis limit

Output:

ax

20.9.1.2 azimuth_bandwidth()

```
def pysar._network.azimuth_bandwidth (
    sensor )
```

Find the hardwired azimuth bandwidth in hertz for the given satellite

20.9.1.3 calculate_doppler_overlap()

```
def pysar._network.calculate_doppler_overlap (
    dop_a,
    dop_b,
    bandwidth_az )
```

Calculate Overlap Percentage of Doppler frequency in azimuth direction

Inputs:

dop_a/b : np.array of 3 floats, doppler frequency
 bandwidth_az : float, azimuth bandwidth

Output:

dop_overlap : float, doppler frequency overlap between a & b.

20.9.1.4 coherence_matrix()

```
def pysar._network.coherence_matrix (
    date12_list,
    coh_list,
    diagValue = np.nan )
```

Return coherence matrix based on input date12 list and its coherence

Inputs:

date12_list - list of string in YYMMDD-YYMMDD format
 coh_list - list of float, average coherence for each interferograms

Output:

coh_matrix - 2D np.array with dimension length = date num
 np.nan value for interferograms non-existed.
 1.0 for diagonal elements

20.9.1.5 critical_perp_baseline()

```
def pysar._network.critical_perp_baseline (
    sensor,
    inc_angle = None,
    print_msg = False )
```

Critical Perpendicular Baseline for each satellite

20.9.1.6 date12_list2index()

```
def pysar._network.date12_list2index (
    date12_list,
    date_list = [] )
```

Convert list of date12 string into list of index

20.9.1.7 get_date12_list()

```
def pysar._network.get_date12_list (
    File,
    check_drop_ifgram = False )
```

Read Date12 info from input file: Pairs.list or multi-group hdf5 file

Inputs:

File - string, path/name of input multi-group hdf5 file or text file

check_drop_ifgram - bool, check the "drop_ifgram" attribute or not for multi-group hdf5 file

Output:

date12_list - list of string in YYMMDD-YYMMDD format

Example:

```
date12List = get_date12_list('unwrapIfgram.h5')
date12List = get_date12_list('unwrapIfgram.h5', check_drop_ifgram=True)
date12List = get_date12_list('Pairs.list')
```

20.9.1.8 igram_perp_baseline_list()

```
def pysar._network.igram_perp_baseline_list (
    File )
```

Get perpendicular baseline list from input multi_group hdf5 file

20.9.1.9 incidence_angle()

```
def pysar._network.incidence_angle (
    sensor,
    inc_angle = None )
```

20.9.1.10 mode()

```
def pysar._network.mode (
    thelist )
```

Find Mode (most common) item in the list
Borrowed from pysar._pysar_utilities

20.9.1.11 pair_merge()

```
def pysar._network.pair_merge (
    pairs1,
    pairs2 )
```

20.9.1.12 pair_sort()

```
def pysar._network.pair_sort (
    pairs )
```

20.9.1.13 plot_coherence_history()

```
def pysar._network.plot_coherence_history (
    ax,
    date12_list,
    coherence_list,
    plot_dict = {} )
```

Plot min/max Coherence of all interferograms for each date

20.9.1.14 plot_coherence_matrix()

```
def pysar._network.plot_coherence_matrix (
    ax,
    date12_list,
    coherence_list,
    date12_list_drop = [],
    plot_dict = {} )
```

Plot Coherence Matrix of input network

if date12_list_drop is not empty, plot KEPT pairs in the upper triangle and
ALL pairs in the lower triangle.

20.9.1.15 plot_network()

```
def pysar._network.plot_network (
    ax,
    date12_list,
    date_list,
    pbase_list,
    plot_dict = {},
    date12_list_drop = [],
    print_msg = True )
```

Plot Temporal-Perp baseline Network

Inputs

```
ax : matplotlib axes object
date12_list : list of string for date12 in YYMMDD-YYMMDD format
date_list : list of string, for date in YYYYMMDD/YYMMDD format
pbase_list : list of float, perp baseline, len=number of acquisition
plot_dict : dictionary with the following items:
    fontsize
    linewidth
    markercolor
    markersize
```

```
coherence_list : list of float, coherence value of each interferogram, len = number of ifgra
disp_min/max : float, min/max range of the color display based on coherence_list
colormap : string, colormap name
coh_thres : float, coherence of where to cut the colormap for display
disp_title : bool, show figure title or not, default: True
disp_drop: bool, show dropped interferograms or not, default: True
```

Output

```
ax : matplotlib axes object
```

20.9.1.16 plot_perp_baseline_hist()

```
def pysar._network.plot_perp_baseline_hist (
    ax,
    date8_list,
    pbase_list,
    plot_dict = {},
    date8_list_drop = [] )
```

Plot Perpendicular Spatial Baseline History

Inputs

```

ax : matplotlib axes object
date8_list : list of string, date in YYYYMMDD format
pbase_list : list of float, perp baseline
plot_dict : dictionary with the following items:
    fontsize
    linewidth
    markercolor
    markersize
    disp_title : bool, show figure title or not, default: True
    every_year : int, number of years for the major tick on xaxis
date8_list_drop : list of string, date dropped in YYYYMMDD format
    e.g. ['20080711', '20081011']

```

Output:

```

ax : matplotlib axes object

```

20.9.1.17 range_bandwidth()

```

def pysar._network.range_bandwidth (
    sensor )

```

20.9.1.18 read_baseline_file()

```

def pysar._network.read_baseline_file (
    baselineFile,
    exDateList = [] )

```

Read bl_list.txt without dates listed in exDateList

#	Date	Bperp	dop0/PRF	dop1/PRF	dop2/PRF	PRF	slcDir
070106		0.0	0.03	0.0000000	0.00000000000	2155.2	/scratch/KyushuT422F650AlosA/SLC/070106/
070709	2631.9	0.07	0.0000000	0.00000000000	2155.2	/scratch/KyushuT422F650AlosA/SLC/070709/	
070824	2787.3	0.07	0.0000000	0.00000000000	2155.2	/scratch/KyushuT422F650AlosA/SLC/070824/	
...							

Examples:

```

date8List, perpBaseList, dopList, prfList, slcDirList = read_baseline_file(baselineFile)
date8List, perpBaseList, dopList, prfList, slcDirList = read_baseline_file(baselineFile, ['080520', '100726'])
date8List, perpBaseList = read_baseline_file(baselineFile)[0:2]

```

20.9.1.19 read_igram_pairs()

```

def pysar._network.read_igram_pairs (
    igramFile )

```

Read pairs index from hdf5 file

20.9.1.20 read_pairs_list()

```
def pysar._network.read_pairs_list (
    date12ListFile,
    dateList = [] )
```

Read Pairs List file like below:
 070311-070426
 070311-070611
 ...

20.9.1.21 select_master_date()

```
def pysar._network.select_master_date (
    date_list,
    pbase_list = [] )
```

Select super master date based on input temporal and/or perpendicular baseline info.
 Return master date in YYYYMMDD format.

20.9.1.22 select_master_interferogram()

```
def pysar._network.select_master_interferogram (
    date12_list,
    date_list,
    pbase_list,
    m_date = None )
```

Select reference interferogram based on input temp/perp baseline info
 If master_date is specified, select its closest slave_date, which is newer than master_date;
 otherwise, choose the closest pair among all pairs as master interferogram.

Example:

```
master_date12 = pnet.select_master_ifgram(date12_list, date_list, pbase_list)
'080211-080326' = pnet.select_master_ifgram(date12_list, date_list, pbase_list, m_date='080211')
```

20.9.1.23 select_pairs_all()

```
def pysar._network.select_pairs_all (
    date_list )
```

Select All Possible Pairs/Interferograms

Input : date_list - list of date in YYYYMMDD/YYYYMMDD format

Output: date12_list - list date12 in YYYYMMDD-YYYYMMDD format

Reference:

Berardino, P., G. Fornaro, R. Lanari, and E. Sansosti (2002), A new algorithm for surface deformation monitoring based on small baseline differential SAR interferograms, IEEE TGRS, 40(11), 2375-2383.

20.9.1.24 select_pairs_delaunay()

```
def pysar._network.select_pairs_delaunay (
    date_list,
    pbase_list,
    norm = True )
```

Select Pairs using Delaunay Triangulation based on temporal/perpendicular baselines

Inputs:

```
date_list : list of date in YYMMDD/YYYYMMDD format
pbase_list : list of float, perpendicular spatial baseline
norm      : normalize temporal baseline to perpendicular baseline
```

Key points

1. Define a ratio between perpendicular and temporal baseline axis units (Pepe and Lanari, 2006, TGRS).
2. Pairs with too large perpendicular / temporal baseline or Doppler centroid difference should be removed after this, using a threshold, to avoid strong decorrelations (Zebker and Villasenor, 1992, TGRS).

Reference:

Pepe, A., and R. Lanari (2006), On the extension of the minimum cost flow algorithm for phase unwrapping of multitemporal differential SAR interferograms, IEEE TGRS, 44(9), 2374-2383.
 Zebker, H. A., and J. Villasenor (1992), Decorrelation in interferometric radar echoes, IEEE TGRS, 30(5),

20.9.1.25 select_pairs_hierarchical()

```
def pysar._network.select_pairs_hierarchical (
    date_list,
    pbase_list,
    temp_perp_list )
```

Select Pairs in a hierarchical way using list of temporal and perpendicular baseline thresholds

For each temporal/perpendicular combination, select all possible pairs; and then merge all combination results together for the final output (Zhao, 2015).

Inputs:

```
date_list : list of date in YYMMDD/YYYYMMDD format
pbase_list : list of float, perpendicular spatial baseline
temp_perp_list : list of list of 2 floats, for list of temporal/perp baseline, e.g.
                [[32.0, 800.0], [48.0, 600.0], [64.0, 200.0]]
```

Examples:

```
pairs = select_pairs_hierarchical(date_list, pbase_list, [[32.0, 800.0], [48.0, 600.0], [64.0, 200.0]])
```

Reference:

Zhao, W., (2015), Small deformation detected from InSAR time-series and their applications in geophysics, dissertation, Univ. of Miami, Section 6.3.

20.9.1.26 select_pairs_mst()

```
def pysar._network.select_pairs_mst (
    date_list,
    pbase_list )
```

Select Pairs using Minimum Spanning Tree technique

Connection Cost is calculated using the baseline distance in perp and scaled temporal baseline (Pepe and Lanari, 2006, TGRS) plane.

Inputs:

```
date_list : list of date in YYMMDD/YYYYMMDD format
pbase_list : list of float, perpendicular spatial baseline
```

References:

Pepe, A., and R. Lanari (2006), On the extension of the minimum cost flow algorithm for phase unwrapping of multitemporal differential SAR interferograms, IEEE TGRS, 44(9), 2374-2383.
 Perissin D., Wang T. (2012), Repeat-pass SAR interferometry with partially coherent targets. IEEE TGRS. 27

20.9.1.27 select_pairs_sequential()

```
def pysar._network.select_pairs_sequential (
    date_list,
    increment_num = 2 )
```

Select Pairs in a Sequential way:

For each acquisition, find its increment_num nearest acquisitions in the past time.

Inputs:

date_list : list of date in YYMMDD/YYYYMMDD format

Reference:

Fattahi, H., and F. Amelung (2013), DEM Error Correction in InSAR Time Series, IEEE TGRS, 51(7), 4249-4259

20.9.1.28 select_pairs_star()

```
def pysar._network.select_pairs_star (
    date_list,
    m_date = None,
    pbase_list = [] )
```

Select Star-like network/interferograms/pairs, it's a single master network, similar to PS approach.

Usage:

m_date : master date, choose it based on the following criteria:

- 1) near the center in temporal and spatial baseline
- 2) prefer winter season than summer season for less temporal decorrelation

Reference:

Ferretti, A., C. Prati, and F. Rocca (2001), Permanent scatterers in SAR interferometry, IEEE TGRS, 39(1),

20.9.1.29 signal2noise_ratio()

```
def pysar._network.signal2noise_ratio (
    sensor )
```

Fine the Signal to Noise Ratio in dB for the given satellite

Reference:

ERS - Zebker et al., 1994, TGRS

Envisat - Guarnieri, A.M., 2013. Introduction to RADAR. POLIMI DEI, Milano.

JERS - <https://directory.eoportal.org/web/eoportal/satellite-missions/j/jers-1>

Sentinel-1 - <https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/acquisition-modes/int>

20.9.1.30 simulate_coherence()

```
def pysar._network.simulate_coherence (
    date12_list,
    baselineFile = 'bl_list.txt',
    sensor = 'Env',
    inc_angle = 22.8,
    decor_time = 200.0,
    coh_resid = 0.2,
    display = False )
```

Simulate coherence for a given set of interferograms

Inputs:

date12_list - list of string in YYMMDD-YYMMDD format, indicating pairs configuration
 baselineFile - string, path of baseline list text file
 sensor - string, SAR sensor
 inc_angle - float, incidence angle
 decor_time - float / 2D np.array in size of (1, pixel_num)
 decorrelation rate in days, time for coherence to drop to 1/e of its initial value
 coh_resid - float / 2D np.array in size of (1, pixel_num)
 long-term coherence, minimum attainable coherence value
 display - bool, display result as matrix or not

Output:

cohs - 2D np.array in size of (ifgram_num, pixel_num)

Example:

```
date12_list = pnet.get_date12_list('ifgram_list.txt')
cohs = simulate_coherences(date12_list, 'bl_list.txt', sensor='Tsx')
```

References:

Zebker, H. A., & Villasenor, J. (1992). Decorrelation in interferometric radar echoes. IEEE-TGRS, 30(5), 950-959.
 Hanssen, R. F. (2001). Radar interferometry: data interpretation and error analysis (Vol. 2). Dordrecht, Netherlands: Kluwer Academic Pub.
 Morishita, Y., & Hanssen, R. F. (2015). Temporal decorrelation in L-, C-, and X-band satellite radar interferometry for pasture on drained peat soils. IEEE-TGRS, 53(2), 1096-1104.
 Parizzi, A., Cong, X., & Eineder, M. (2009). First Results from Multifrequency Interferometry. A comparison of different decorrelation time constants at L, C, and X Band. ESA Scientific Publications(SP-677), 1-5.

20.9.1.31 threshold_coherence_based_mst()

```
def pysar._network.threshold_coherence_based_mst (
    date12_list,
    coh_list )
```

Return a minimum spanning tree of network based on the coherence inverse.

Inputs:

date12_list - list of string in YYMMDD-YYMMDD format
 coh_list - list of float, average coherence for each interferogram

Output:

mst_date12_list - list of string in YYMMDD-YYMMDD format, for MST network of interferograms

20.9.1.32 threshold_doppler_overlap()

```
def pysar._network.threshold_doppler_overlap (
    date12_list,
    date_list,
    dop_list,
    bandwidth_az,
    dop_overlap_min = 0.15 )
```

Remove pairs/interferogram with doppler overlap larger than critical value

Inputs:

```
date12_list : list of string, for date12 in YYMMDD-YYMMDD format
date_list   : list of string, for date in YYMMDD/YYYYMMDD format, optional
dop_list    : list of list of 3 float, for centroid Doppler frequency
bandwidth_az : float, bandwidth in azimuth direction
dop_overlap_min : float, minimum overlap of azimuth Doppler frequency
```

Outputs:

```
date12_list : list of string, for date12 in YYMMDD-YYMMDD format
```

20.9.1.33 threshold_perp_baseline()

```
def pysar._network.threshold_perp_baseline (
    date12_list,
    date_list,
    pbase_list,
    pbase_max,
    pbase_min = 0.0 )
```

Remove pairs/interferogram out of [pbase_min, pbase_max]

Inputs:

```
date12_list : list of string for date12 in YYMMDD-YYMMDD format
date_list   : list of string for date in YYMMDD/YYYYMMDD format, optional
pbase_list  : list of float for perpendicular spatial baseline
pbase_max   : float, maximum perpendicular baseline
pbase_min   : float, minimum perpendicular baseline
```

Output:

```
date12_list_out : list of string for date12 in YYMMDD-YYMMDD format
```

Example:

```
date12_list = threshold_perp_baseline(date12_list, date_list, pbase_list, 500)
```

20.9.1.34 threshold_temporal_baseline()

```
def pysar._network.threshold_temporal_baseline (
    date12_list,
    btemp_max,
    keep_seasonal = True,
    btemp_min = 0.0 )
```

Remove pairs/interferograms out of min/max/seasonal temporal baseline limits

Inputs:

```
date12_list : list of string for date12 in YYMMDD-YYMMDD format
btemp_max   : float, maximum temporal baseline
btemp_min   : float, minimum temporal baseline
keep_seasonal : keep interferograms with seasonal temporal baseline
```

Output:

```
date12_list_out : list of string for date12 in YYMMDD-YYMMDD format
```

Example:

```
date12_list = threshold_temporal_baseline(date12_list, 200)
date12_list = threshold_temporal_baseline(date12_list, 200, False)
```


20.9.1.35 wavelength()

```
def pysar._network.wavelength (
    sensor )
```

20.9.1.36 write_pairs_list()

```
def pysar._network.write_pairs_list (
    pairs,
    dateList,
    outName )
```

Write pairs list file.

20.9.2 Variable Documentation

20.9.2.1 BASELINE_LIST_FILE

```
string BASELINE_LIST_FILE
```

Initial value:

```
1 = '''
2 # Date   Bperp   dop0/PRF   dop1/PRF   dop2/PRF   PRF       slcDir
3 070106    0.0     0.03      0.0000000 0.0000000 2155.2    /KyushuT422F650AlosA/SLC/070106/
4 070709   2631.9    0.07      0.0000000 0.0000000 2155.2    /KyushuT422F650AlosA/SLC/070709/
5 070824   2787.3    0.07      0.0000000 0.0000000 2155.2    /KyushuT422F650AlosA/SLC/070824/
6 ...
7 '''
```

20.9.2.2 IFGRAM_LIST_FILE

```
string IFGRAM_LIST_FILE
```

Initial value:

```
1 = '''
2 060713-070113
3 060828-070113
4 060828-070831
5 ...
6 '''
```

20.10 pysar._plot Namespace Reference

Functions

- def [plot_bar_std](#) (ax, date_list, std_list, fig_name=None, ref_date=None)

20.10.1 Function Documentation

20.10.1.1 `plot_bar_std()`

```
def pysar._plot.plot_bar_std (
    ax,
    date_list,
    std_list,
    fig_name = None,
    ref_date = None )
```

Plot Residual Standard Deviation into a Bar figure

Inputs

```
ax          - matplotlib axes object
date_list   - list of string, date in YYYYMMDD or YYMMDD format
std_list    - list of float, residual standard deviation
fig_name    - string, output figure name
ref_date    - string, reference date in YYYYMMDD or YYMMDD format
```

Output:

```
ax - matplotlib axes object
```

20.11 `pysar._pysar_utilities` Namespace Reference

Functions

- def `touch` (fname_list, times=None)
- def `get_lookup_file` (filePattern=None, abspath=False, print_msg=True)
- def `get_geometry_file` (dset, coordType=None, filePattern=None, abspath=False, print_msg=True)
- def `check_loaded_dataset` (work_dir='./', inps=None, print_msg=True)
- def `is_file_exist` (file_list, abspath=True)
- def `four_corners` (atr)
- def `circle_index` (atr, circle_par)
- def `update_template_file` (template_file, extra_dict)
- def `get_residual_std` (timeseries_resid_file, mask_file='maskTempCoh.h5', ramp_type='quadratic')
- def `timeseries_std` (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def `get_residual_rms` (timeseries_resid_file, mask_file='maskTempCoh.h5', ramp_type='quadratic')
- def `timeseries_rms` (inFile, maskFile='maskTempCoh.h5', outFile=None, dimension=2)
- def `timeseries_coherence` (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def `normalize_timeseries` (ts_mat, nanValue=0)
- def `normalize_timeseries_old` (ts_mat, nanValue=0)
- def `update_file` (outFile, inFile=None, overwrite=False, check_readable=True)
- def `update_attribute_or_not` (atr_new, atr_orig, update=False)
- def `add_attribute` (File, atr_new=dict())
- def `check_parallel` (file_num=1, print_msg=True)
- def `perp_baseline_timeseries` (atr, dimension=1)
- def `range_distance` (atr, dimension=2)
- def `incidence_angle` (atr, dimension=2, print_msg=True)
- def `which` (program)
- def `check_drop_ifgram` (h5, print_msg=True)
- def `nonzero_mask` (File, outFile='mask.h5')
- def `spatial_average` (File, maskFile=None, box=None, saveList=False, checkAoi=True)
- def `temporal_average` (File, outFile=None)

- def [get_file_list](#) (fileList, abspath=False, coord=None)
- def [check_file_size](#) (fname_list, mode_width=None, mode_length=None)
- def [mode](#) (thelist)
- def [range_ground_resolution](#) (atr, print_msg=False)
- def [azimuth_ground_resolution](#) (atr)
- def [get_lookup_row_col](#) (y, x, lut_y, lut_x, y_factor=10, x_factor=10, geoCoord=False)
 - Use geomap*.trans file for precious (pixel-level) coord conversion.*
- def [glob2radar](#) (lat, lon, lookupFile=None, atr_rdr=dict(), print_msg=True)
- def [radar2glob](#) (az, rg, lookupFile=None, atr_rdr=dict(), print_msg=True)
- def [check_variable_name](#) (path)
- def [hillshade](#) (data, scale)
- def [date_list](#) (h5file)
- def [design_matrix](#) (ifgramFile=None, date12_list=[], referenceDate=None, zero_first=True)
- def [timeseries_inversion_FGLS](#) (h5flat, h5timeseries)
- def [timeseries_inversion_L1](#) (h5flat, h5timeseries)
- def [perp_baseline_ifgram2timeseries](#) (ifgramFile, ifgram_list=[])
- def [dBh_dBv_timeseries](#) (ifgramFile)
- def [Bh_Bv_timeseries](#) (ifgramFile)
- def [get_file_stack](#) (File, maskFile=None)
- def [stacking](#) (File)
- def [yymmdd2YYYYMMDD](#) (date)
- def [yyyymmdd](#) (dates)
- def [yymmdd](#) (dates)
- def [make_triangle](#) (dates12, igram1, igram2, igram3)
- def [get_triangles](#) (h5file)
- def [generate_curls](#) (curlfile, h5file, Triangles, curls)

20.11.1 Function Documentation

20.11.1.1 [add_attribute\(\)](#)

```
def pysar._pysar_utilities.add_attribute (
    File,
    atr_new = dict() )
```

Add/update input attribute into File

Inputs:

File - string, path/name of file
 atr_new - dict, attributes to be added/updated
 if value is None, delete the item from input File attributes

Output:

File - string, path/name of updated file

20.11.1.2 [azimuth_ground_resolution\(\)](#)

```
def pysar._pysar_utilities.azimuth_ground_resolution (
    atr )
```

Get azimuth resolution on the ground in meters, from ROI_PAC attributes, for file in radar coord

20.11.1.3 Bh_Bv_timeseries()

```
def pysar._pysar_utilities.Bh_Bv_timeseries (
    ifgramFile )
```

20.11.1.4 check_drop_ifgram()

```
def pysar._pysar_utilities.check_drop_ifgram (
    h5,
    print_msg = True )
```

Update ifgram_list based on 'drop_ifgram' attribute

Input:

h5 - HDF5 file object

Output:

dsListOut - list of string, group name with drop_ifgram = 'yes'

Example:

```
h5 = h5py.File('unwrapIfgram.h5','r')
ifgram_list = ut.check_drop_ifgram(h5)
```

20.11.1.5 check_file_size()

```
def pysar._pysar_utilities.check_file_size (
    fname_list,
    mode_width = None,
    mode_length = None )
```

Check file size in the list of files, and drop those not in the same size with majority.

20.11.1.6 check_loaded_dataset()

```
def pysar._pysar_utilities.check_loaded_dataset (
    work_dir = './',
    inps = None,
    print_msg = True )
```

Check the result of loading data for the following two rules:

1. file existence
2. file attribute readability

If inps is valid/not_empty: return updated inps;

Otherwise, return True/False if all recommended file are loaded and readably or not

Inputs:

work_dir : string, PySAR working directory

inps : Namespace, optional, variable for pysarApp.py. Not needed for check loading result.

Outputs:

load_complete : bool, complete loading or not

ifgram_file : string, file name/path of unwrapped interferograms

coherence_file : string, file name/path of spatial coherence

dem_file_radar : string, file name/path of DEM file in radar coord (for interferograms in radar coord)

dem_file_geo : string, file name/path of DEM file in geo coord

lookup_file : string, file name/path of lookup table file (for interferograms in radar coord)

Example:

```
from pysar.pysarApp import check_loaded_dataset
```

```
True = check_loaded_dataset($SCRATCHDIR+'SinabungT495F50AlosA/PYSAR') #if True, PROCESS, SLC folder could
```

```
inps = check_loaded_dataset(inps.work_dir, inps)
```

20.11.1.7 check_parallel()

```
def pysar._pysar_utilities.check_parallel (
    file_num = 1,
    print_msg = True )
```

Check parallel option based on pysar setting, file num and installed module

Examples:

```
num_cores, inps.parallel, Parallel, delayed = ut.check_parallel(len(inps.file))
num_cores, inps.parallel, Parallel, delayed = ut.check_parallel(1000)
```

20.11.1.8 check_variable_name()

```
def pysar._pysar_utilities.check_variable_name (
    path )
```

20.11.1.9 circle_index()

```
def pysar._pysar_utilities.circle_index (
    atr,
    circle_par )
```

Return Index of Elements within a Circle centered at input pixel

Inputs: atr : dictionary

containing the following attributes:

WIDT

FILE_LENGTH

circle_par : string in the format of 'y,x,radius'

i.e. '200,300,20' for radar coord

'31.0214,130.5699,20' for geo coord

Output: idx : 2D np.array in bool type

mask matrix for those pixel falling into the circle defined by circle_par

Examples: idx_mat = ut.circle_index(atr, '200,300,20')

idx_mat = ut.circle_index(atr, '31.0214,130.5699,20')

20.11.1.10 date_list()

```
def pysar._pysar_utilities.date_list (
    h5file )
```

20.11.1.11 dBh_dBv_timeseries()

```
def pysar._pysar_utilities.dBh_dBv_timeseries (
    ifgramFile )
```

20.11.1.12 design_matrix()

```
def pysar._pysar_utilities.design_matrix (
    ifgramFile = None,
    date12_list = [],
    referenceDate = None,
    zero_first = True )
```

Make the design matrix for the inversion based on date12_list.

Reference:

Berardino, P., Fornaro, G., Lanari, R., & Sansosti, E. (2002).
A new algorithm for surface deformation monitoring based on small
baseline differential SAR interferograms. IEEE TGRS, 40(11), 2375-2383.

Input:

ifgramFile - string, name/path of interferograms file
date12_list - list of string, date12 used in calculation in YYMMDD-YYMMDD format
use all date12 from ifgramFile if input is empty

Outputs:

A - 2D np.array in size of (ifgram_num, date_num-1)
representing date combination for each interferogram (-1 for master, 1 for slave, 0 for others)
B - 2D np.array in size of (ifgram_num, date_num-1)
representing temporal baseline timeseries between master and slave date for each interferogram

20.11.1.13 four_corners()

```
def pysar._pysar_utilities.four_corners (
    atr )
```

Return 4 corners lat/lon

20.11.1.14 generate_curls()

```
def pysar._pysar_utilities.generate_curls (
    curlfile,
    h5file,
    Triangles,
    curls )
```

20.11.1.15 get_file_list()

```
def pysar._pysar_utilities.get_file_list (
    fileList,
    abspath = False,
    coord = None )
```

Get all existed files matching the input list of file pattern

Inputs:

fileList - string or list of string, input file/directory pattern
abspath - bool, return absolute path or not
coord - string, return files with specific coordinate type: geo or radar
if none, skip the checking and return all files

Output:

fileListOut - list of string, existed file path/name, [] if not existed

Example:

```
fileList = get_file_list(['*velocity*.h5', 'timeseries*.h5'])
fileList = get_file_list('timeseries*.h5')
```

20.11.1.16 get_file_stack()

```
def pysar._pysar_utilities.get_file_stack (
    File,
    maskFile = None )
```

Get stack file of input File and return the stack 2D matrix

Input: File/maskFile - string

Output: stack - 2D np.array matrix

20.11.1.17 get_geometry_file()

```
def pysar._pysar_utilities.get_geometry_file (
    dset,
    coordType = None,
    filePattern = None,
    abspath = False,
    print_msg = True )
```

Find geometry file containing input specific dataset

20.11.1.18 get_lookup_file()

```
def pysar._pysar_utilities.get_lookup_file (
    filePattern = None,
    abspath = False,
    print_msg = True )
```

Find lookup table file with/without input file pattern

20.11.1.19 get_lookup_row_col()

```
def pysar._pysar_utilities.get_lookup_row_col (
    y,
    x,
    lut_y,
    lut_x,
    y_factor = 10,
    x_factor = 10,
    geoCoord = False )
```

Use geomap*.trans file for precious (pixel-level) coord conversion.

Get row/col number in y/x value matrix from input y/x

Use overlap mean value between y and x buffer;

To support point outside of value pool/matrix, could use np.polyfit to fit a line for y and x value buffer and return the intersection point row/col

20.11.1.20 get_residual_rms()

```
def pysar._pysar_utilities.get_residual_rms (
    timeseries_resid_file,
    mask_file = 'maskTempCoh.h5',
    ramp_type = 'quadratic' )
```

Calculate deramped Root Mean Square in space for each epoch of input timeseries file.

Inputs:

timeseries_resid_file - string, timeseries HDF5 file, e.g. timeseries_ECMWF_demErrInvResid.h5
 mask_file - string, mask file, e.g. maskTempCoh.h5
 ramp_type - string, ramp type, e.g. plane, quadratic, no for do not remove ramp

outputs:

rms_list - list of float, Root Mean Square of deramped input timeseries file
 date_list - list of string in YYYYMMDD format, corresponding dates

Example:

```
import pysar._pysar_utilities as ut
rms_list, date_list = ut.get_residual_rms('timeseriesResidual.h5', 'maskTempCoh.h5')
```

20.11.1.21 get_residual_std()

```
def pysar._pysar_utilities.get_residual_std (
    timeseries_resid_file,
    mask_file = 'maskTempCoh.h5',
    ramp_type = 'quadratic' )
```

Calculate deramped standard deviation in space for each epoch of input timeseries file.

Inputs:

timeseries_resid_file - string, timeseries HDF5 file, e.g. timeseries_ECMWF_demErrInvResid.h5
 mask_file - string, mask file, e.g. maskTempCoh.h5
 ramp_type - string, ramp type, e.g. plane, quadratic, no for do not remove ramp

outputs:

std_list - list of float, standard deviation of deramped input timeseries file
 date_list - list of string in YYYYMMDD format, corresponding dates

Example:

```
import pysar._pysar_utilities as ut
std_list, date_list = ut.get_residual_std('timeseries_ECMWF_demErrInvResid.h5', 'maskTempCoh.h5')
```

20.11.1.22 get_triangles()

```
def pysar._pysar_utilities.get_triangles (
    h5file )
```

20.11.1.23 glob2radar()

```
def pysar._pysar_utilities.glob2radar (
    lat,
    lon,
    lookupFile = None,
    atr_rdr = dict(),
    print_msg = True )
```

Convert geo coordinates into radar coordinates.

Inputs:

lat/lon - np.array, float, latitude/longitude
 lookupFile - string, trans/look up file
 atr_rdr - dict, attributes of file in radar coord, optional but recommended.

Output:

az/rg - np.array, float, range/azimuth pixel number
 az/rg_res - float, residul/uncertainty of coordinate conversion

20.11.1.24 hillshade()

```
def pysar._pysar_utilities.hillshade (
    data,
    scale )
```

from scott baker, ptisk library

20.11.1.25 incidence_angle()

```
def pysar._pysar_utilities.incidence_angle (
    atr,
    dimension = 2,
    print_msg = True )
```

Calculate 2D matrix of incidence angle from ROI_PAC attributes, very accurate.

Input:

```
    dictionary - ROI_PAC attributes including the following items:
        STARTING_RANGE
        RANGE_PIXEL_SIZE
        EARTH_RADIUS
        HEIGHT
        FILE_LENGTH
        WIDTH
    dimension - int,
        2 for 2d matrix
        1 for 1d array
        0 for one center value
```

Output: 2D np.array - incidence angle in degree for each pixel

20.11.1.26 is_file_exist()

```
def pysar._pysar_utilities.is_file_exist (
    file_list,
    abspath = True )
```

Check if any file in the file list 1) exists and 2) readable

Inputs:

```
    file_list : list of string, file name with/without wildcards
    abspath   : bool, return absolute file name/path or not
```

Output:

```
    file_path : string, found file name/path; None if not.
```

20.11.1.27 make_triangle()

```
def pysar._pysar_utilities.make_triangle (
    dates12,
    igram1,
    igram2,
    igram3 )
```

20.11.1.28 mode()

```
def pysar._pysar_utilities.mode (
    thelist )
```

Find Mode (most common) item in the list

20.11.1.29 nonzero_mask()

```
def pysar._pysar_utilities.nonzero_mask (
    File,
    outFile = 'mask.h5' )
```

Generate mask file for non-zero value of input multi-group hdf5 file

20.11.1.30 normalize_timeseries()

```
def pysar._pysar_utilities.normalize_timeseries (
    ts_mat,
    nanValue = 0 )
```

Normalize timeseries of 2D matrix in time domain

20.11.1.31 normalize_timeseries_old()

```
def pysar._pysar_utilities.normalize_timeseries_old (
    ts_mat,
    nanValue = 0 )
```

20.11.1.32 perp_baseline_ifgram2timeseries()

```
def pysar._pysar_utilities.perp_baseline_ifgram2timeseries (
    ifgramFile,
    ifgram_list = [] )
```

Calculate perpendicular baseline timeseries from input interferograms file

Input:

```
ifgramFile - string, file name/path of interferograms file
ifgram_list - list of string, group name that is used for calculation
              use all if it's empty
```

Outputs:

```
pbase          - 1D np.array, P_BASELINE_TIMESERIES
pbase_top      - 1D np.array, P_BASELINE_TOP_TIMESERIES
pbase_bottom   - 1D np.array, P_BASELINE_BOTTOM_TIMESERIES
```

20.11.1.33 perp_baseline_timeseries()

```
def pysar._pysar_utilities.perp_baseline_timeseries (
    atr,
    dimension = 1 )
```

Calculate perpendicular baseline for each acquisition within timeseries

Inputs:

```
atr - dict, including the following PySAR attribute
    FILE_LENGTH
    P_BASELINE_TIMESERIES
    P_BASELINE_TOP_TIMESERIES (optional)
    P_BASELINE_BOTTOM_TIMESERIES (optional)
dimension - int, choices = [0, 1]
    0 for constant P_BASELINE in azimuth direction
    1 for linear P_BASELINE in azimuth direction, for radar coord only
```

Output:

```
pbase - np.array, with shape = [date_num, 1] or [date_num, length]
```

20.11.1.34 radar2glob()

```
def pysar._pysar_utilities.radar2glob (
    az,
    rg,
    lookupFile = None,
    atr_rdr = dict(),
    print_msg = True )
```

Convert radar coordinates into geo coordinates

Inputs:

```
rg/az - np.array, int, range/azimuth pixel number
lookupFile - string, trans/look up file
atr_rdr - dict, attributes of file in radar coord, optional but recommended.
```

Output:

```
lon/lat - np.array, float, longitude/latitude of input point (rg,az); nan if not found.
latlon_res - float, residul/uncertainty of coordinate conversion
```

20.11.1.35 range_distance()

```
def pysar._pysar_utilities.range_distance (
    atr,
    dimension = 2 )
```

Calculate range distance from input attribute dict

Inputs:

```
atr - dict, including the following ROI_PAC attributes:
    STARTING_RANGE
    RANGE_PIXEL_SIZE
    FILE_LENGTH
    WIDTH
dimension - int, choices = [0,1,2]
    2 for 2d matrix, vary in range direction, constant in az direction, for radar coord only
    1 for 1d matrix, in range direction, for radar coord file
    0 for center value
```

Output: np.array (0, 1 or 2 D) - range distance between antenna and ground target in meters

20.11.1.36 range_ground_resolution()

```
def pysar._pysar_utilities.range_ground_resolution (
    atr,
    print_msg = False )
```

Get range resolution on the ground in meters, from ROI_PAC attributes, for file in radar coord

20.11.1.37 spatial_average()

```
def pysar._pysar_utilities.spatial_average (
    File,
    maskFile = None,
    box = None,
    saveList = False,
    checkAoi = True )
```

Read/Calculate Spatial Average of input file.

If input file is text file, read it directly;

If input file is data matrix file:

 If corresponding text file exists with the same mask file/AOI info, read it directly;
 Otherwise, calculate it from data file.

 Only non-nan pixel is considered.

Input:

 File : string, path of input file
 maskFile : string, path of mask file, e.g. maskTempCoh.h5
 box : 4-tuple defining the left, upper, right, and lower pixel coordinate
 saveList : bool, save (list of) mean value into text file

Output:

 mean_list : list for float, average value in space for each epoch of input file
 date_list : list of string for date info
 date12_list, e.g. 101120-110220, for interferograms/coherence
 date8_list, e.g. 20101120, for timeseries
 file name, e.g. velocity.h5, for all the other file types

Example:

```
mean_list = spatial_average('coherence.h5')[0]
ref_list = spatial_average('unwrapIfgram.h5', box=(100,200,101,201))[0]
mean_list, date12_list = spatial_average('coherence.h5', 'maskTempCoh.h5', saveList=True)

stack = ut.get_file_stack('unwrapIfgram.h5', 'mask.h5')
mask = ~np.isnan(stack)
ref_list = ut.spatial_average('unwrapIfgram.h5', mask, (100,200,101,201))
```

20.11.1.38 stacking()

```
def pysar._pysar_utilities.stacking (
    File )
```

Stack multi-temporal dataset into one equivalent to temporal sum
For interferograms, the averaged velocity is calculated.

20.11.1.39 temporal_average()

```
def pysar._pysar_utilities.temporal_average (
    File,
    outFile = None )
```

Calculate temporal average.

20.11.1.40 timeseries_coherence()

```
def pysar._pysar_utilities.timeseries_coherence (
    inFile,
    maskFile = 'maskTempCoh.h5',
    outFile = None )
```

Calculate spatial average coherence for each epoch of input time series file

Inputs:

```
inFile    - string, timeseries HDF5 file
maskFile  - string, mask file
outFile   - string, output text file
```

Example:

```
txtFile = timeseries_coherence('timeseries_ECMWF_demErrInvResid_quadratic.h5')
```

20.11.1.41 timeseries_inversion_FGLS()

```
def pysar._pysar_utilities.timeseries_inversion_FGLS (
    h5flat,
    h5timeseries )
```

Implementation of the SBAS algorithm.

Usage:

```
timeseries_inversion(h5flat,h5timeseries)
  h5flat: hdf5 file with the interferograms
  h5timeseries: hdf5 file with the output from the inversion
#####
```

20.11.1.42 timeseries_inversion_L1()

```
def pysar._pysar_utilities.timeseries_inversion_L1 (
    h5flat,
    h5timeseries )
```

20.11.1.43 timeseries_rms()

```
def pysar._pysar_utilities.timeseries_rms (
    inFile,
    maskFile = 'maskTempCoh.h5',
    outFile = None,
    dimension = 2 )
```

Calculate the Root Mean Square for each epoch of input timeseries file and output result to a text file.

20.11.1.44 timeseries_std()

```
def pysar._pysar_utilities.timeseries_std (
    inFile,
    maskFile = 'maskTempCoh.h5',
    outFile = None )
```

Calculate the standard deviation for each epoch of input timeseries file and output result to a text file.

20.11.1.45 touch()

```
def pysar._pysar_utilities.touch (
    fname_list,
    times = None )
```

python equivalent function to Unix utility - touch

It sets the modification and access times of files to the current time of day. If the file doesn't exist, it is created with default permissions.

Inputs/Output:

fname_list - string / list of string

20.11.1.46 update_attribute_or_not()

```
def pysar._pysar_utilities.update_attribute_or_not (
    atr_new,
    atr_orig,
    update = False )
```

Compare new attributes with existing ones

20.11.1.47 update_file()

```
def pysar._pysar_utilities.update_file (
    outFile,
    inFile = None,
    overwrite = False,
    check_readable = True )
```

Check whether to update outFile/outDir or not.

return True if any of the following meets:

1. if overwrite option set to True
2. outFile is empty, e.g. None, []
3. outFile is not existed
4. outFile is not readable by readfile.read_attribute() when check_readable=True
5. outFile is older than inFile, if inFile is not None

Otherwise, return False.

If inFile=None and outFile exists and readable, return False

Inputs:

inFile - string or list of string, input file(s)/directories

Output:

True/False - bool, whether to update output file or not

Example:

```
if ut.update_file('timeseries_ECMWF_demErr.h5', 'timeseries_ECMWF.h5'):
    if ut.update_file('exclude_date.txt', ['timeseries_ECMWF_demErrInvResid.h5', 'maskTempCoh.h5', 'pysar_template.h5',
                                           'timeseries_ECMWF_demErr.h5', 'timeseries_ECMWF.h5'],
                      check_readable=False):
```

20.11.1.48 update_template_file()

```
def pysar._pysar_utilities.update_template_file (
    template_file,
    extra_dict )
```

Update option value in template_file with value from input extra_dict

20.11.1.49 which()

```
def pysar._pysar_utilities.which (
    program )
```

Test if executable exists

20.11.1.50 yymmdd()

```
def pysar._pysar_utilities.yymmdd (
    dates )
```

20.11.1.51 `yymdd2YYYYMMDD()`

```
def pysar._pysar_utilities.yymdd2YYYYMMDD (
    date )
```

20.11.1.52 `yyyymmdd()`

```
def pysar._pysar_utilities.yyyymmdd (
    dates )
```

20.12 `pysar._readfile` Namespace Reference

Functions

- def `read` (File, box=None, epoch=None, print_msg=True)
- def `read_attribute` (File, epoch=None)
- def `check_variable_name` (path)
- def `is_plot_attribute` (attribute)
- def `read_template` (File, delimiter='=')
- def `read_roipac_rsc` (File)
- def `read_gamma_par` (fname, delimiter=':', skiprows=3, convert2roipac=True)
- def `read_isce_xml` (File)
- def `attribute_gamma2roipac` (par_dict_in)
- def `attribute_isce2roipac` (metaDict, dates=[], baselineDict={})
- def `attribute_envi2roipac` (metaDict)
- def `read_float32` (File, box=None, byte_order='I')
- def `read_real_float64` (fname, box=None, byte_order='I')
- def `read_complex_float32` (fname, box=None, byte_order='I', cpx=False)
- def `read_real_float32` (fname, box=None, byte_order='I')
- def `read_complex_int16` (File, box=None, byte_order='I', cpx=False)
- def `read_real_int16` (File, box=None, byte_order='I')
- def `read_bool` (File, box=None)
- def `read_GPS_USGS` (File)
- def `read_multiple` (File, box="")

Variables

- list `multi_group_hdf5_file` = ['interferograms', 'coherence', 'wrapped', 'snaphu_connect_component']
- list `multi_dataset_hdf5_file` = ['timeseries', 'geometry']
- list `single_dataset_hdf5_file` = ['dem', 'mask', 'rmse', 'temporal_coherence', 'velocity']
- list `geometry_dataset`

20.12.1 Function Documentation

20.12.1.1 attribute_envi2roipac()

```
def pysar._readfile.attribute_envi2roipac (
    metaDict )
```

Convert ISCE xml attribute into ROI_PAC format

20.12.1.2 attribute_gamma2roipac()

```
def pysar._readfile.attribute_gamma2roipac (
    par_dict_in )
```

Convert Gamma par attribute into ROI_PAC format

20.12.1.3 attribute_isce2roipac()

```
def pysar._readfile.attribute_isce2roipac (
    metaDict,
    dates = [],
    baselineDict = {} )
```

Convert ISCE xml attribute into ROI_PAC format

20.12.1.4 check_variable_name()

```
def pysar._readfile.check_variable_name (
    path )
```

20.12.1.5 is_plot_attribute()

```
def pysar._readfile.is_plot_attribute (
    attribute )
```

20.12.1.6 read()

```
def pysar._readfile.read (
    File,
    box = None,
    epoch = None,
    print_msg = True )
```

Read one dataset and its attributes from input file.

Read one dataset, i.e. interferogram, coherence, velocity, dem ...
return 0 if failed.

Inputs:

```
File   : str, path of file to read
PySAR  file: interferograms, timeseries, velocity, etc.
ROI_PAC file: .unw .cor .hgt .dem .trans
Gamma  file: .mli .slc
Image  file: .jpeg .jpg .png .ras .bmp
box    : 4-tuple of int, area to read, defined in (x0, y0, x1, y1) in pixel coordinate
epoch  : string, epoch to read, for multi-dataset files
for .trans file:
    '' - return both dataset
    rg, range - for geomap*.trans file
    az, azimuth - for geomap*.trans file
```

Outputs:

```
data : 2-D matrix in numpy.array format, return None if failed
atr  : dictionary, attributes of data, return None if failed
```

Examples:

```
data, atr = read('velocity.h5')
data, atr = read('100120-110214.unw', box=(100,1100, 500, 2500))
data, atr = read('timeseries.h5', epoch='20101120')
data, atr = read('timeseries.h5', box=(100,1100, 500, 2500), epoch='20101120')
az, atr = read('geomap*.trans', epoch='azimuth')
rg,az,atr = read('geomap*.trans')
```

20.12.1.7 read_attribute()

```
def pysar._readfile.read_attribute (
    File,
    epoch = None )
```

Read attributes of input file into a dictionary

Input : string, file name and epoch (optional)

Output : dictionary, attributes dictionary

20.12.1.8 read_bool()

```
def pysar._readfile.read_bool (
    File,
    box = None )
```

Read binary file with flags, 1-byte values with flags set in bits
For ROI_PAC .flg, *_snap_connect.byf file.

20.12.1.9 read_complex_float32()

```
def pysar._readfile.read_complex_float32 (
    fname,
    box = None,
    byte_order = 'l',
    cpx = False )
```

Read complex float 32 data matrix, i.e. roi_pac int or slc data.
old name: read_complex64()

ROI_PAC file: .slc, .int, .amp

Data is sotred as:

```
real, imaginary, real, imaginary, ...
real, imaginary, real, imaginary, ...
...
```

Inputs:

```
fname      : str, input file name
box        : 4-tuple defining (left, upper, right, lower) pixel coordinate.
byte_order : str, optional, order of reading byte in the file
cpx        : flag for output format,
              0 for amplitude and phase [by default],
              non-0 : for real and imagery
```

Output:

```
data : 2D np.array in complex float32
```

Example:

```
amp, phase, atr = read_complex_float32('geo_070603-070721_0048_00018.int')
data, atr       = read_complex_float32('150707.slc', 1)
```

20.12.1.10 read_complex_int16()

```
def pysar._readfile.read_complex_int16 (
    File,
    box = None,
    byte_order = 'l',
    cpx = False )
```

Read complex int 16 data matrix, i.e. GAMMA SCOMPLEX file (.slc)

Gamma file: .slc

Inputs:

```
file: complex data matrix (cpx_int16)
box: 4-tuple defining the left, upper, right, and lower pixel coordinate.
```

Example:

```
data,rsc = read_complex_int16('100102.slc')
data,rsc = read_complex_int16('100102.slc', (100,1200,500,1500))
```

20.12.1.11 read_float32()

```
def pysar._readfile.read_float32 (
    File,
    box = None,
    byte_order = 'l' )
```

Reads roi_pac data (RMG format, interleaved line by line)
should rename it to read_rmg_float32()

ROI_PAC file: .unw, .cor, .hgt, .trans, .msk

RMG format (named after JPL radar pionner Richard M. Goldstein): made up of real*4 numbers in two arrays side-by-side. The two arrays often show the magnitude of the radar image and the phase, although not always (sometimes the phase is the correlation). The length and width of each array are given as lines in the metadata (.rsc) file. Thus the total width width of the binary file is (2*width) and length is (length), data are stored as:

```
magnitude, magnitude, magnitude, ...,phase, phase, phase, ...
magnitude, magnitude, magnitude, ...,phase, phase, phase, ...
.....
```

box : 4-tuple defining the left, upper, right, and lower pixel coordinate.
Example:

```
a,p,r = read_float32('100102-100403.unw')
a,p,r = read_float32('100102-100403.unw', (100,1200,500,1500))
```

20.12.1.12 read_gamma_par()

```
def pysar._readfile.read_gamma_par (
    fname,
    delimiter = ':',
    skiprows = 3,
    convert2roipac = True )
```

Read GAMMA .par/.off file into a python dictionary structure.
Parameters: fname : file, str, or path.

File path of .par, .off file.
delimiter : str, optional
String used to separate values.
skiprows : int, optional
Skip the first skiprows lines.

Returns: par_dict : dict
Attributes dictionary

20.12.1.13 read_GPS_USGS()

```
def pysar._readfile.read_GPS_USGS (
    File )
```

20.12.1.14 read_isce_xml()

```
def pysar._readfile.read_isce_xml (
    File )
```

Read ISCE .xml file input a python dictionary structure.

20.12.1.15 read_multiple()

```
def pysar._readfile.read_multiple (
    File,
    box = '' )
```

Read multi-temporal 2D datasets into a 3-D data stack

Inputs:

File : input file, interferograms, coherence, timeseries, ...

box : 4-tuple defining the left, upper, right, and lower pixel coordinate [optional]

Examples:

```
stack = stacking('timeseries.h5', (100, 1200, 500, 1500))
```

20.12.1.16 read_real_float32()

```
def pysar._readfile.read_real_float32 (
    fname,
    box = None,
    byte_order = 'l' )
```

Read real float 32 data matrix, i.e. GAMMA .mli file

Parameters: *fname* : str, path, filename to be read

byte_order : str, optional, order of reading byte in the file

Returns: *data* : 2D np.array, data matrix

atr : dict, attribute dictionary

Usage: *data*, *atr* = read_real_float32('20070603.mli')

data, *atr* = read_real_float32('diff_filt_130118-130129_4rlks.unw')

20.12.1.17 read_real_float64()

```
def pysar._readfile.read_real_float64 (
    fname,
    box = None,
    byte_order = 'l' )
```

Read real float64/double data matrix, i.e. isce lat/lon.rdr

20.12.1.18 read_real_int16()

```
def pysar._readfile.read_real_int16 (
    File,
    box = None,
    byte_order = 'l' )
```

20.12.1.19 read_roipac_rsc()

```
def pysar._readfile.read_roipac_rsc (
    File )
```

Read ROI_PAC .rsc file into a python dictionary structure.

20.12.1.20 read_template()

```
def pysar._readfile.read_template (
    File,
    delimiter = '=' )
```

Reads the template file into a python dictionary structure.

Input : string, full path to the template file

Output: dictionary, pysar template content

Example:

```
tmpl = read_template(KyushuT424F610_640AlosA.template)
tmpl = read_template(R1_54014_ST5_L0_F898.000.pi, ':')
```

20.12.2 Variable Documentation**20.12.2.1 geometry_dataset**

```
list geometry_dataset
```

Initial value:

```
1 = ['rangeCoord', 'azimuthCoord', 'latitude', 'longitude', 'height', \
2     'incidenceAngle', 'headingAngle', 'slantRangeDistance', 'waterMask', 'shadowMask']
```

20.12.2.2 multi_dataset_hdf5_file

```
list multi_dataset_hdf5_file = ['timeseries', 'geometry']
```

20.12.2.3 multi_group_hdf5_file

```
list multi_group_hdf5_file = ['interferograms','coherence','wrapped','snaphu_connect_component']
```

20.12.2.4 single_dataset_hdf5_file

```
list single_dataset_hdf5_file = ['dem','mask','rmse','temporal_coherence','velocity']
```

20.13 pysar._remove_surface Namespace Reference

Functions

- def [remove_data_surface](#) (data, mask, surf_type='plane')
- def [remove_data_multiple_surface](#) (data, mask, surf_type, ysub)
- def [remove_surface](#) (File, surf_type, maskFile=None, outFile=None, ysub=None)

20.13.1 Function Documentation

20.13.1.1 remove_data_multiple_surface()

```
def pysar._remove_surface.remove_data_multiple_surface (
    data,
    mask,
    surf_type,
    ysub )
```

20.13.1.2 remove_data_surface()

```
def pysar._remove_surface.remove_data_surface (
    data,
    mask,
    surf_type = 'plane' )
```

Remove surface from input data matrix based on pixel marked by mask

20.13.1.3 remove_surface()

```
def pysar._remove_surface.remove_surface (
    File,
    surf_type,
    maskFile = None,
    outFile = None,
    ysub = None )
```

20.14 pysar._sensor Namespace Reference

Classes

- class [JERS](#)

Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang #.

20.15 pysar._variance Namespace Reference

Functions

- def [get_lat_lon](#) (atr)
- def [sample_data](#) (lat, lon, mask=None, num_sample=500)
- def [get_distance](#) (lat, lon, i)
- def [structure_function](#) (data, lat, lon, step=5e3, min_pair_num=100e3, print_msg=True)
- def [bin_variance](#) (distance, variance, step=5e3, min_pair_num=100e3, print_msg=True)

20.15.1 Function Documentation

20.15.1.1 bin_variance()

```
def pysar._variance.bin_variance (
    distance,
    variance,
    step = 5e3,
    min_pair_num = 100e3,
    print_msg = True )
```

20.15.1.2 get_distance()

```
def pysar._variance.get_distance (
    lat,
    lon,
    i )
```

Return the distance of all points in lat/lon from its ith point

20.15.1.3 get_lat_lon()

```
def pysar._variance.get_lat_lon (
    atr )
```

Get lat/lon of all pixels

20.15.1.4 sample_data()

```
def pysar._variance.sample_data (
    lat,
    lon,
    mask = None,
    num_sample = 500 )
```

20.15.1.5 structure_function()

```
def pysar._variance.structure_function (
    data,
    lat,
    lon,
    step = 5e3,
    min_pair_num = 100e3,
    print_msg = True )
```

20.16 pysar._writefile Namespace Reference

Functions

- def [write](#) (args)
- def [write_roipac_rsc](#) (atr, outname, sorting=True)
- def [write_float32](#) (args)
- def [write_complex64](#) (data, outname)
- def [write_real_int16](#) (data, outname)
- def [write_dem](#) (data, outname)
- def [write_real_float32](#) (data, outname)
- def [write_complex_int16](#) (data, outname)

20.16.1 Function Documentation

20.16.1.1 write()

```
def pysar._writefile.write (
    args )
```

Write one dataset, i.e. interferogram, coherence, velocity, dem ...
Return 0 if failed.

Usage:

```
write(data,atr,outname)
write(rg,az,atr,outname)
```

Inputs:

```
data : 2D data matrix
atr  : attribute object
outname : output file name
```

Output:

```
output file name
```

Examples:

```
write(data,atr,'velocity.h5')
write(data,atr,'temporal_coherence.h5')
write(data,atr,'100120-110214.unw')
write(data,atr,'strml.dem')
write(data,atr,'100120.mli')
write(rg,az,atr,'geomap_4lks.trans')
```

20.16.1.2 write_complex64()

```
def pysar._writefile.write_complex64 (
    data,
    outname )
```

Writes roi_pac .int data

20.16.1.3 write_complex_int16()

```
def pysar._writefile.write_complex_int16 (
    data,
    outname )
```

Write gamma scomplex data, i.e. .slc file.
data is complex 2-D matrix
real, imagery, real, ...

20.16.1.4 write_dem()

```
def pysar._writefile.write_dem (
    data,
    outname )
```

20.16.1.5 write_float32()

```
def pysar._writefile.write_float32 (
    args )
```

Write ROI_PAC rmg format with float32 precision
 Format of the binary file is same as roi_pac unw, cor, or hgt data.
 should rename to write_rmg_float32()

Exmaple:

```
write_float32(phase, outname)
write_float32(amp, phase, outname)
```

20.16.1.6 write_real_float32()

```
def pysar._writefile.write_real_float32 (
    data,
    outname )
```

write gamma float data, i.e. .mli file.

20.16.1.7 write_real_int16()

```
def pysar._writefile.write_real_int16 (
    data,
    outname )
```

20.16.1.8 write_roipac_rsc()

```
def pysar._writefile.write_roipac_rsc (
    atr,
    outname,
    sorting = True )
```

Write attribute dict into ROI_PAC .rsc file

Inputs:

```
atr      - dict, attributes dictionary
outname  - rsc file name, to which attribute is written
sorting  - bool, sort attributes in alphabetic order while writing
```

Output:

```
outname
```

20.17 pysar.add Namespace Reference**Functions**

- def [add_matrix](#) (data1, data2)
- def [add_files](#) (fname_list, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- string [EXAMPLE](#)

20.17.1 Function Documentation

20.17.1.1 add_files()

```
def pysar.add.add_files (
    fname_list,
    fname_out = None )
```

Generate sum of all input files

Inputs:

fname_list - list of string, path/name of input files to be added
fname_out - string, optional, path/name of output file

Output:

fname_out - string, path/name of output file

Example:

```
'mask_all.h5' = add_file(['mask_1.h5','mask_2.h5','mask_3.h5'], 'mask_all.h5')
```

20.17.1.2 add_matrix()

```
def pysar.add.add_matrix (
    data1,
    data2 )
```

Sum of 2 input matrix

20.17.1.3 cmdLineParse()

```
def pysar.add.cmdLineParse ( )
```

20.17.1.4 main()

```
def pysar.add.main (
    argv )
```

20.17.2 Variable Documentation

20.17.2.1 EXAMPLE

```
string EXAMPLE
```

Initial value:

```
1 = '''example:
2   add.py mask_1.h5 mask_2.h5 mask_3.h5           -o mask_all.h5
3   add.py 081008_100220.unw 100220_110417.unw      -o 081008_110417.unw
4   add.py timeseries_ECMWF.h5 ECMWF.h5            -o timeseries.h5
5 '''
```

20.18 pysar.add_attribute Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.18.1 Function Documentation

20.18.1.1 main()

```
def pysar.add_attribute.main (
    argv )
```

20.18.1.2 usage()

```
def pysar.add_attribute.usage ( )
```

20.19 pysar.add_attribute_insarmaps Namespace Reference

Classes

- class [InsarDatabaseController](#)
- class [InsarDatasetController](#)

Functions

- def [build_parser](#) ()
- def [main](#) (argv)

20.19.1 Function Documentation

20.19.1.1 build_parser()

```
def pysar.add_attribute_insarmaps.build_parser ( )
```

20.19.1.2 main()

```
def pysar.add_attribute_insarmaps.main (
    argv )
```

20.20 pysar.asc_desc Namespace Reference

Functions

- def [get_overlap_lalo](#) (atr1, atr2)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [REFERENCE](#)
- [EXAMPLE](#)

20.20.1 Function Documentation

20.20.1.1 cmdLineParse()

```
def pysar.asc_desc.cmdLineParse ( )
```

20.20.1.2 get_overlap_lalo()

```
def pysar.asc_desc.get_overlap_lalo (
    atr1,
    atr2 )
```

Find overlap area in lat/lon of two geocoded files

Inputs:

atr1/2 - dict, attribute dictionary of two input files in geo coord

Outputs:

W/E/S/N - float, West/East/South/North in deg

20.20.1.3 main()

```
def pysar.asc_desc.main (
    argv )
```

20.20.2 Variable Documentation

20.20.2.1 EXAMPLE

EXAMPLE

20.20.2.2 REFERENCE

REFERENCE

20.21 pysar.baseline_error Namespace Reference

Functions

- def [to_percent](#) (y, position)
- def [usage](#) ()
- def [main](#) (argv)

20.21.1 Function Documentation

20.21.1.1 main()

```
def pysar.baseline_error.main (
    argv )
```

20.21.1.2 to_percent()

```
def pysar.baseline_error.to_percent (
    y,
    position )
```

20.21.1.3 usage()

```
def pysar.baseline_error.usage ( )
```

20.22 pysar.baseline_trop Namespace Reference

Functions

- def [to_percent](#) (y, position)
- def [usage](#) ()
- def [main](#) (argv)

20.22.1 Function Documentation

20.22.1.1 main()

```
def pysar.baseline_trop.main (
    argv )
```

20.22.1.2 to_percent()

```
def pysar.baseline_trop.to_percent (
    y,
    position )
```

20.22.1.3 usage()

```
def pysar.baseline_trop.usage ( )
```

20.23 pysar.coord_glob2radar Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.23.1 Function Documentation

20.23.1.1 main()

```
def pysar.coord_glob2radar.main (
    argv )
```


20.23.1.2 usage()

```
def pysar.coord_glob2radar.usage ( )
```

20.24 pysar.coord_radar2glob Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.24.1 Function Documentation

20.24.1.1 main()

```
def pysar.coord_radar2glob.main (
    argv )
```

20.24.1.2 usage()

```
def pysar.coord_radar2glob.usage ( )
```

20.25 pysar.correct_dem Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.25.1 Function Documentation

20.25.1.1 main()

```
def pysar.correct_dem.main (
    argv )
```

20.25.1.2 usage()

```
def pysar.correct_dem.usage ( )
```

20.26 pysar.correlation_with_dem Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.26.1 Function Documentation

20.26.1.1 [main\(\)](#)

```
def pysar.correlation_with_dem.main (
    argv )
```

20.26.1.2 [usage\(\)](#)

```
def pysar.correlation_with_dem.usage ( )
```

20.27 pysar.dem_error Namespace Reference

Functions

- def [topographic_residual_inversion](#) (ts0, A0, inps)
- def [read_template2inps](#) (template_file, inps=None)
- def [check_exclude_date](#) (exDateIn, dateList)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)
- [REFERENCE](#)

20.27.1 Function Documentation

20.27.1.1 check_exclude_date()

```
def pysar.dem_error.check_exclude_date (
    exDateIn,
    dateList )
```

Read exclude dates info

Inputs:

exDateIn - list of string, date in YYMMDD or YYYYMMDD format,
or text file with date in it
dateList - list of string, date in YYYYMMDD format

Output:

exDateOut - list of string, date in YYYYMMDD format

20.27.1.2 cmdLineParse()

```
def pysar.dem_error.cmdLineParse ( )
```

20.27.1.3 main()

```
def pysar.dem_error.main (
    argv )
```

20.27.1.4 read_template2inps()

```
def pysar.dem_error.read_template2inps (
    template_file,
    inps = None )
```

Read input template file into inps.ex_date

20.27.1.5 topographic_residual_inversion()

```
def pysar.dem_error.topographic_residual_inversion (
    ts0,
    A0,
    inps )
```

Inputs:

ts0 - 2D np.array in size of (date_num, pixel_num), original time series displacement
A0 - 2D np.array in size of (date_num, model_num), design matrix in [A_deltaZ, A_def]
inps - Namespace with the following settings:
tbase - 2D np.array in size of (date_num, 1), temporal baseline
date_flag - 1D np.array in bool data type, mark the date used in the estimation
phase_velocity - bool, use phase history or phase velocity for minimization

Outputs:

deltaZ - 2D np.array in size of (1, pixel_num), estimated DEM residual
tsCor - 2D np.array in size of (date_num, pixel_num), corrected timeseries = tsOrig - topoRes
tsRes - 2D np.array in size of (date_num, pixel_num), residual timeseries = tsOrig - topoRes - defModel
stepEst - 2D np.array in size of (step_num, pixel_num), estimated step deformation

Example:

```
deltaZ, tsCor, tsRes = topographic_residual_inversion(ts, A, inps)
```

20.27.2 Variable Documentation

20.27.2.1 EXAMPLE

EXAMPLE

20.27.2.2 REFERENCE

REFERENCE

20.27.2.3 TEMPLATE

TEMPLATE

20.28 pysar.diff Namespace Reference

Functions

- def [diff_data](#) (data1, data2)
- def [diff_file](#) (file1, file2, outName=None, force=False)
- def [usage](#) ()
- def [cmdLineParse](#) ()
- def [main](#) (argv)

20.28.1 Function Documentation

20.28.1.1 cmdLineParse()

```
def pysar.diff.cmdLineParse ( )
```

20.28.1.2 diff_data()

```
def pysar.diff.diff_data (
    data1,
    data2 )
```

```
data1 - data2
```

20.28.1.3 diff_file()

```
def pysar.diff.diff_file (
    file1,
    file2,
    outName = None,
    force = False )
```

Subtraction/difference of two input files

20.28.1.4 main()

```
def pysar.diff.main (
    argv )
```

20.28.1.5 usage()

```
def pysar.diff.usage ( )
```

20.29 pysar.gamma_view Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.29.1 Function Documentation

20.29.1.1 main()

```
def pysar.gamma_view.main (
    argv )
```

20.29.1.2 usage()

```
def pysar.gamma_view.usage ( )
```

20.30 pysar.generate_mask Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.30.1 Function Documentation

20.30.1.1 cmdLineParse()

```
def pysar.generate_mask.cmdLineParse ( )
```

20.30.1.2 main()

```
def pysar.generate_mask.main (
    argv )
```

20.30.2 Variable Documentation

20.30.2.1 EXAMPLE

EXAMPLE

20.31 pysar.geocode Namespace Reference

Functions

- def [geocode_output_filename](#) (fname)
- def [update_attribute_geo_lut](#) (atr_rdr, atr_lut, print_msg=True)

Geocoded with lut in geo coord #####
- def [geocode_file_geo_lut](#) (fname, lookup_file, fname_out, inps)
- def [interp_weights](#) (xy, uv, d=2)

Geocoded with lut in radar coord ##### Reference: <https://stackoverflow.com/questions/20915502/speedup-scipy-griddata-for-multiple-interpolations-between-two-irregular-grids>
- def [interpolate](#) (values, vtx, wts, fill_value=np.nan)
- def [update_attribute_radar_lut](#) (atr_rdr, inps, lat=None, lon=None, print_msg=True)
- def [geocode_file_radar_lut](#) (fname, lookup_file, fname_out=None, inps=None)
- def [geocode_file](#) (fname, lookup_file, fname_out, inps)
- def [read_template2inps](#) (template_file, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

20.31.1 Function Documentation

20.31.1.1 cmdLineParse()

```
def pysar.geocode.cmdLineParse ( )
```

20.31.1.2 geocode_file()

```
def pysar.geocode.geocode_file (
    fname,
    lookup_file,
    fname_out,
    inps )
```

Geocode input file with lookup table file

20.31.1.3 geocode_file_geo_lut()

```
def pysar.geocode.geocode_file_geo_lut (
    fname,
    lookup_file,
    fname_out,
    inps )
```

Geocode file using ROI_PAC/Gamma lookup table file.

Related module: `scipy.interpolate.RegularGridInterpolator`

Inputs:

```
fname      : string, file to be geocoded
lookup_file : string, optional, lookup table file generated by ROI_PAC or Gamma
              i.e. geomap_4rlks.trans      from ROI_PAC
              sim_150911-150922.UTM_TO_RDC from Gamma
interp_method : string, optional, interpolation/resampling method, supporting nearest, linear
fill_value : value used for points outside of the interpolation domain.
fname_out  : string, optional, output geocoded filename
```

Output:

```
fname_out : string, optional, output geocoded filename
```

20.31.1.4 geocode_file_radar_lut()

```
def pysar.geocode.geocode_file_radar_lut (
    fname,
    lookup_file,
    fname_out = None,
    inps = None )
```

Geocode file using lookup table file in radar coordinates (isce).

Two solutions:

- 1) `scipy.interpolate.griddata`, with a speed up solution from Jaime and Jeff (Stack Overflow) <https://stackoverflow.com/questions/20915502/speedup-scipy-griddata-for-multiple-interpolations-between-two-irregular-grids>
- 2) `matplotlib.tri`, interpolation from triangular grid to quad grid, which is much slower than 1).

Inputs:

```
fname      : string, file to be geocoded
lookup_file : string, lookup table file, geometryRadar.h5
fname_out   : string, optional, output geocoded filename
inps       : namespace, object with the following items:
              interp_method : string, interpolation/resampling method, supporting linear
              fill_value    : value used for points outside of the interpolation domain
```

Output:

```
fname_out   : string, optional, output geocoded filename
```

20.31.1.5 geocode_output_filename()

```
def pysar.geocode.geocode_output_filename (
    fname )
```

20.31.1.6 interp_weights()

```
def pysar.geocode.interp_weights (
    xy,
    uv,
    d = 2 )
```

Geocoded with lut in radar coord ##### Reference: <https://stackoverflow.com/questions/20915502/speedup-scipy-griddata-for-multiple-interpolations-between-two-irregular-grids>.

calculate triangulation and coordinates transformation using `qhull.Delaunay`

- 1) Triangulate the irregular grid coordinates `xy`;
- 2) For each point in the new grid `uv`, search which simplex does it lay
- 3) Calculate barycentric coordinates with respect to the vertices of enclosing simplex

20.31.1.7 interpolate()

```
def pysar.geocode.interpolate (
    values,
    vtx,
    wts,
    fill_value = np.nan )
```

Interpolate values on new points

20.31.1.8 main()

```
def pysar.geocode.main (
    argv )
```

20.31.1.9 read_template2inps()

```
def pysar.geocode.read_template2inps (
    template_file,
    inps )
```

Read input template options into Namespace inps

20.31.1.10 update_attribute_geo_lut()

```
def pysar.geocode.update_attribute_geo_lut (
    atr_rdr,
    atr_lut,
    print_msg = True )
```

Geocoded with lut in geo coord #####.

Get attributes in geo coord from atr_rdr dict and atr_lut dict
Inputs:

atr_rdr : dict, attributes of file in radar coord
atr_lut : dict, attributes of mapping transformation file
print_msg : bool, print out message or not

Output:

atr : dict, attributes of output file in geo coord.

20.31.1.11 update_attribute_radar_lut()

```
def pysar.geocode.update_attribute_radar_lut (
    atr_rdr,
    inps,
    lat = None,
    lon = None,
    print_msg = True )

Get attributes in geo coord from atr_rdr dict and geo_data matrix
Inputs:
    atr_rdr - dict, attribute of file in radar coord
    inps    - Namespace, including items of the following:
        lat0/lon0
        lat_step/lon_step
        lat_num/lon_num
    lat/lon - 2D np.array of lat/lon value
Output:
    atr - dict, attributes of output file in geo coord.
```

20.31.2 Variable Documentation

20.31.2.1 EXAMPLE

EXAMPLE

20.31.2.2 TEMPLATE

TEMPLATE

20.32 pysar.hdfeos5_2insarmaps Namespace Reference

Functions

- def [get_H5_filename](#) (path)
- def [build_parser](#) ()
- def [main](#) ()

20.32.1 Function Documentation

20.32.1.1 build_parser()

```
def pysar.hdfeos5_2insarmaps.build_parser ( )
```

20.32.1.2 get_H5_filename()

```
def pysar.hdfeos5_2insarmaps.get_H5_filename (
    path )
```

20.32.1.3 main()

```
def pysar.hdfeos5_2insarmaps.main ( )
```

20.33 pysar.hdfeos5_2json_mbtiles Namespace Reference

Functions

- def [get_date](#) (date_string)
- def [get_decimal_date](#) (d)
- def [region_name_from_project_name](#) (project_name)
- def [serialize_dictionary](#) (dictionary, fileName)
- def [convert_data](#) (attributes, decimal_dates, timeseries_datasets, dates, json_path, folder_name)
- def [make_json_file](#) (chunk_num, points, dates, json_path, folder_name)
- def [build_parser](#) ()
- def [main](#) ()

Variables

- [needed_attributes](#)

20.33.1 Function Documentation

20.33.1.1 build_parser()

```
def pysar.hdfeos5_2json_mbtiles.build_parser ( )
```

20.33.1.2 convert_data()

```
def pysar.hdfeos5_2json_mbtiles.convert_data (
    attributes,
    decimal_dates,
    timeseries_datasets,
    dates,
    json_path,
    folder_name )
```

20.33.1.3 get_date()

```
def pysar.hdfeos5_2json_mbtiles.get_date (
    date_string )
```

20.33.1.4 get_decimal_date()

```
def pysar.hdfeos5_2json_mbtiles.get_decimal_date (
    d )
```

20.33.1.5 main()

```
def pysar.hdfeos5_2json_mbtiles.main ( )
```

20.33.1.6 make_json_file()

```
def pysar.hdfeos5_2json_mbtiles.make_json_file (
    chunk_num,
    points,
    dates,
    json_path,
    folder_name )
```

20.33.1.7 region_name_from_project_name()

```
def pysar.hdfeos5_2json_mbtiles.region_name_from_project_name (
    project_name )
```

20.33.1.8 serialize_dictionary()

```
def pysar.hdfeos5_2json_mbtiles.serialize_dictionary (
    dictionary,
    fileName )
```

20.33.2 Variable Documentation**20.33.2.1 needed_attributes**

```
needed_attributes
```

20.34 pysar.ifgram_closure Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.34.1 Function Documentation

20.34.1.1 main()

```
def pysar.ifgram_closure.main (
    argv )
```

20.34.1.2 usage()

```
def pysar.ifgram_closure.usage ( )
```

20.35 pysar.ifgram_inversion Namespace Reference

Functions

- def [phase_pdf_ds](#) (L, coherence=None, phiNum=1000)
- def [phase_variance_ds](#) (L, coherence=None)
- def [phase_variance_ps](#) (L, coherence=None)
- def [coherence2phase_variance_ds](#) (coherence, L=32, print_msg=False)
- def [coherence2fisher_info_index](#) (coherence, L=32, epsilon=1e-4)
- def [round_to_1](#) (x)
- def [ceil_to_1](#) (x)
- def [network_inversion_sbass](#) (B, ifgram, tbase_diff, skipZeroPhase=True)
- def [network_inversion_wls](#) (A, ifgram, weight, skipZeroPhase=True, Astd=None)
- def [temporal_coherence](#) (A, ts, ifgram, weight=None, chunk_size=500)
- def [ifgram_inversion_patch](#) (ifgramFile, coherenceFile, meta, box=None)
- def [ifgram_inversion](#) (ifgramFile='unwrapIfgram.h5', coherenceFile='coherence.h5', meta=None)
- def [write_timeseries_hdf5_file](#) (timeseries, date8_list, atr, timeseriesFile=None)
- def [read_template2inps](#) (template_file, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [TEMPLATE](#)
- [REFERENCE](#)

20.35.1 Function Documentation

20.35.1.1 `ceil_to_1()`

```
def pysar.ifgram_inversion.ceil_to_1 (
    x )
```

Return the most significant digit of input number and ceiling it

20.35.1.2 `cmdLineParse()`

```
def pysar.ifgram_inversion.cmdLineParse ( )
```

20.35.1.3 `coherence2fisher_info_index()`

```
def pysar.ifgram_inversion.coherence2fisher_info_index (
    coherence,
    L = 32,
    epsilon = 1e-4 )
```

Convert coherence to Fisher information index (Seymour & Cumming, 1994, IGARSS)

20.35.1.4 `coherence2phase_variance_ds()`

```
def pysar.ifgram_inversion.coherence2phase_variance_ds (
    coherence,
    L = 32,
    print_msg = False )
```

Convert coherence to phase variance based on DS phase PDF (Tough et al., 1995)

20.35.1.5 ifgram_inversion()

```
def pysar.ifgram_inversion.ifgram_inversion (
    ifgramFile = 'unwrapIfgram.h5',
    coherenceFile = 'coherence.h5',
    meta = None )
```

Implementation of the SBAS algorithm.
modified from sbas.py written by scott baker, 2012

Inputs:

```
ifgramFile    - string, HDF5 file name of the interferograms
coherenceFile - string, HDF5 file name of the coherence
meta          - dict, including the following options:
                  weight_function
                  chunk_size - float, max number of data (ifgram_num*row_num*col_num)
                              to read per loop; to control the memory
```

Output:

```
timeseriesFile - string, HDF5 file name of the output timeseries
tempCohFile    - string, HDF5 file name of temporal coherence
```

Example:

```
meta = dict()
meta['weight_function'] = 'variance'
meta['chunk_size'] = 0.5e9
meta['timeseriesFile'] = 'timeseries_var.h5'
meta['tempCohFile'] = 'temporalCoherence_var.h5'
ifgram_inversion('unwrapIfgram.h5', 'coherence.h5', meta)
```

20.35.1.6 ifgram_inversion_patch()

```
def pysar.ifgram_inversion.ifgram_inversion_patch (
    ifgramFile,
    coherenceFile,
    meta,
    box = None )
```

Inputs:

```
ifgramFile    - string, interferograms hdf5 file
coherenceFile - string, coherence hdf5 file
box           - 4-tuple, left, upper, right, and lower pixel coordinate of area of interest
meta          - dict, including the following attributes:

                #Interferograms
                length/width - int, file size for each interferogram
                ifgram_list  - list of string, interferogram dataset name
                date12_list  - list of string, YYMMDD-YYMMDD
                ref_value    - np.array in size of (ifgram_num, 1)
                              reference pixel coordinate in row/column number
                ref_y/x     - int, reference pixel coordinate in row/column number

                #Time-series
                date8_list   - list of string in YYYYMMDD
                tbase_diff   - np.array in size of (date_num-1, 1), differential temporal baseline

                #Inversion
                weight_function - no, fim, var, coh
```

Outputs:

```
ts          - 3D np.array in size of (date_num, row_num, col_num)
temp_coh    - 2D np.array in size of (row_num, col_num)
tsStd       - 3D np.array in size of (date_num, row_num, col_num)
```

20.35.1.7 main()

```
def pysar.ifgram_inversion.main (
    argv )
```

20.35.1.8 network_inversion_sbass()

```
def pysar.ifgram_inversion.network_inversion_sbass (
    B,
    ifgram,
    tbase_diff,
    skipZeroPhase = True )
```

Network inversion based on Small BAseline Subsets (SBAS) algorithm (Berardino et al., 2002, IEEE-TGRS). For full rank design matrix, a.k.a., fully connected network, ordinary least square (OLS) inversion is applied; otherwise, Singular Value Decomposition (SVD).

Inputs:

B - 2D np.array in size of (ifgram_num, date_num-1)
design matrix B, which represents temporal baseline timeseries between master and slave date for each interferogram

ifgram - 2D np.array in size of (ifgram_num, pixel_num)
phase of all interferograms

tbase_diff - 2D np.array in size of (date_num-1, 1)
differential temporal baseline of time-series

skipZeroPhase - bool, skip ifgram with zero phase value

Output:

ts - 2D np.array in size of (date_num-1, pixel_num), phase time series

tempCoh - 1D np.array in size of (pixel_num), temporal coherence

20.35.1.9 network_inversion_wls()

```
def pysar.ifgram_inversion.network_inversion_wls (
    A,
    ifgram,
    weight,
    skipZeroPhase = True,
    Astd = None )
```

Network inversion based on Weighted Least Square (WLS) solution.

Inputs:

A - 2D np.array in size of (ifgram_num, date_num-1)
representing date configuration for each interferogram
(-1 for master, 1 for slave, 0 for others)

ifgram - np.array in size of (ifgram_num,) or (ifgram_num, 1)
phase of all interferograms

weight - np.array in size of (ifgram_num,) or (ifgram_num, 1)
weight of ifgram

skipZeroPhase - bool, skip ifgram with zero phase value

Astd - 2D np.array in size of (ifgram_num, date_num-1)
design matrix for STD calculation excluding the reference date

Output:

ts - 1D np.array in size of (date_num-1,), phase time series

tempCoh - float32, temporal coherence

tsStd - 1D np.array in size of (date_num-1,), decor noise std time series

20.35.1.10 phase_pdf_ds()

```
def pysar.ifgram_inversion.phase_pdf_ds (
    L,
    coherence = None,
    phiNum = 1000 )
```

Marginal PDF of interferometric phase for distributed scatterers (DS)
Eq. 66 (Tough et al., 1995) and Eq. 4.2.23 (Hanssen, 2001)

Inputs:

```
L          - int, number of independent looks
coherence  - 1D np.array for the range of coherence, with value < 1.0 for valid operation
phiNum     - int, number of phase sample for the numerical calculation
```

Output:

```
pdf        - 2D np.array, phase pdf in size of (phiNum, len(coherence))
coherence  - 1D np.array for the range of coherence
```

Example:

```
epsilon = 1e-4
coh = np.linspace(0., 1-epsilon, 1000)
pdf, coh = phase_pdf_ds(1, coherence=coh)
```

20.35.1.11 phase_variance_ds()

```
def pysar.ifgram_inversion.phase_variance_ds (
    L,
    coherence = None )
```

Interferometric phase variance for distributed scatterers (DS)
Eq. 2.1.2 (Box et al., 2015) and Eq. 4.2.27 (Hanssen, 2001)

Inputs:

```
L          - int, number of independent looks
coherence  - 1D np.array for the range of coherence, with value < 1.0 for valid operation
phiNum     - int, number of phase sample for the numerical calculation
```

Output:

```
var        - 1D np.array, phase variance in size of (len(coherence))
coherence  - 1D np.array for the range of coherence
```

Example:

```
epsilon = 1e-4
coh = np.linspace(0., 1-epsilon, 1000)
var, coh = phase_variance_ds(1, coherence=coh)
```

20.35.1.12 phase_variance_ps()

```
def pysar.ifgram_inversion.phase_variance_ps (
    L,
    coherence = None )
```

the Cramer-Rao bound (CRB) of phase variance

Given by Eq. 25 (Rodriguez and Martin, 1992) and Eq 4.2.32 (Hanssen, 2001)

Valid when coherence is close to 1.

20.35.1.13 read_template2inps()

```
def pysar.ifgram_inversion.read_template2inps (
    template_file,
    inps )
```

Read input template options into Namespace inps

20.35.1.14 round_to_1()

```
def pysar.ifgram_inversion.round_to_1 (
    x )
```

Return the most significant digit of input number

20.35.1.15 temporal_coherence()

```
def pysar.ifgram_inversion.temporal_coherence (
    A,
    ts,
    ifgram,
    weight = None,
    chunk_size = 500 )
```

Calculate temporal coherence based on Tizzani et al. (2007, RSE)

Inputs:

- A - 2D np.array in size of (ifgram_num, date_num-1)
representing date configuration for each interferogram
(-1 for master, 1 for slave, 0 for others)
- ts - 2D np.array in size of (date_num-1, pixel_num), phase time series
- ifgram - 2D np.array in size of (ifgram_num, pixel_num), observed interferometric phase
- weight - 2D np.array in size of (ifgram_num, pixel_num), weight of ifgram
- chunk_size - int, max number of pixels per loop during the calculation

Output:

- temp_coh - 1D np.array in size of (pixel_num), temporal coherence

20.35.1.16 write_timeseries_hdf5_file()

```
def pysar.ifgram_inversion.write_timeseries_hdf5_file (
    timeseries,
    date8_list,
    atr,
    timeseriesFile = None )
```

Write to timeseries HDF5 file

Inputs:

- timeseries - 3D np.array in size of (date_num, length, width)
cumulative time series phase
- date8_list - list of string in YYYYMMDD format
- atr - dict, attributes of time-series file, including two parts:
 - 1) attributes inherited from interferograms
 - 2) attributes of time-series inverted from network of interferograms:
 - P_BASELINE_TIMESERIES
 - P_BASELINE_TOP_TIMESERIES
 - P_BASELINE_BOTTOM_TIMESERIES
 - ref_date
- timeseriesFile - string, file name of output time-series file

Output:

- timeseriesFile - string, file name of output time-series file

20.35.2 Variable Documentation

20.35.2.1 EXAMPLE

EXAMPLE

20.35.2.2 REFERENCE

REFERENCE

20.35.2.3 TEMPLATE

TEMPLATE

20.36 pysar.ifgram_reconstruction Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.36.1 Function Documentation

20.36.1.1 main()

```
def pysar.ifgram_reconstruction.main (  
    argv )
```

20.36.1.2 usage()

```
def pysar.ifgram_reconstruction.usage ( )
```

20.37 pysar.ifgram_simulation Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.37.1 Function Documentation

20.37.1.1 cmdLineParse()

```
def pysar.ifgram_simulation.cmdLineParse ( )
```

20.37.1.2 main()

```
def pysar.ifgram_simulation.main (
    argv )
```

20.37.2 Variable Documentation

20.37.2.1 EXAMPLE

EXAMPLE

20.38 pysar.image_math Namespace Reference

Functions

- def [data_operation](#) (data, operator, operand)
- def [file_operation](#) (fname, operator, operand, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.38.1 Function Documentation

20.38.1.1 cmdLineParse()

```
def pysar.image_math.cmdLineParse ( )
```

20.38.1.2 data_operation()

```
def pysar.image_math.data_operation (
    data,
    operator,
    operand )
```

Mathmatic operation of 2D matrix

20.38.1.3 file_operation()

```
def pysar.image_math.file_operation (
    fname,
    operator,
    operand,
    fname_out = None )
```

Mathmathic operation of file

20.38.1.4 main()

```
def pysar.image_math.main (
    argv )
```

20.38.2 Variable Documentation

20.38.2.1 EXAMPLE

EXAMPLE

20.39 pysar.incidence_angle Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.39.1 Function Documentation

20.39.1.1 main()

```
def pysar.incidence_angle.main (
    argv )
```

20.39.1.2 usage()

```
def pysar.incidence_angle.usage ( )
```

20.40 pysar.info Namespace Reference

Functions

- def [print_attributes](#) (atr, sorting=True)
- def [print_hdf5_structure](#) (File)
By andrewcollette at <https://github.com/h5py/h5py/issues/406>.
- def [print_timseries_date_info](#) (dateList)
- def [usage](#) ()
- def [main](#) (argv)

20.40.1 Function Documentation

20.40.1.1 main()

```
def pysar.info.main (
    argv )
```

20.40.1.2 print_attributes()

```
def pysar.info.print_attributes (
    atr,
    sorting = True )
```

20.40.1.3 print_hdf5_structure()

```
def pysar.info.print_hdf5_structure (
    File )
```

By andrewcollette at <https://github.com/h5py/h5py/issues/406>.

20.40.1.4 print_timseries_date_info()

```
def pysar.info.print_timseries_date_info (
    dateList )
```

20.40.1.5 usage()

```
def pysar.info.usage ( )
```

20.41 pysar.insar_vs_gps Namespace Reference

Functions

- def [readGPSfile](#) (gpsFile, gps_source)
- def [nearest](#) (x, tbase, xstep)
- def [find_row_column](#) (Lon, Lat, lon, lat, lon_step, lat_step)
- def [usage](#) ()
- def [main](#) (argv)

20.41.1 Function Documentation

20.41.1.1 find_row_column()

```
def pysar.insar_vs_gps.find_row_column (
    Lon,
    Lat,
    lon,
    lat,
    lon_step,
    lat_step )
```

20.41.1.2 main()

```
def pysar.insar_vs_gps.main (
    argv )
```

20.41.1.3 nearest()

```
def pysar.insar_vs_gps.nearest (
    x,
    tbase,
    xstep )
```

20.41.1.4 readGPSfile()

```
def pysar.insar_vs_gps.readGPSfile (
    gpsFile,
    gps_source )
```

20.41.1.5 usage()

```
def pysar.insar_vs_gps.usage ( )
```

20.42 pysar.insarmaps_query Namespace Reference

Classes

- class [BasicHTTP](#)

Functions

- def [buildURL](#) (args)
- def [build_parser](#) ()
- def [main](#) ()

20.42.1 Function Documentation

20.42.1.1 build_parser()

```
def pysar.insarmaps_query.build_parser ( )
```

20.42.1.2 buildURL()

```
def pysar.insarmaps_query.buildURL (
    args )
```

20.42.1.3 main()

```
def pysar.insarmaps_query.main ( )
```


20.43 pysar.json_mbtiles2insarmaps Namespace Reference

Functions

- def [get_unavco_name](#) (json_path)
- def [upload_insarmaps_metadata](#) (fileName)
- def [upload_json](#) (folder_path)
- def [build_parser](#) ()
- def [main](#) ()

Variables

- [dbUsername](#)
- [dbPassword](#)
- [dbHost](#)

20.43.1 Function Documentation

20.43.1.1 build_parser()

```
def pysar.json_mbtiles2insarmaps.build_parser ( )
```

20.43.1.2 get_unavco_name()

```
def pysar.json_mbtiles2insarmaps.get_unavco_name (
    json_path )
```

20.43.1.3 main()

```
def pysar.json_mbtiles2insarmaps.main ( )
```

20.43.1.4 upload_insarmaps_metadata()

```
def pysar.json_mbtiles2insarmaps.upload_insarmaps_metadata (
    fileName )
```

20.43.1.5 upload_json()

```
def pysar.json_mbtiles2insarmaps.upload_json (
    folder_path )
```

20.43.2 Variable Documentation

20.43.2.1 dbHost

dbHost

20.43.2.2 dbPassword

dbPassword

20.43.2.3 dbUsername

dbUsername

20.44 pysar.l1 Namespace Reference

Functions

- def [l1mosek](#) (P, q)
- def [l1mosek2](#) (P, q)
- def [l1](#) (P, q)
- def [l1blas](#) (P, q)

Variables

- [__MOSEK](#)
- [task](#)
- [x](#)

20.44.1 Function Documentation

20.44.1.1 l1()

```
def pysar.l1.l1 (
    P,
    q )
```

Returns the solution u of the ℓ_1 - ℓ_1 approximation problem

```
(primal) minimize ||P*u - q||_1

(dual)  maximize   q'*w
        subject to P'*w = 0
                ||w||_infty <= 1.
```

20.44.1.2 l1blas()

```
def pysar.l1.l1blas (
    P,
    q )
```

Returns the solution u of the ℓ_1 -1 approximation problem

```
(primal) minimize ||P*u - q||_1

(dual)  maximize   q'*w
        subject to P'*w = 0
                ||w||_infty <= 1.
```

20.44.1.3 l1mosek()

```
def pysar.l1.l1mosek (
    P,
    q )
```

```
minimize   e'*v

subject to P*u - v <= q
          -P*u - v <= -q
```

20.44.1.4 l1mosek2()

```
def pysar.l1.l1mosek2 (
    P,
    q )
```

```
minimize   e'*s + e'*t

subject to P*u - q = s - t
          s, t >= 0
```

20.44.2 Variable Documentation**20.44.2.1 __MOSEK**

```
__MOSEK [private]
```

20.44.2.2 task

task

20.44.2.3 x

x

20.45 pysar.load_data Namespace Reference

Functions

- def [project_name2sensor](#) (projectName)
Sub Functions #####.
- def [auto_path_miami](#) (inps, template={})
- def [mode](#) (thelist)
- def [check_file_size](#) (fileList, mode_width=None, mode_length=None)
- def [check_existed_hdf5_file](#) (inFiles, hdf5File)
- def [load_multi_group_hdf5](#) (fileType, fileList, outfile='unwrapIfgram.h5', exDict=dict())
- def [load_geometry_hdf5](#) (fileType, fileList, outfile=None, exDict=dict())
- def [load_single_dataset_hdf5](#) (file_type, infile, outfile=None, exDict=dict())
- def [copy_file](#) (targetFile, destDir)
- def [load_file](#) (fileList, inps_dict=dict(), outfile=None, file_type=None)
- def [load_data_from_template](#) (inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [sensorList](#)
- [EXAMPLE](#)
Usage #####.
- [TEMPLATE](#)

20.45.1 Function Documentation

20.45.1.1 auto_path_miami()

```
def pysar.load_data.auto_path_miami (
    inps,
    template = {} )
```

Auto File Path Setting for Geodesy Lab - University of Miami

20.45.1.2 check_existed_hdf5_file()

```
def pysar.load_data.check_existed_hdf5_file (
    inFiles,
    hdf5File )
```

Check file list with existed hdf5 file
Return list of files that are not included in the existed readable hdf5 file.
If all included, return None.

20.45.1.3 check_file_size()

```
def pysar.load_data.check_file_size (
    fileList,
    mode_width = None,
    mode_length = None )
```

Update file list and drop those not in the same size with majority.

20.45.1.4 cmdLineParse()

```
def pysar.load_data.cmdLineParse ( )
```

20.45.1.5 copy_file()

```
def pysar.load_data.copy_file (
    targetFile,
    destDir )
```

Copy file and its .rsc/.par/.xml file to destination directory.

20.45.1.6 load_data_from_template()

```
def pysar.load_data.load_data_from_template (
    inps )
```

Load dataset for PySAR time series using input template

20.45.1.7 load_file()

```
def pysar.load_data.load_file (
    fileList,
    inps_dict = dict(),
    outfile = None,
    file_type = None )
```

Load input file(s) into one HDF5 file
It supports ROI_PAC files only for now.

Inputs:

```
fileList - string / list of string, path of files to load
inps_dict - dict, including the following attributes
    PROJECT_NAME : KujuAlosAT422F650 (extra attribute dictionary to add to output file)
    sensor       : (optional)
    timeseries_dir : directory of time series analysis, e.g. KujuAlosAT422F650/PYSAR
    insarProcessor: InSAR processor, roipac, isce, gamma, doris
outfile - string, output file name
file_type - string, group name for output HDF5 file, interferograms, coherence, dem, etc.
```

Output:

```
outfile - string, output file name
```

Example:

```
unwrapIfgram.h5 = load_file('filt*.unw', inps_dict=vars(inps))
```

20.45.1.8 load_geometry_hdf5()

```
def pysar.load_data.load_geometry_hdf5 (
    fileType,
    fileList,
    outfile = None,
    exDict = dict() )
```

Load multiple geometry files into hdf5 file: geometryGeo.h5 or geometryRadar.h5.

File structure:

```
/geometry.attrs
/geometry/latitude      #for geometryRadar.h5 only, from ISCE/Doris lookup table
/geometry/longitude     #for geometryRadar.h5 only, from ISCE/Doris lookup table
/geometry/rangeCoord    #for geometryGeo.h5 only, from ROI_PAC/Gamma lookup table
/geometry/azimuthCoord  #for geometryGeo.h5 only, from ROI_PAC/Gamma lookup table
/geometry/height
/geometry/incidenceAngle
/geometry/headingAngle
/geometry/slantRangeDistance
/geometry/shadowMask
/geometry/waterMask
```

20.45.1.9 load_multi_group_hdf5()

```
def pysar.load_data.load_multi_group_hdf5 (
    fileType,
    fileList,
    outfile = 'unwrapIfgram.h5',
    exDict = dict() )
```

Load multiple ROI_PAC files into HDF5 file (Multi-group, one dataset and one attribute dict per group).

Inputs:

```
fileType : string, i.e. interferograms, coherence, snaphu_connect_component, etc.
fileList : list of path, ROI_PAC .unw/.cor/.int/.byt file
outfile  : string, file name/path of the multi-group hdf5 PySAR file
exDict  : dict, extra attribute dictionary
```

Outputs:

```
outfile : output hdf5 file name
fileList : list of string, files newly added
```

20.45.1.10 load_single_dataset_hdf5()

```
def pysar.load_data.load_single_dataset_hdf5 (
    file_type,
    infile,
    outfile = None,
    exDict = dict() )
```

Convert ROI_PAC .dem / .hgt file to hdf5 file

Based on load_dem.py written by Emre Havazli

Inputs:

```
file_type : string, group name of hdf5 file, i.e. dem, mask
infile    : string, input ROI_PAC file name
outfile   : string, output hdf5 file name
exDict    : dict, extra attributes to output file
```

Output:

```
outfile    : string, output hdf5 file name
```

20.45.1.11 main()

```
def pysar.load_data.main (
    argv )
```

Main Function #####.

20.45.1.12 mode()

```
def pysar.load_data.mode (
    thelist )
```

Find Mode (most common) item in the list

20.45.1.13 project_name2sensor()

```
def pysar.load_data.project_name2sensor (
    projectName )
```

Sub Functions #####.

20.45.2 Variable Documentation

20.45.2.1 EXAMPLE

EXAMPLE

Usage #####.

20.45.2.2 sensorList

sensorList

20.45.2.3 TEMPLATE

TEMPLATE

20.46 pysar.load_dem Namespace Reference

Variables

- [demFile](#)
- [ext](#)
- [amp](#)
- [dem](#)
- [demRsc](#)
- [outName](#)
- [h5](#)
- [group](#)
- [dset](#)
- [data](#)
- [compression](#)

20.46.1 Variable Documentation

20.46.1.1 amp

amp

20.46.1.2 compression

compression

20.46.1.3 data

data

20.46.1.4 dem

dem

20.46.1.5 demFile

demFile

20.46.1.6 demRsc

demRsc

20.46.1.7 dset

dset

20.46.1.8 ext

ext

20.46.1.9 group

group

20.46.1.10 h5

h5

20.46.1.11 outName

outName

20.47 pysar.lod Namespace Reference

Functions

- def [correct_lod_file](#) (File, rangeDistFile=None, outFile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [REFERENCE](#)
- [EXAMPLE](#)

20.47.1 Function Documentation

20.47.1.1 cmdLineParse()

```
def pysar.lod.cmdLineParse ( )
```

20.47.1.2 correct_lod_file()

```
def pysar.lod.correct_lod_file (
    File,
    rangeDistFile = None,
    outFile = None )
```

20.47.1.3 main()

```
def pysar.lod.main (
    argv )
```

20.47.2 Variable Documentation

20.47.2.1 EXAMPLE

EXAMPLE

20.47.2.2 REFERENCE

REFERENCE

20.48 pysar.look_angle Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.48.1 Function Documentation

20.48.1.1 main()

```
def pysar.look_angle.main (
    argv )
```

20.48.1.2 usage()

```
def pysar.look_angle.usage ( )
```

20.49 pysar.los2enu Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.49.1 Function Documentation

20.49.1.1 main()

```
def pysar.los2enu.main (
    argv )
```

20.49.1.2 usage()

```
def pysar.los2enu.usage ( )
```

20.50 pysar.mask Namespace Reference

Functions

- def [mask_matrix](#) (data_mat, mask_mat, fill_value=None)
- def [update_mask](#) (mask, inps_dict, print_msg=True)
- def [mask_file](#) (File, maskFile, outFile=None, inps_dict=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.50.1 Function Documentation

20.50.1.1 cmdLineParse()

```
def pysar.mask.cmdLineParse ( )
```

20.50.1.2 main()

```
def pysar.mask.main (
    argv )
```

20.50.1.3 mask_file()

```
def pysar.mask.mask_file (
    File,
    maskFile,
    outFile = None,
    inps_dict = None )
```

Mask input File with maskFile

Inputs:

```
File/maskFile - string,
inps_dict - dictionary including the following options:
    subset_x/y - list of 2 ints, subset in x/y direction
    thr - float, threshold/minValue to generate mask
```

Output:

```
outFile - string
```

20.50.1.4 mask_matrix()

```
def pysar.mask.mask_matrix (
    data_mat,
    mask_mat,
    fill_value = None )
```

mask a 2D matrix data with mask

20.50.1.5 update_mask()

```
def pysar.mask.update_mask (
    mask,
    inps_dict,
    print_msg = True )
```

Update mask matrix from input options: subset_x/y and threshold

20.50.2 Variable Documentation

20.50.2.1 EXAMPLE

EXAMPLE

20.51 pysar.match Namespace Reference

Functions

- def [corners](#) (atr)
- def [nearest](#) (x, X)
- def [manual_offset_estimate](#) (matrix1, matrix2)
- def [match_two_files](#) (File1, File2, outName=None, manual_match=False, disp_fig=False)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.51.1 Function Documentation

20.51.1.1 cmdLineParse()

```
def pysar.match.cmdLineParse ( )
```

20.51.1.2 corners()

```
def pysar.match.corners (
    atr )
```

Get corners coordinate.

20.51.1.3 main()

```
def pysar.match.main (
    argv )
```

20.51.1.4 manual_offset_estimate()

```
def pysar.match.manual_offset_estimate (
    matrix1,
    matrix2 )
```

Manually estimate offset between two data matrix.
By manually selecting a line from each of them, and estimate the difference.
It usually used when 2 input data matrix have no area in common.

20.51.1.5 match_two_files()

```
def pysar.match.match_two_files (
    File1,
    File2,
    outName = None,
    manual_match = False,
    disp_fig = False )
```

Match two geocoded files by estimating their offset.
Better for two files with common area overlaping.

20.51.1.6 nearest()

```
def pysar.match.nearest (
    x,
    X )
```

find nearest neighbour

20.51.2 Variable Documentation

20.51.2.1 EXAMPLE

EXAMPLE

20.52 pysar.modify_network Namespace Reference

Functions

- def [nearest_neighbor](#) (x, y, x_array, y_array)
Sub Function #####.
- def [reset_pairs](#) (File)
- def [manual_select_pairs_to_remove](#) (File)
- def [modify_file_date12_list](#) (File, date12_to_rmv, mark_attribute=False, outFile=None)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [EXAMPLE](#)
Usage #####.
- [TEMPLATE](#)

20.52.1 Function Documentation

20.52.1.1 cmdLineParse()

```
def pysar.modify_network.cmdLineParse ( )
```

20.52.1.2 main()

```
def pysar.modify_network.main (
    argv )
```

Main Function #####.

20.52.1.3 manual_select_pairs_to_remove()

```
def pysar.modify_network.manual_select_pairs_to_remove (
    File )
```

Manually select interferograms to remove

20.52.1.4 modify_file_date12_list()

```
def pysar.modify_network.modify_file_date12_list (
    File,
    date12_to_rmv,
    mark_attribute = False,
    outFile = None )
```

Update multiple group hdf5 file using date12 to remove

Inputs:

```
File          - multi_group HDF5 file, i.e. unwrapIfgram.h5, coherence.h5
date12_to_rmv - list of string indicating interferograms in YYMMDD-YYMMDD format
mark_attribute- bool, if True, change 'drop_ifgram' attribute only; otherwise, write
                result1 to a new file
outFile        - string, output file name
```

Output:

```
outFile        - string, output file name, if mark_attribute=True, outFile = File
```

20.52.1.5 nearest_neighbor()

```
def pysar.modify_network.nearest_neighbor (
    x,
    y,
    x_array,
    y_array )
```

Sub Function #####.

find nearest neighbour

Input:

```
x/y          : float
x/y_array    : numpy.array, temporal/perpendicular spatial baseline
```

Output:

```
idx : int, index of min distance - nearest neighbour
```


20.52.1.6 read_template2inps()

```
def pysar.modify_network.read_template2inps (
    template_file,
    inps = None )
```

Read input template options into Namespace inps

20.52.1.7 reset_pairs()

```
def pysar.modify_network.reset_pairs (
    File )
```

Reset/restore all pairs within the input file by set all drop_ifgram=no

20.52.2 Variable Documentation

20.52.2.1 EXAMPLE

EXAMPLE

Usage #####.

20.52.2.2 TEMPLATE

TEMPLATE

20.53 pysar.multi_transect Namespace Reference

Functions

- def [usage](#) ()
- def [dms2d](#) (Coord)
- def [gps_to_LOS](#) (Ve, Vn, theta, heading)
- def [check_st_in_box](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [check_st_in_box2](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [line](#) (x0, y0, x1, y1)
- def [dist_point_from_line](#) (m, c, x, y, dx, dy)
- def [get_intersect](#) (m, c, x, y)
- def [readGPSfile](#) (gpsFile, gps_source)
- def [redGPSfile](#) (gpsFile)
- def [redGPSfile_cmm4](#) (gpsFile)
- def [nearest](#) (x, tbase, xstep)
- def [find_row_column](#) (Lon, Lat, lon, lat, lon_step, lat_step)
- def [get_lat_lon](#) (h5file)
- def [nanmean](#) (data, args)
- def [nanstd](#) (data, args)
- def [get_transect](#) (z, x0, y0, x1, y1)
- def [get_start_end_point](#) (Xf0, Yf0, Xf1, Yf1, L, dx, dy)
- def [point_with_distance_from_line](#) (Xf0, Yf0, Xf1, Yf1, L)
- def [point_on_line_with_distance_from_beginning](#) (Xf0, Yf0, Xf1, Yf1, L)
- def [read_fault_coords](#) (Fault_coord_file, Dp)
- def [main](#) (argv)
- def [onclick](#) (event)

Variables

- lat
- lon
- lat_step
- lon_step
- lat_all
- lon_all
- Fault_lon
- Fault_lat
- Num_profiles
- FaultCoords
- Lat0
- Lon0
- Lat1
- Lon1
- Length
- Width
- Yf0
- Xf0
- Yf1
- Xf1
- y0
- x0
- y1
- x1
- fig
- ax
- xc
- yc
- cid
- length

*try: mf=float(Yf1-Yf0)/float((Xf1-Xf0)) # slope of the fault line cf=float(Yf0-mf*Xf0) # intercept of the fault line df0=dist_↔
_point_from_line(mf,cf,x0,y0,1,1) #distance of the profile start point from the Fault line df1=dist_point_from_↔
line(mf,cf,x1,y1,1,1) #distance of the profile end point from the Fault line*

- x
- y
- zi
- lat_transect
- lon_transect
- dx
- dy
- DX
- DY
- D
- mf
- cf
- df0_km
- transect
- XX0
- XX1
- YY0
- YY1
- m
- c

- [m1](#)
- [dp](#)
- [X0](#)
- [Y0](#)
- [X1](#)
- [Y1](#)
- [transect_lat](#)
- [transect_lon](#)
- [m_prof_edge](#)
- [c_prof_edge](#)
- [gpsFile](#)
- [insarData](#)
- [fileName](#)
- [fileExtension](#)
- [Stations](#)
- [Lat](#)
- [Lon](#)
- [Ve](#)
- [Se](#)
- [Vn](#)
- [Sn](#)
- [idxRef](#)
- [IDYref](#)
- [IDXref](#)
- [stationsList](#)
- [h5file_theta](#)
- [dset](#)
- [theta](#)
- [heading](#)
- [unitVec](#)
- [gpsLOS_ref](#)
- [GPS](#)
- [GPS_station](#)
- [GPSx](#)
- [GPSy](#)
- [GPS_lat](#)
- [GPS_lon](#)
- [idx](#)
- [IDY](#)
- [IDX](#)
- [gpsLOS](#)
- [NoInSAR](#)
- [DistGPS](#)
- [GPS_in_bound](#)
- [GPS_in_bound_st](#)
- [GPSxx](#)
- [GPSyy](#)
- [gx](#)
- [gy](#)
- [check_result](#)
- [check_result2](#)
- [dg](#)
- [axes](#)
- [nrows](#)
- [ms](#)

```

ax.fill_between(D/1000.0, (avglnSAR-stdlnSAR)*1000, (avglnSAR+stdlnSAR)*1000, where=(avglnSAR+stdlnSAR)*1000>=(avglnSAR-stdlnSAR)*1000, alpha=1, facecolor='Red')

```

- [avglnSAR](#)
- [axis](#)
- [stdlnSAR](#)
- [fig2](#)
- [axes2](#)
- [FaultLine](#)
- [figName](#)

Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minorLocator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)

- [mfc](#)
- [linewidth](#)
- [matFile](#)
- [dataset](#)
- [color](#)

```

ax.plot(D/1000.0, avglnSAR*1000, 'r-')

```

- [alpha](#)
- [fontsize](#)
- [lbound](#)

lower and higher bounds for displaying the profile

- [hbound](#)
- [ylim](#)
- [xlim](#)

20.53.1 Function Documentation

20.53.1.1 `check_st_in_box()`

```

def pysar.multi_transect.check_st_in_box (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )

```

20.53.1.2 `check_st_in_box2()`

```

def pysar.multi_transect.check_st_in_box2 (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )

```

20.53.1.3 dist_point_from_line()

```
def pysar.multi_transect.dist_point_from_line (
    m,
    c,
    x,
    y,
    dx,
    dy )
```

20.53.1.4 dms2d()

```
def pysar.multi_transect.dms2d (
    Coord )
```

20.53.1.5 find_row_column()

```
def pysar.multi_transect.find_row_column (
    Lon,
    Lat,
    lon,
    lat,
    lon_step,
    lat_step )
```

20.53.1.6 get_intersect()

```
def pysar.multi_transect.get_intersect (
    m,
    c,
    x,
    y )
```

20.53.1.7 get_lat_lon()

```
def pysar.multi_transect.get_lat_lon (
    h5file )
```

20.53.1.8 get_start_end_point()

```
def pysar.multi_transect.get_start_end_point (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L,
    dx,
    dy )
```

20.53.1.9 get_transect()

```
def pysar.multi_transect.get_transect (
    z,
    x0,
    y0,
    x1,
    y1 )
```

20.53.1.10 gps_to_LOS()

```
def pysar.multi_transect.gps_to_LOS (
    Ve,
    Vn,
    theta,
    heading )
```

20.53.1.11 line()

```
def pysar.multi_transect.line (
    x0,
    y0,
    x1,
    y1 )
```

20.53.1.12 main()

```
def pysar.multi_transect.main (
    argv )
```

20.53.1.13 nanmean()

```
def pysar.multi_transect.nanmean (
    data,
    args )
```

20.53.1.14 nanstd()

```
def pysar.multi_transect.nanstd (
    data,
    args )
```

20.53.1.15 nearest()

```
def pysar.multi_transect.nearest (
    x,
    tbase,
    xstep )
```

20.53.1.16 onclick()

```
def pysar.multi_transect.onclick (
    event )
```

20.53.1.17 point_on_line_with_distance_from_beginning()

```
def pysar.multi_transect.point_on_line_with_distance_from_beginning (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L )
```

20.53.1.18 point_with_distance_from_line()

```
def pysar.multi_transect.point_with_distance_from_line (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L )
```

20.53.1.19 read_fault_coords()

```
def pysar.multi_transect.read_fault_coords (
    Fault_coord_file,
    Dp )
```

20.53.1.20 readGPSfile()

```
def pysar.multi_transect.readGPSfile (
    gpsFile,
    gps_source )
```

20.53.1.21 redGPSfile()

```
def pysar.multi_transect.redGPSfile (
    gpsFile )
```

20.53.1.22 redGPSfile_cmm4()

```
def pysar.multi_transect.redGPSfile_cmm4 (
    gpsFile )
```

20.53.1.23 usage()

```
def pysar.multi_transect.usage ( )
```

20.53.2 Variable Documentation**20.53.2.1 alpha**

alpha

20.53.2.2 avgInSAR

avgInSAR

20.53.2.3 ax

ax

20.53.2.4 axes

axes

20.53.2.5 axes2

axes2

20.53.2.6 axis

axis

20.53.2.7 c

c

20.53.2.8 c_prof_edge

c_prof_edge

20.53.2.9 cf

cf

20.53.2.10 check_result

check_result

20.53.2.11 check_result2

check_result2

20.53.2.12 cid

cid

20.53.2.13 color

color

```
ax.plot(D/1000.0, avglnSAR*1000, 'r-')
```

To plot the Fault location on the profile try:

20.53.2.14 D

D

20.53.2.15 dataset

dataset

20.53.2.16 df0_km

df0_km

20.53.2.17 dg

dg

20.53.2.18 DistGPS

DistGPS

20.53.2.19 dp

dp

20.53.2.20 dset

dset

20.53.2.21 dx

dx

20.53.2.22 DX

DX

20.53.2.23 dy

dy

20.53.2.24 DY

DY

20.53.2.25 Fault_lat

Fault_lat

20.53.2.26 Fault_lon

Fault_lon

20.53.2.27 FaultCoords

FaultCoords

20.53.2.28 FaultLine

FaultLine

20.53.2.29 fig

fig

20.53.2.30 fig2

fig2

20.53.2.31 figName

figName

Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minorLocator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)

20.53.2.32 fileExtension

fileExtension

20.53.2.33 fileName

fileName

20.53.2.34 fontsize

fontsize

20.53.2.35 GPS

GPS

20.53.2.36 GPS_in_bound

GPS_in_bound

20.53.2.37 GPS_in_bound_st

GPS_in_bound_st

20.53.2.38 GPS_lat

GPS_lat

20.53.2.39 GPS_lon

GPS_lon

20.53.2.40 GPS_station

GPS_station

20.53.2.41 gpsFile

gpsFile

20.53.2.42 gpsLOS

gpsLOS

20.53.2.43 gpsLOS_ref

gpsLOS_ref

20.53.2.44 GPSx

GPSx

20.53.2.45 GPSxx

GPSxx

20.53.2.46 GPSy

GPSy

20.53.2.47 GPSyy

GPSyy

20.53.2.48 gx

gx

20.53.2.49 gy

gy

20.53.2.50 h5file_theta

h5file_theta

20.53.2.51 hbound

hbound

20.53.2.52 heading

heading

20.53.2.53 idx

idx

20.53.2.54 IDX

IDX

20.53.2.55 idxRef

idxRef

20.53.2.56 IDXref

IDXref

20.53.2.57 IDY

IDY

20.53.2.58 IDYref

IDYref

20.53.2.59 insarData

insarData

20.53.2.60 lat

lat

20.53.2.61 Lat

Lat

20.53.2.62 Lat0

Lat0

20.53.2.63 Lat1

Lat1

20.53.2.64 lat_all

lat_all

20.53.2.65 lat_step

lat_step

20.53.2.66 lat_transect

lat_transect

20.53.2.67 lbound

lbound

lower and higher bounds for displaying the profile

20.53.2.68 Length

Length

20.53.2.69 length

length

```
try: mf=float(Yf1-Yf0)/float((Xf1-Xf0)) # slope of the fault line cf=float(Yf0-mf*Xf0) # intercept of the fault line
df0=dist_point_from_line(mf,cf,x0,y0,1,1) #distance of the profile start point from the Fault line df1=dist_point_↵
from_line(mf,cf,x1,y1,1,1) #distance of the profile end point from the Fault line
```

20.53.2.70 linewidth

linewidth

20.53.2.71 lon

lon

20.53.2.72 Lon

Lon

20.53.2.73 Lon0

Lon0

20.53.2.74 Lon1

Lon1

20.53.2.75 lon_all

lon_all

20.53.2.76 lon_step

lon_step

20.53.2.77 lon_transect

lon_transect

20.53.2.78 m

m

20.53.2.79 m1

m1

20.53.2.80 m_prof_edge

m_prof_edge

20.53.2.81 matFile

matFile

20.53.2.82 mf

mf

20.53.2.83 mfc

mfc

20.53.2.84 ms

ms

ax.fill_between(D/1000.0, (avglnSAR-stdlnSAR)*1000, (avglnSAR+stdlnSAR)*1000,where=(avglnSAR+stdlnSAR)*1000>=(avglnSAR-stdlnSAR)*1000,alpha=1, facecolor='Red')

20.53.2.85 NoInSAR

NoInSAR

20.53.2.86 nrows

nrows

20.53.2.87 Num_profiles

Num_profiles

20.53.2.88 Se

Se

20.53.2.89 Sn

Sn

20.53.2.90 Stations

Stations

20.53.2.91 stationsList

stationsList

20.53.2.92 stdInSAR

stdInSAR

20.53.2.93 theta

theta

20.53.2.94 transect

transect

20.53.2.95 transect_lat

transect_lat

20.53.2.96 transect_lon

transect_lon

20.53.2.97 unitVec

unitVec

20.53.2.98 Ve

Ve

20.53.2.99 Vn

Vn

20.53.2.100 Width

Width

20.53.2.101 x

x

20.53.2.102 x0

x0

20.53.2.103 X0

X0

20.53.2.104 x1

x1

20.53.2.105 X1

X1

20.53.2.106 xc

xc

20.53.2.107 Xf0

Xf0

20.53.2.108 Xf1

Xf1

20.53.2.109 xlim

xlim

20.53.2.110 XX0

XX0

20.53.2.111 XX1

XX1

20.53.2.112 y

y

20.53.2.113 y0

y0

20.53.2.114 Y0

Y0

20.53.2.115 y1

y1

20.53.2.116 Y1

Y1

20.53.2.117 yc

yc

20.53.2.118 Yf0

Yf0

20.53.2.119 Yf1

Yf1

20.53.2.120 ylim

ylim

20.53.2.121 YY0

YY0

20.53.2.122 YY1

YY1

20.53.2.123 zi

zi

20.54 pysar.multilook Namespace Reference**Functions**

- def [multilook_matrix](#) (matrix, lks_y, lks_x)
 Sub Functions #####
- def [multilook_attribute](#) (atr_dict, lks_y, lks_x, print_msg=True)
- def [multilook_file](#) (infile, lks_y, lks_x, outfile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.54.1 Function Documentation

20.54.1.1 cmdLineParse()

```
def pysar.multilook.cmdLineParse ( )
```

20.54.1.2 main()

```
def pysar.multilook.main (
    argv )
```

20.54.1.3 multilook_attribute()

```
def pysar.multilook.multilook_attribute (
    atr_dict,
    lks_y,
    lks_x,
    print_msg = True )
```

20.54.1.4 multilook_file()

```
def pysar.multilook.multilook_file (
    infile,
    lks_y,
    lks_x,
    outfile = None )
```

20.54.1.5 multilook_matrix()

```
def pysar.multilook.multilook_matrix (
    matrix,
    lks_y,
    lks_x )
```

Sub Functions #####.

20.54.2 Variable Documentation

20.54.2.1 EXAMPLE

EXAMPLE

20.55 pysar.perp_baseline Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.55.1 Function Documentation

20.55.1.1 main()

```
def pysar.perp_baseline.main (
    argv )
```

20.55.1.2 usage()

```
def pysar.perp_baseline.usage ( )
```

20.56 pysar.plot_network Namespace Reference

Functions

- def [read_template2inps](#) (template_file, inps=None)
Sub Function #####.
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [BL_LIST](#)
Sub Function #####.
- [DATE12_LIST](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

20.56.1 Function Documentation

20.56.1.1 cmdLineParse()

```
def pysar.plot_network.cmdLineParse ( )
```

20.56.1.2 main()

```
def pysar.plot_network.main (
    argv )
```

Main Function #####.

20.56.1.3 read_template2inps()

```
def pysar.plot_network.read_template2inps (
    template_file,
    inps = None )
```

Sub Function #####.

Read input template options into Namespace inps

20.56.2 Variable Documentation

20.56.2.1 BL_LIST

BL_LIST

Sub Function #####.

20.56.2.2 DATE12_LIST

DATE12_LIST

20.56.2.3 EXAMPLE

EXAMPLE

20.56.2.4 TEMPLATE

TEMPLATE

20.57 pysar.prep4timeseries Namespace Reference

Functions

- def [createParser](#) ()
- def [cmdLineParse](#) (iargs=None)
- def [extractIsceMetadata](#) (xmlFile)
- def [read_baseline](#) (baselineFile)
- def [baselineTimeseries](#) (baselineDir)
- def [read_rsc](#) (rscFile)
- def [write_rsc](#) (rscDict, rscFile)
- def [attribute_isce2roipac](#) (metaDict, dates=[], baselineDict={})
- def [prepare_stack](#) (inputDir, filePattern, metaDictIn, baselineDict)
- def [prepare_geometry](#) (geometryDir, exDict=None)
- def [read_template](#) (File, delimiter='=')
- *from _read_file.py Need to be removed once we can import _readfile.py*
- def [check_variable_name](#) (path)
- def [main](#) (iargs=None)

Variables

- [GDAL2NUMPY_DATATYPE](#)
- [EXAMPLE](#)

20.57.1 Function Documentation

20.57.1.1 attribute_isce2roipac()

```
def pysar.prep4timeseries.attribute_isce2roipac (
    metaDict,
    dates = [],
    baselineDict = {} )
```

20.57.1.2 baselineTimeseries()

```
def pysar.prep4timeseries.baselineTimeseries (
    baselineDir )
```

20.57.1.3 check_variable_name()

```
def pysar.prep4timeseries.check_variable_name (
    path )
```

20.57.1.4 cmdLineParse()

```
def pysar.prep4timeseries.cmdLineParse (
    iargs = None )
```

20.57.1.5 createParser()

```
def pysar.prep4timeseries.createParser ( )
```

Command line parser.

20.57.1.6 extractIsceMetadata()

```
def pysar.prep4timeseries.extractIsceMetadata (
    xmlFile )
```

20.57.1.7 main()

```
def pysar.prep4timeseries.main (
    iargs = None )
```

20.57.1.8 prepare_geometry()

```
def pysar.prep4timeseries.prepare_geometry (
    geometryDir,
    exDict = None )
```

Prepare Geometry files for PySAR: DEM in radar coord, and lookup table

Input:

geometryDir - string, path to the directory of merged/geo
exDict - dictionary, interferogram attributes to be updated with geometry file

Output:

geometryRadar.h5 - HDF5 file with group - geometry and sub-datasets:
latitude
longitude
height
incidenceAngle
headingAngle

20.57.1.9 prepare_stack()

```
def pysar.prep4timeseries.prepare_stack (
    inputDir,
    filePattern,
    metaDictIn,
    baselineDict )
```

20.57.1.10 read_baseline()

```
def pysar.prep4timeseries.read_baseline (
    baselineFile )
```

20.57.1.11 read_rsc()

```
def pysar.prep4timeseries.read_rsc (
    rscFile )
```

20.57.1.12 read_template()

```
def pysar.prep4timeseries.read_template (
    File,
    delimiter = '=' )
```

from `_read_file.py` Need to be removed once we can import `_readfile.py`

Reads the template file into a python dictionary structure.

Input : string, full path to the template file

Output: dictionary, pysar template content

Example:

```
tmpl = read_template(KyushuT424F610_640AlosA.template)
tmpl = read_template(R1_54014_ST5_L0_F898.000.pi, ':')
```

20.57.1.13 write_rsc()

```
def pysar.prep4timeseries.write_rsc (
    rscDict,
    rscFile )
```

20.57.2 Variable Documentation

20.57.2.1 EXAMPLE

EXAMPLE

20.57.2.2 GDAL2NUMPY_DATATYPE

GDAL2NUMPY_DATATYPE

20.58 pysar.prep_gamma Namespace Reference

Functions

- def [get_perp_baseline](#) (m_par_file, s_par_file, off_file, atr_dict={})
Sub Functions #####
- def [get_lalo_ref](#) (m_par_file, atr_dict={})
- def [extract_attribute_interferogram](#) (fname)
- def [extract_attribute_lookup_table](#) (fname)
- def [extract_attribute_dem_geo](#) (fname)
- def [extract_attribute_dem_radar](#) (fname)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

20.58.1 Function Documentation

20.58.1.1 cmdLineParse()

```
def pysar.prep_gamma.cmdLineParse ( )
```

20.58.1.2 extract_attribute_dem_geo()

```
def pysar.prep_gamma.extract_attribute_dem_geo (
    fname )
```

Read/extract attribute for .dem file from Gamma to ROI_PAC
 For example, it read input file, sim_150911-150922.utm.dem,
 find its associated par file, sim_150911-150922.utm.dem.par, read it, and
 convert to ROI_PAC style and write it to an rsc file, sim_150911-150922.utm.dem.rsc

20.58.1.3 extract_attribute_dem_radar()

```
def pysar.prep_gamma.extract_attribute_dem_radar (
    fname )
```

Read/extract attribute for .hgt_sim file from Gamma to ROI_PAC
 Input:
 sim_150911-150922.hgt_sim
 sim_150911-150922.rdc.dem
 Search for:
 sim_150911-150922.diff_par
 Output:
 sim_150911-150922.hgt_sim.rsc
 sim_150911-150922.rdc.dem.rsc

20.58.1.4 extract_attribute_interferogram()

```
def pysar.prep_gamma.extract_attribute_interferogram (
    fname )
```

Read/extract attributes for PySAR from Gamma .unw, .cor and .int file

Inputs:

fname : str, Gamma interferogram filename or path, i.e. /PopoSLT143TsxD/diff_filt_HDR_130118-130129_4rlks.amp

Output:

atr : dict, Attributes dictionary

20.58.1.5 extract_attribute_lookup_table()

```
def pysar.prep_gamma.extract_attribute_lookup_table (
    fname )
```

Read/extract attribute for .UTM_TO_RDC file from Gamma to ROI_PAC

For example, it read input file, sim_150911-150922.UTM_TO_RDC,

find its associated par file, sim_150911-150922.utm.dem.par, read it, and

convert to ROI_PAC style and write it to an rsc file, sim_150911-150922.UTM_TO_RDC.rsc

20.58.1.6 get_lalo_ref()

```
def pysar.prep_gamma.get_lalo_ref (
    m_par_file,
    atr_dict = {} )
```

Extract LAT/LON_REF1/2/3/4 from corner file, e.g. 130118_4rlks.amp.corner.

If it's not existed, call Gamma script - SLC_corners - to generate it from SLC par file, e.g. 130118_4rlks.amp.par

Parameters: m_par_file : str, path, master date parameter file, i.e. 130118_4rlks.amp.par

atr_dict : dict, optional, attributes dictionary

Returns: lalo_ref

20.58.1.7 get_perp_baseline()

```
def pysar.prep_gamma.get_perp_baseline (
    m_par_file,
    s_par_file,
    off_file,
    atr_dict = {} )
```

Sub Functions #####

Get perpendicular baseline info from master/slave par file and off file.

Parameters: m_par_file : str, path, master parameter file, i.e. 130118_4rlks.amp.par

s_par_file : str, path, slave parameter file, i.e. 130129_4rlks.amp.oar

off_file : str, path, interferogram off file, i.e. 130118-130129_4rlks.off

atr_dict : dict, optional, attributes dictionary

Returns: bperp : str, perpendicular baseline for pixel at [0,0]

20.58.1.8 main()

```
def pysar.prep_gamma.main (
    argv )
```

20.58.2 Variable Documentation

20.58.2.1 DESCRIPTION

DESCRIPTION

20.58.2.2 EXAMPLE

EXAMPLE

20.59 pysar.prep_giant_ifg_list Namespace Reference

Functions

- def [get_mission_name](#) (meta_dict)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.59.1 Function Documentation

20.59.1.1 cmdLineParse()

```
def pysar.prep_giant_ifg_list.cmdLineParse ( )
```

20.59.1.2 get_mission_name()

```
def pysar.prep_giant_ifg_list.get_mission_name (
    meta_dict )
```

Get mission name in UNAVCO InSAR Archive format from attribute mission/PLATFORM
Input: meta_dict : dict, attributes
Output: mission : string, mission name in standard UNAVCO format.

20.59.1.3 main()

```
def pysar.prep_giant_ifg_list.main (
    argv )
```

20.59.2 Variable Documentation

20.59.2.1 EXAMPLE

EXAMPLE

20.60 pysar.prep_isce Namespace Reference

Functions

- def [createParser](#) ()
- def [cmdLineParse](#) (iargs=None)
- def [extractIsceMetadata](#) (xmlFile)
- def [write_rsc](#) (isceFile, dates, metadata, baselineDict)
- def [prepare_stack](#) (inputDir, filePattern, metadata, baselineDict)
- def [read_baseline](#) (baselineFile)
- def [baselineTimeseries](#) (baselineDir)
- def [prepare_geometry](#) (geometryDir)
- def [main](#) (iargs=None)

Variables

- [GDAL2NUMPY_DATATYPE](#)

20.60.1 Function Documentation

20.60.1.1 baselineTimeseries()

```
def pysar.prep_isce.baselineTimeseries (
    baselineDir )
```

20.60.1.2 cmdLineParse()

```
def pysar.prep_isce.cmdLineParse (
    iargs = None )
```


20.60.1.3 createParser()

```
def pysar.prep_isce.createParser ( )
```

Command line parser.

20.60.1.4 extractIsceMetadata()

```
def pysar.prep_isce.extractIsceMetadata (
    xmlFile )
```

20.60.1.5 main()

```
def pysar.prep_isce.main (
    iargs = None )
```

20.60.1.6 prepare_geometry()

```
def pysar.prep_isce.prepare_geometry (
    geometryDir )
```

20.60.1.7 prepare_stack()

```
def pysar.prep_isce.prepare_stack (
    inputDir,
    filePattern,
    metadata,
    baselineDict )
```

20.60.1.8 read_baseline()

```
def pysar.prep_isce.read_baseline (
    baselineFile )
```

20.60.1.9 write_rsc()

```
def pysar.prep_isce.write_rsc (
    isceFile,
    dates,
    metadata,
    baselineDict )
```

20.60.2 Variable Documentation

20.60.2.1 GDAL2NUMPY_DATATYPE

GDAL2NUMPY_DATATYPE

20.61 pysar.prep_roipac Namespace Reference

Functions

- def [extract_attribute](#) (fname)
 Sub Functions #####
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

20.61.1 Function Documentation

20.61.1.1 cmdLineParse()

```
def pysar.prep_roipac.cmdLineParse ( )
```

20.61.1.2 extract_attribute()

```
def pysar.prep_roipac.extract_attribute (
    fname )
```

Sub Functions #####

Read/extract attributes for PySAR from ROI_PAC .unw, .int, .cor file.

For each unwrapped interferogram or spatial coherence file, there are 2 .rsc files:
basic metadata file and baseline parameter file.

```
e.g. filt_100901-110117-sim_HDR_4rlks_c10.unw
     filt_100901-110117-sim_HDR_4rlks_c10.unw.rsc
     100901-110117_baseline.rsc
```

Inputs:

```
fname : string, ROI_PAC interferogram filename or path,
       i.e. /KujuT422F650AlosA/filt_100901-110117-sim_HDR_4rlks_c10.unw
```

Outputs:

```
atr : dict, Attributes dictionary
```

20.61.1.3 main()

```
def pysar.prep_roipac.main (
    argv )
```

20.61.2 Variable Documentation

20.61.2.1 DESCRIPTION

DESCRIPTION

20.61.2.2 EXAMPLE

EXAMPLE

20.62 pysar.pysarApp Namespace Reference

Functions

- def [check_geocode_file](#) (lookupFile, File, templateFile=None, outFile=None)
- def [check_subset_file](#) (File, inps_dict, outFile=None, overwrite=False)
- def [subset_dataset](#) (inps, template_file)
- def [multilook_dataset](#) (inps, lks_y=None, lks_x=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [LOGO](#)
- [TEMPLATE](#)
- [EXAMPLE](#)
- [UM_FILE_STRUCT](#)

20.62.1 Function Documentation

20.62.1.1 check_geocode_file()

```
def pysar.pysarApp.check_geocode_file (
    lookupFile,
    File,
    templateFile = None,
    outFile = None )
```

Geocode input file or use existed geocoded file.

20.62.1.2 check_subset_file()

```
def pysar.pysarApp.check_subset_file (
    File,
    inps_dict,
    outFile = None,
    overwrite = False )
```

Subset input file or use existed subsetting file.

20.62.1.3 cmdLineParse()

```
def pysar.pysarApp.cmdLineParse ( )
```

20.62.1.4 main()

```
def pysar.pysarApp.main (
    argv )
```

20.62.1.5 multilook_dataset()

```
def pysar.pysarApp.multilook_dataset (
    inps,
    lks_y = None,
    lks_x = None )
```

Create a multilooked dataset

20.62.1.6 subset_dataset()

```
def pysar.pysarApp.subset_dataset (
    inps,
    template_file )
```

Create/prepare subset of datasets in different folder for time series analysis.

- 1) Read subset info from lat/lon or y/x, and convert into y/x
where lat/lon > y/x in priority unless a) no lookup file AND b) dataset is in radar coord
While converting lalo to yx, yx should be the bounding box of lalo.
- 2) for geo-coord dataset, use y/x from 1) to subset all the files
for radar-coord dataset, use y/x from 1) to subset all radar-coord files; then get y/x bounding box
in lat/lon and use it to subset all geo-coord files.

20.62.2 Variable Documentation

20.62.2.1 EXAMPLE

EXAMPLE

20.62.2.2 LOGO

LOGO

20.62.2.3 TEMPLATE

TEMPLATE

20.62.2.4 UM_FILE_STRUCT

UM_FILE_STRUCT

20.63 pysar.quality_map Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.63.1 Function Documentation

20.63.1.1 main()

```
def pysar.quality_map.main (  
    argv )
```

20.63.1.2 usage()

```
def pysar.quality_map.usage ( )
```

20.64 pysar.range_distance Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.64.1 Function Documentation

20.64.1.1 main()

```
def pysar.range_distance.main (
    argv )
```

20.64.1.2 usage()

```
def pysar.range_distance.usage ( )
```

20.65 pysar.reference_epoch Namespace Reference

Functions

- def [ref_date_attribute](#) (atr_in, ref_date, date_list)
- def [ref_date_file](#) (inFile, ref_date, outFile=None)
- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

20.65.1 Function Documentation

20.65.1.1 cmdLineParse()

```
def pysar.reference_epoch.cmdLineParse ( )
```

20.65.1.2 main()

```
def pysar.reference_epoch.main (
    argv )
```

20.65.1.3 read_template2inps()

```
def pysar.reference_epoch.read_template2inps (
    templateFile,
    inps = None )
```

Update inps with options from templateFile

20.65.1.4 ref_date_attribute()

```
def pysar.reference_epoch.ref_date_attribute (
    atr_in,
    ref_date,
    date_list )
```

Update attribute dictionary for reference date

20.65.1.5 ref_date_file()

```
def pysar.reference_epoch.ref_date_file (
    inFile,
    ref_date,
    outFile = None )
```

Change input file reference date to a different one.

20.65.2 Variable Documentation

20.65.2.1 EXAMPLE

EXAMPLE

20.65.2.2 TEMPLATE

TEMPLATE

20.66 pysar.remove_plane Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.66.1 Function Documentation

20.66.1.1 cmdLineParse()

```
def pysar.remove_plane.cmdLineParse ( )
```

20.66.1.2 main()

```
def pysar.remove_plane.main (
    argv )
```

20.66.2 Variable Documentation

20.66.2.1 EXAMPLE

EXAMPLE

20.67 pysar.rewrap Namespace Reference

Functions

- def [usage](#) ()
- def [rewrap](#) (unw, cycle=2 *np.pi)
- def [main](#) (argv)

20.67.1 Function Documentation

20.67.1.1 main()

```
def pysar.rewrap.main (
    argv )
```


20.67.1.2 rewrap()

```
def pysar.rewrap.rewrap (
    unw,
    cycle = 2*np.pi )
```

20.67.1.3 usage()

```
def pysar.rewrap.usage ( )
```

20.68 pysar.save_gmt Namespace Reference

Functions

- def [get_geo_lat_lon](#) (atr)
- def [write_grd_file](#) (data, atr, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.68.1 Function Documentation

20.68.1.1 cmdLineParse()

```
def pysar.save_gmt.cmdLineParse ( )
```

20.68.1.2 get_geo_lat_lon()

```
def pysar.save_gmt.get_geo_lat_lon (
    atr )
```

20.68.1.3 main()

```
def pysar.save_gmt.main (
    argv )
```

20.68.1.4 write_grd_file()

```
def pysar.save_gmt.write_grd_file (
    data,
    atr,
    fname_out = None )
```

Write GMT .grd file for input data matrix, using giant._gmt module.

Inputs:

```
data - 2D np.array in int/float, data matrix to write
atr - dict, attributes of input data matrix
fname_out - string, output file name
```

Output:

```
fname_out - string, output file name
```

20.68.2 Variable Documentation

20.68.2.1 EXAMPLE

EXAMPLE

20.69 pysar.save_hdfeos5 Namespace Reference

Functions

- def [get_mission_name](#) (meta_dict)
- def [metadata_pysar2unavco](#) (pysar_meta_dict, dateList)
- def [get_hdfeos5_filename](#) (timeseriesFile)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [BOOL_ZERO](#)
- [INT_ZERO](#)
- [FLOAT_ZERO](#)
- [CPX_ZERO](#)
- [TEMPALTE](#)
- [EXAMPLE](#)

20.69.1 Function Documentation

20.69.1.1 cmdLineParse()

```
def pysar.save_hdfeos5.cmdLineParse ( )
```

20.69.1.2 get_hdfeos5_filename()

```
def pysar.save_hdfeos5.get_hdfeos5_filename (
    timeseriesFile )
```

Get output file name of HDF-EOS5 time series file

20.69.1.3 get_mission_name()

```
def pysar.save_hdfeos5.get_mission_name (
    meta_dict )
```

Get mission name in UNAVCO InSAR Archive format from attribute mission/PLATFORM

Input: meta_dict : dict, attributes

Output: mission : string, mission name in standard UNAVCO format.

20.69.1.4 main()

```
def pysar.save_hdfeos5.main (
    argv )
```

20.69.1.5 metadata_pysar2unavco()

```
def pysar.save_hdfeos5.metadata_pysar2unavco (
    pysar_meta_dict,
    dateList )
```

20.69.1.6 read_template2inps()

```
def pysar.save_hdfeos5.read_template2inps (
    template_file,
    inps = None )
```

Read input template options into Namespace inps

20.69.2 Variable Documentation

20.69.2.1 BOOL_ZERO

BOOL_ZERO

20.69.2.2 CPX_ZERO

CPX_ZERO

20.69.2.3 EXAMPLE

EXAMPLE

20.69.2.4 FLOAT_ZERO

FLOAT_ZERO

20.69.2.5 INT_ZERO

INT_ZERO

20.69.2.6 TEMPALTE

TEMPALTE

20.70 pysar.save_kml Namespace Reference

Functions

- def [write_kmz_file](#) (data, atr, out_name_base, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.70.1 Function Documentation

20.70.1.1 cmdLineParse()

```
def pysar.save_kml.cmdLineParse ( )
```

20.70.1.2 main()

```
def pysar.save_kml.main (
    argv )
```

20.70.1.3 write_kmz_file()

```
def pysar.save_kml.write_kmz_file (
    data,
    atr,
    out_name_base,
    inps = None )
```

Generate Google Earth KMZ file for input data matrix.

Inputs:

```
data - 2D np.array in int/float, data matrix to write
out_name_base - string, output file name base
atr - dict, containing the following attributes:
    WIDTH/FILE_LENGTH : required, file size
    X/Y_FIRST/STEP    : required, for lat/lon spatial coverage
    ref_x/y           : optional, column/row number of reference pixel
    PROJECT_NAME      : optional, for KMZ folder name
inps - Namespace, optional, input options for display
```

Output:

```
kmz_file - string, output KMZ filename
```

Example:

```
import pysar._readfile as readfile
import pysar.view as pview
import pysar.save_kml as save_kml
fname = 'geo_velocity_masked.h5'
data, atr = readfile.read(fname)
out_name_base = pview.auto_figure_title(fname, None)
save_kml.write_kmz_file(data, atr, out_name_base)
```

20.70.2 Variable Documentation

20.70.2.1 EXAMPLE

EXAMPLE

20.71 pysar.save_mat Namespace Reference

Functions

- def [usage](#) ()
- def [yyyyymmdd2years](#) (date)
- def [main](#) (argv)

20.71.1 Function Documentation

20.71.1.1 `main()`

```
def pysar.save_mat.main (
    argv )
```

20.71.1.2 `usage()`

```
def pysar.save_mat.usage ( )
```

20.71.1.3 `yyyymmdd2years()`

```
def pysar.save_mat.yyyymmdd2years (
    date )
```

20.72 `pysar.save_roipac` Namespace Reference

Functions

- def [usage](#) ()
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.72.1 Function Documentation

20.72.1.1 `cmdLineParse()`

```
def pysar.save_roipac.cmdLineParse ( )
```

20.72.1.2 `main()`

```
def pysar.save_roipac.main (
    argv )
```

20.72.1.3 `usage()`

```
def pysar.save_roipac.usage ( )
```

20.72.2 Variable Documentation

20.72.2.1 EXAMPLE

EXAMPLE

20.73 pysar.seed_data Namespace Reference

Functions

- def [nearest](#) (x, tbase, xstep)
Sub Functions #####.
- def [seed_file_reference_value](#) (File, outName, refList, ref_y="", ref_x="")
- def [seed_file_inps](#) (File, inps=None, outFile=None)
- def [seed_attributes](#) (atr_in, x, y)
- def [manual_select_reference_yx](#) (stack, inps)
- def [select_max_coherence_yx](#) (cohFile, mask=None, min_coh=0.85)
- def [random_select_reference_yx](#) (data_mat, print_msg=True)
- def [print_warning](#) (next_method)
- def [read_seed_template2inps](#) (template_file, inps=None)
- def [read_seed_reference2inps](#) (reference_file, inps=None)
- def [remove_reference_pixel](#) (File)
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [TEMPLATE](#)
Usage #####.
- [NOTE](#)
- [EXAMPLE](#)

20.73.1 Function Documentation

20.73.1.1 cmdLineParse()

```
def pysar.seed_data.cmdLineParse ( )
```

20.73.1.2 main()

```
def pysar.seed_data.main (
    argv )
```

Main Function #####.

20.73.1.3 manual_select_reference_yx()

```
def pysar.seed_data.manual_select_reference_yx (
    stack,
    inps )
```

Input:

```
data4display : 2D np.array, stack of input file
inps         : namespace, with key 'ref_x' and 'ref_y', which will be updated
```

20.73.1.4 nearest()

```
def pysar.seed_data.nearest (
    x,
    tbase,
    xstep )
```

Sub Functions #####.

20.73.1.5 print_warning()

```
def pysar.seed_data.print_warning (
    next_method )
```

20.73.1.6 random_select_reference_yx()

```
def pysar.seed_data.random_select_reference_yx (
    data_mat,
    print_msg = True )
```

20.73.1.7 read_seed_reference2inps()

```
def pysar.seed_data.read_seed_reference2inps (
    reference_file,
    inps = None )
```

Read seed/reference info from reference file and update input namespace

20.73.1.8 read_seed_template2inps()

```
def pysar.seed_data.read_seed_template2inps (
    template_file,
    inps = None )
```

Read seed/reference info from template file and update input namespace

20.73.1.9 remove_reference_pixel()

```
def pysar.seed_data.remove_reference_pixel (
    File )
```

Remove reference pixel info from input file

20.73.1.10 seed_attributes()

```
def pysar.seed_data.seed_attributes (
    atr_in,
    x,
    y )
```

20.73.1.11 seed_file_inps()

```
def pysar.seed_data.seed_file_inps (
    File,
    inps = None,
    outFile = None )
```

Seed input file with option from input namespace
Return output file name if succeed; otherwise, return None

20.73.1.12 seed_file_reference_value()

```
def pysar.seed_data.seed_file_reference_value (
    File,
    outName,
    refList,
    ref_y = '',
    ref_x = '' )
```

20.73.1.13 select_max_coherence_yx()

```
def pysar.seed_data.select_max_coherence_yx (
    cohFile,
    mask = None,
    min_coh = 0.85 )
```

Select pixel with coherence > min_coh in random

20.73.2 Variable Documentation

20.73.2.1 EXAMPLE

EXAMPLE

20.73.2.2 NOTE

NOTE

20.73.2.3 TEMPLATE

TEMPLATE

Usage #####.

20.74 pysar.select_network Namespace Reference

Functions

- def [log](#) (msg)
- def [project_name2sensor](#) (projectName)
- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [sar_sensor_list](#)
- [REFERENCE](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

20.74.1 Function Documentation

20.74.1.1 cmdLineParse()

```
def pysar.select_network.cmdLineParse ( )
```

20.74.1.2 log()

```
def pysar.select_network.log (
    msg )
```

Log function writen by Falk

20.74.1.3 main()

```
def pysar.select_network.main (
    argv )
```

20.74.1.4 project_name2sensor()

```
def pysar.select_network.project_name2sensor (
    projectName )
```

20.74.1.5 read_template2inps()

```
def pysar.select_network.read_template2inps (
    templateFile,
    inps = None )
```

Read network options from template file into Namespace variable inps

20.74.2 Variable Documentation

20.74.2.1 EXAMPLE

EXAMPLE

20.74.2.2 REFERENCE

REFERENCE

20.74.2.3 sar_sensor_list

sar_sensor_list

20.74.2.4 TEMPLATE

TEMPLATE

20.75 pysar.spatial_average Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [EXAMPLE](#)
Usage #####.

20.75.1 Function Documentation

20.75.1.1 cmdLineParse()

```
def pysar.spatial_average.cmdLineParse ( )
```

20.75.1.2 main()

```
def pysar.spatial_average.main (
    argv )
```

Main Function #####.

20.75.2 Variable Documentation

20.75.2.1 EXAMPLE

EXAMPLE

Usage #####.

20.76 pysar.spatial_filter Namespace Reference

Functions

- def [filter_data](#) (data, filter_type, filter_par=None)
- def [filter_file](#) (fname, filter_type, filter_par=None, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.76.1 Function Documentation

20.76.1.1 cmdLineParse()

```
def pysar.spatial_filter.cmdLineParse ( )
```

20.76.1.2 filter_data()

```
def pysar.spatial_filter.filter_data (
    data,
    filter_type,
    filter_par = None )
```

Filter 2D matrix with selected filter

Inputs:

```
data      : 2D np.array, matrix to be filtered
filter_type : string, filter type
filter_par : string, optional, parameter for low/high pass filter
              for low/highpass_avg, it's kernel size in int
              for low/highpass_gaussain, it's sigma in float
```

Output:

```
data_filt : 2D np.array, matrix after filtering.
```

20.76.1.3 filter_file()

```
def pysar.spatial_filter.filter_file (
    fname,
    filter_type,
    filter_par = None,
    fname_out = None )
```

Filter 2D matrix with selected filter

Inputs:

```
fname      : string, name/path of file to be filtered
filter_type : string, filter type
filter_par  : string, optional, parameter for low/high pass filter
               for low/highpass_avg, it's kernel size in int
               for low/highpass_gaussian, it's sigma in float
```

Output:

```
fname_out   : string, optional, output file name/path
```

20.76.1.4 main()

```
def pysar.spatial_filter.main (
    argv )
```

20.76.2 Variable Documentation

20.76.2.1 EXAMPLE

EXAMPLE

20.77 pysar.stacking Namespace Reference

Functions

- def [cmdLineParse](#) ()
 - def [main](#) (argv)
- Main Function #####.*

Variables

- [EXAMPLE](#)
- Usage #####.*

20.77.1 Function Documentation

20.77.1.1 cmdLineParse()

```
def pysar.stacking.cmdLineParse ( )
```

20.77.1.2 main()

```
def pysar.stacking.main (
    argv )
```

Main Function #####.

20.77.2 Variable Documentation

20.77.2.1 EXAMPLE

EXAMPLE

Usage #####.

20.78 pysar.subset Namespace Reference

Functions

- def [coord_geo2radar](#) (geoCoordIn, atr, coordType)

Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')
- def [coord_radar2geo](#) (radarCoordIn, atr, coordType)

Inputs: radarCoord : coordinate (list) in row/col in int atr : dictionary of file attributes coordType : coordinate type: row, col, y, x.
- def [check_box_within_data_coverage](#) (pixel_box, atr_dict)
- def [subset_attribute](#) (atr_dict, subset_box, print_msg=True)
- def [get_coverage_box](#) (atr)
- def [read_subset_template2box](#) (templateFile)
- def [bbox_geo2radar](#) (geo_box, atr_rdr=dict(), lookupFile=None)
- def [bbox_radar2geo](#) (pix_box, atr_rdr=dict(), lookupFile=None)
- def [subset_box2inps](#) (inps, pix_box, geo_box)
- def [get_box_overlap_index](#) (box1, box2)
- def [subset_input_dict2box](#) (subset_dict, meta_dict)
- def [box_pixel2geo](#) (pixel_box, meta_dict)
- def [box_geo2pixel](#) (geo_box, meta_dict)
- def [subset_file](#) (File, subset_dict_input, outFile=None)
- def [subset_file_list](#) (fileList, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.78.1 Function Documentation

20.78.1.1 bbox_geo2radar()

```
def pysar.subset.bbox_geo2radar (
    geo_box,
    atr_rdr = dict(),
    lookupFile = None )
```

Calculate bounding box in x/y for file in radar coord, based on input geo box.

Inputs:

geo_box - tuple of 4 float, indicating the UL/LR lon/lat
 atr_rdr - dict, attributes of file in radar coord
 lookupFile - string, path of transformation file, i.e. geomap_4rlks.trans

Output:

pix_box - tuple of 4 int, indicating the UL/LR x/y of the bounding box in radar coord
 for the corresponding lat/lon coverage.

20.78.1.2 bbox_radar2geo()

```
def pysar.subset.bbox_radar2geo (
    pix_box,
    atr_rdr = dict(),
    lookupFile = None )
```

Calculate bounding box in lat/lon for file in geo coord, based on input radar/pixel box

Inputs:

pix_box - tuple of 4 int, indicating the UL/LR x/y
 atr_rdr - dict, attributes of file in radar coord
 lookupFile - string, path of transformation file, i.e. geomap_4rlks.trans

Output:

geo_box - tuple of 4 float, indicating the UL/LR lon/lat of the bounding box

20.78.1.3 box_geo2pixel()

```
def pysar.subset.box_geo2pixel (
    geo_box,
    meta_dict )
```

Convert geo_box to pixel_box

20.78.1.4 box_pixel2geo()

```
def pysar.subset.box_pixel2geo (
    pixel_box,
    meta_dict )
```

Convert pixel_box to geo_box

20.78.1.5 check_box_within_data_coverage()

```
def pysar.subset.check_box_within_data_coverage (
    pixel_box,
    atr_dict )
```

Check the subset box's conflict with data coverage

Inputs:

pixel_box : 4-tuple of int, indicating y/x coordinates of subset
atr : dictionary of file attributes

20.78.1.6 cmdLineParse()

```
def pysar.subset.cmdLineParse ( )
```

20.78.1.7 coord_geo2radar()

```
def pysar.subset.coord_geo2radar (
    geoCoordIn,
    atr,
    coordType )
```

Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')

20.78.1.8 coord_radar2geo()

```
def pysar.subset.coord_radar2geo (
    radarCoordIn,
    atr,
    coordType )
```

Inputs: radarCoord : coordinate (list) in row/col in int atr : dictionary of file attributes coordType : coordinate type: row, col, y, x.

Example: 32.104990 = coord_radar2geo(300, atr,'y') [130.5,131.4] = coord_radar2geo([1000,1500],atr,'x')

20.78.1.9 get_box_overlap_index()

```
def pysar.subset.get_box_overlap_index (
    box1,
    box2 )
```

Get index box overlap area of two input boxes

Inputs:

box1/2 : 4-tuple of int, indicating coverage of box1/2
defining in (x0, y0, x1, y1)

Outputs:

overlap_idx_box1/2 : 4-tuple of int, indicating index of overlap area in box1/2
defining in (idx_x0, idx_y0, idx_x1, idx_y1)

20.78.1.10 get_coverage_box()

```
def pysar.subset.get_coverage_box (
    atr )
```

Get Coverage Box of data in geo and pixel coordinates

Inputs: atr - dict, meta data dictionary

Outputs:

pix_box : 4-tuple of int, defining in (UL_X, UL_Y, LR_X, LR_Y)
geo_box : 4-tuple of float in lat/lon

20.78.1.11 main()

```
def pysar.subset.main (
    argv )
```

20.78.1.12 read_subset_template2box()

```
def pysar.subset.read_subset_template2box (
    templateFile )
```

Read pysar.subset.lalo/yx option from template file into box type
Return None if not specified.

20.78.1.13 subset_attribute()

```
def pysar.subset.subset_attribute (
    atr_dict,
    subset_box,
    print_msg = True )
```

Update attributes dictionary due to subset

Inputs:

atr_dict : dict, data attributes to update
subset_box : 4-tuple of int, subset box defined in (x0, y0, x1, y1)

Outputs:

atr : dict, updated data attributes

20.78.1.14 subset_box2inps()

```
def pysar.subset.subset_box2inps (
    inps,
    pix_box,
    geo_box )
```

Update inps.subset_y/x/lat/lon from pixel_box and geo_box

20.78.1.15 subset_file()

```
def pysar.subset.subset_file (
    File,
    subset_dict_input,
    outFile = None )
```

Subset file with

Inputs:

File : str, path/name of file
outFile : str, path/name of output file
subset_dict : dict, subsut parameter, including the following items:
subset_x : list of 2 int, subset in x direction, default=None
subset_y : list of 2 int, subset in y direction, default=None
subset_lat : list of 2 float, subset in lat direction, default=None
subset_lon : list of 2 float, subset in lon direction, default=None
fill_value : float, optional. filled value for area outside of data coverage. default=None
None/not-existed to subset within data coverage only.
tight : bool, tight subset or not, for lookup table file, i.e. geomap*.trans

Outputs:

outFile : str, path/name of output file;
outFile = 'subset_'+File, if File is in current directory;
outFile = File, if File is not in the current directory.

20.78.1.16 subset_file_list()

```
def pysar.subset.subset_file_list (
    fileList,
    inps )
```

Subset file list

20.78.1.17 subset_input_dict2box()

```
def pysar.subset.subset_input_dict2box (
    subset_dict,
    meta_dict )
```

Convert subset inputs dict into box in radar and/or geo coord.

Inputs:

```
subset_dict : dict, including the following 4 objects:
    subset_x   : list of 2 int,   subset in x direction,   default=None
    subset_y   : list of 2 int,   subset in y direction,   default=None
    subset_lat : list of 2 float, subset in lat direction, default=None
    subset_lon : list of 2 float, subset in lon direction, default=None
meta_dict    : dict, including the following items:
    'WIDTH'      : int
    'FILE_LENGTH': int
    'X_FIRST'     : float, optional
    'Y_FIRST'     : float, optional
    'X_STEP'      : float, optional
    'Y_STEP'      : float, optional
```

Outputs:

```
# box defined by 4-tuple of number, defining (left, upper, right, lower) coordinate,
#                                     (UL_X, UL_Y, LR_X, LR_Y )
pixel_box : 4-tuple of int, in pixel unit - 1
geo_box   : 4-tuple of float, in lat/lon unit - degree
            None if file is in radar coordinate.
```

example:

```
subset_dict = {'subset_x': None, 'subset_y': None, 'subset_lat': [30.5, 31.0], 'subset_lon': [130.0, 131.0]}
subset_dict = {'subset_x': [100, 1100], 'subset_y': [2050, 2550], 'subset_lat': None, 'subset_lon': None}
pixel_box   = subset_input_dict2box(subset_dict, pysar_meta_dict)[0]
pixel_box, geo_box = subset_input_dict2box(subset_dict, pysar_meta_dict)
```

20.78.2 Variable Documentation**20.78.2.1 EXAMPLE**

EXAMPLE

20.79 pysar.sum_epochs Namespace Reference**Functions**

- def [usage](#) ()
- def [main](#) (argv)

20.79.1 Function Documentation

20.79.1.1 main()

```
def pysar.sum_epochs.main (
    argv )
```

20.79.1.2 usage()

```
def pysar.sum_epochs.usage ( )
```

20.80 pysar.temporal_average Namespace Reference

Functions

- def [usage](#) ()
Usage #####.
- def [main](#) (argv)
Main Function #####.

20.80.1 Function Documentation

20.80.1.1 main()

```
def pysar.temporal_average.main (
    argv )
```

Main Function #####.

20.80.1.2 usage()

```
def pysar.temporal_average.usage ( )
```

Usage #####.

20.81 pysar.temporal_coherence Namespace Reference

Functions

- def [temporal_coherence](#) (timeseriesFile, ifgramFile)
- def [usage](#) ()
- def [main](#) (argv)

Variables

- [USAGE](#)
- [DESCRIPTION](#)
- [REFERENCE](#)
- [EXAMPLE](#)

20.81.1 Function Documentation

20.81.1.1 main()

```
def pysar.temporal_coherence.main (
    argv )
```

20.81.1.2 temporal_coherence()

```
def pysar.temporal_coherence.temporal_coherence (
    timeseriesFile,
    ifgramFile )
```

Calculate temporal coherence based on input timeseries file and interferograms file

Inputs:

timeseriesFile - string, path of time series file
ifgramFile - string, path of interferograms file

Output:

temp_coh - 2D np.array, temporal coherence in float32

20.81.1.3 usage()

```
def pysar.temporal_coherence.usage ( )
```

20.81.2 Variable Documentation

20.81.2.1 DESCRIPTION

DESCRIPTION

20.81.2.2 EXAMPLE

EXAMPLE

20.81.2.3 REFERENCE

REFERENCE

20.81.2.4 USAGE

USAGE

20.82 pysar.temporal_derivative Namespace Reference

Functions

- def [usage](#) ()
- def [main](#) (argv)

20.82.1 Function Documentation

20.82.1.1 [main\(\)](#)

```
def pysar.temporal_derivative.main (
    argv )
```

20.82.1.2 [usage\(\)](#)

```
def pysar.temporal_derivative.usage ( )
```

20.83 pysar.temporal_filter Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.83.1 Function Documentation

20.83.1.1 cmdLineParse()

```
def pysar.temporal_filter.cmdLineParse ( )
```

20.83.1.2 main()

```
def pysar.temporal_filter.main (
    argv )
```

20.83.2 Variable Documentation

20.83.2.1 EXAMPLE

EXAMPLE

20.84 pysar.timeseries2velocity Namespace Reference

Functions

- def [get_exclude_date](#) (inps, date_list_all)
- def [get_velocity_filename](#) (timeseries_file, template_file=None, vel_file='velocity.h5', inps=None)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [TEMPLATE](#)
- [DROP_DATE_TXT](#)

20.84.1 Function Documentation

20.84.1.1 cmdLineParse()

```
def pysar.timeseries2velocity.cmdLineParse ( )
```


20.84.1.2 get_exclude_date()

```
def pysar.timeseries2velocity.get_exclude_date (
    inps,
    date_list_all )
```

Get inps.ex_date full list

Inputs:

inps - Namespace,
date_list_all - list of string for all available date in YYYYMMDD format

Output:

inps.ex_date - list of string for exclude date in YYYYMMDD format

20.84.1.3 get_velocity_filename()

```
def pysar.timeseries2velocity.get_velocity_filename (
    timeseries_file,
    template_file = None,
    vel_file = 'velocity.h5',
    inps = None )
```

Get output velocity filename

Example: velocity_file = get_output_filename('timeseries_ECMWF_demErr_refDate.h5', 'KujuAlosAT422F650.template')

20.84.1.4 main()

```
def pysar.timeseries2velocity.main (
    argv )
```

20.84.1.5 read_template2inps()

```
def pysar.timeseries2velocity.read_template2inps (
    template_file,
    inps = None )
```

Read input template file into inps.ex_date

20.84.2 Variable Documentation

20.84.2.1 DROP_DATE_TXT

DROP_DATE_TXT

20.84.2.2 EXAMPLE

EXAMPLE

20.84.2.3 TEMPLATE

TEMPLATE

20.85 pysar.timeseries_rms Namespace Reference

Functions

- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

20.85.1 Function Documentation

20.85.1.1 cmdLineParse()

```
def pysar.timeseries_rms.cmdLineParse ( )
```

20.85.1.2 main()

```
def pysar.timeseries_rms.main (
    argv )
```

20.85.1.3 read_template2inps()

```
def pysar.timeseries_rms.read_template2inps (
    templateFile,
    inps = None )
```

Update inps with pysar.residualRms.* option from templateFile

20.85.2 Variable Documentation

20.85.2.1 EXAMPLE

EXAMPLE

20.85.2.2 TEMPLATE

TEMPLATE

20.86 pysar.transect Namespace Reference

Functions

- def [get_scale_from_disp_unit](#) (disp_unit, data_unit)
 - def [read_lonlat_file](#) (lonlat_file)
 - def [manual_select_start_end_point](#) (File)
 - def [transect_yx](#) (z, atr, start_yx, end_yx, interpolation='nearest')
 - def [transect_lalo](#) (z, atr, start_lalo, end_lalo, interpolation='nearest')
 - def [transect_list](#) (fileList, inps)
 - def [cmdLineParse](#) ()
 - def [main](#) (argv)
- Main #####.*

Variables

- [EXAMPLE](#)

20.86.1 Function Documentation

20.86.1.1 cmdLineParse()

```
def pysar.transect.cmdLineParse ( )
```

20.86.1.2 get_scale_from_disp_unit()

```
def pysar.transect.get_scale_from_disp_unit (
    disp_unit,
    data_unit )
```

20.86.1.3 main()

```
def pysar.transect.main (
    argv )
```

Main #####.

20.86.1.4 manual_select_start_end_point()

```
def pysar.transect.manual_select_start_end_point (
    File )
```

Manual Select Start/End Point in display figure.

20.86.1.5 read_lonlat_file()

```
def pysar.transect.read_lonlat_file (
    lonlat_file )
```

Read Start/End lat/lon from lonlat text file in gmt format.

Inputs:

lonlat_file : text file in gmt lonlat point file

Outputs:

start/end_lalo : list of 2 float

20.86.1.6 transect_lalo()

```
def pysar.transect.transect_lalo (
    z,
    atr,
    start_lalo,
    end_lalo,
    interpolation = 'nearest' )
```

Extract 2D matrix (z) value along the line [start_lalo, end_lalo]

20.86.1.7 transect_list()

```
def pysar.transect.transect_list (
    fileList,
    inps )
```

Get transection along input line from file list

Inputs:

fileList : list of str, path of files to get transect

inps : Namespace including the following items:

start/end_lalo

start/end_yx

interpolation

Outputs:

transectList : list of N*2 matrix containing distance and its value

atrList : list of attribute dictionary, for each input file

20.86.1.8 transect_yx()

```
def pysar.transect.transect_yx (
    z,
    atr,
    start_yx,
    end_yx,
    interpolation = 'nearest' )
```

Extract 2D matrix (z) value along the line [x0,y0;x1,y1]

Ref link: <http://stackoverflow.com/questions/7878398/how-to-extract-an-arbitrary-line-of-values-from-a-numpy-array>

Inputs:

```
z          - (np.array) 2D data matrix
atr        - (dictionary) 2D data matrix attribute dictionary
start_yx   - (list) y,x coordinate of start point
end_yx     - (list) y,x coordinate of end point
interpolation - sampling/interpolation method, including:
    'nearest' - nearest neighbour, by default
    'cubic'   - cubic interpolation
    'bilinear' - bilinear interpolation
```

Output:

```
transect - N*2 matrix containing distance - 1st col - and its corresponding
          values - 2nd col - along the line, N is the number of points.
```

Example:

```
transect = transect_yx(dem,demRsc,[10,15],[100,115])
```

20.86.2 Variable Documentation

20.86.2.1 EXAMPLE

EXAMPLE

20.87 pysar.transect_legacy Namespace Reference

Functions

- def [dms2d](#) (Coord)
- def [gps_to_LOS](#) (Ve, Vn, theta, heading)
- def [check_st_in_box](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [check_st_in_box2](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [line](#) (x0, y0, x1, y1)
- def [dist_point_from_line](#) (m, c, x, y, dx, dy)
- def [get_intersect](#) (m, c, x, y)
- def [readGPSfile](#) (gpsFile, gps_source)
- def [redGPSfile](#) (gpsFile)
- def [redGPSfile_cmm4](#) (gpsFile)
- def [nearest](#) (x, tbase, xstep)
- def [find_row_column](#) (Lon, Lat, lon, lat, lon_step, lat_step)
- def [get_lat_lon](#) (atr)
- def [nanmean](#) (data, args)
- def [nanstd](#) (data, args)
- def [get_transect](#) (z, x0, y0, x1, y1, interpolation='nearest')

Option: interpolation : sampling/interpolation method, including: 'nearest' - nearest neighbour, by default 'cubic' - cubic interpolation 'bilinear' - bilinear interpolation.
- def [Usage](#) ()
- def [main](#) (argv)
- def [onclick](#) (event)

Variables

- [fig](#)
- [ax](#)
- [xc](#)
- [yc](#)
- [cid](#)
- [x0](#)
- [x1](#)
- [y0](#)
- [y1](#)
- [mf](#)
- [cf](#)
- [df0](#)
- [df1](#)
- [mp](#)
- [Info_aboutFault](#)
- [length](#)
- [x](#)
- [y](#)
- [zi](#)
- [lat_transect](#)
- [lon_transect](#)
- [earth_radius](#)
- [dx](#)
- [dy](#)
- [DX](#)
- [DY](#)
- [D](#)
- [df0_km](#)
- [transect](#)
- [XX0](#)
- [XX1](#)
- [YY0](#)
- [YY1](#)
- [m](#)
- [c](#)
- [m1](#)
- [X0](#)
- [Y0](#)
- [X1](#)
- [Y1](#)
- [transect_lat](#)
- [transect_lon](#)
- [m_prof_edge](#)
- [c_prof_edge](#)
- [gpsFile](#)
- [insarData](#)
- [fileName](#)
- [fileExtension](#)
- [Stations](#)
- [Lat](#)
- [Lon](#)
- [Ve](#)
- [Se](#)

- [Vn](#)
- [Sn](#)
- [idxRef](#)
- [Length](#)
- [Width](#)
- [lat](#)
- [lon](#)
- [lat_step](#)
- [lon_step](#)
- [lat_all](#)
- [lon_all](#)
- [IDYref](#)
- [IDXref](#)
- [stationsList](#)
- [h5file__theta](#)
- [dset](#)
- [theta](#)
- [heading](#)
- [unitVec](#)
- [gpsLOS_ref](#)
- [GPS](#)
- [GPS_station](#)
- [GPSx](#)
- [GPSy](#)
- [GPS_lat](#)
- [GPS_lon](#)
- [idx](#)
- [IDY](#)
- [IDX](#)
- [gpsLOS](#)
- [NoInSAR](#)
- [DistGPS](#)
- [GPS_in_bound](#)
- [GPS_in_bound_st](#)
- [GPSxx](#)
- [GPSyy](#)
- [gx](#)
- [gy](#)
- [check_result](#)
- [check_result2](#)
- [dg](#)
- [axes](#)
- [nrows](#)
- [ms](#)
- [ax.fill_between\(D/1000.0, \(avgInSAR-stdInSAR\)*1000, \(avgInSAR+stdInSAR\)*1000,where=\(avgInSAR+stdInSAR\)*1000>=\(avgInSAR-stdInSAR\)*1000,alpha=1, facecolor='Red'\)](#)
- [avgInSAR](#)
- [axis](#)
- [stdInSAR](#)
- [fig2](#)
- [axes2](#)
- [FaultLine](#)
- [figName](#)

*Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minor←
Locator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)*

- [mfc](#)
- [linewidth](#)
- [matFile](#)
- [dataset](#)
- [color](#)

*ax.plot(D/1000.0, avglnSAR*1000, 'r-')*

- [alpha](#)
- [fontsize](#)
- [lbound](#)

lower and higher bounds for displaying the profile

- [hbound](#)
- [fault_loc](#)
- [ylim](#)

20.87.1 Function Documentation

20.87.1.1 `check_st_in_box()`

```
def pysar.transect_legacy.check_st_in_box (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```

20.87.1.2 `check_st_in_box2()`

```
def pysar.transect_legacy.check_st_in_box2 (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```


20.87.1.3 dist_point_from_line()

```
def pysar.transect_legacy.dist_point_from_line (
    m,
    c,
    x,
    y,
    dx,
    dy )
```

20.87.1.4 dms2d()

```
def pysar.transect_legacy.dms2d (
    Coord )
```

20.87.1.5 find_row_column()

```
def pysar.transect_legacy.find_row_column (
    Lon,
    Lat,
    lon,
    lat,
    lon_step,
    lat_step )
```

20.87.1.6 get_intersect()

```
def pysar.transect_legacy.get_intersect (
    m,
    c,
    x,
    y )
```

20.87.1.7 get_lat_lon()

```
def pysar.transect_legacy.get_lat_lon (
    atr )
```

20.87.1.8 get_transect()

```
def pysar.transect_legacy.get_transect (
    z,
    x0,
    y0,
    x1,
    y1,
    interpolation = 'nearest' )
```

Option: interpolation : sampling/interpolation method, including: 'nearest' - nearest neighbour, by default 'cubic' - cubic interpolation 'bilinear' - bilinear interpolation.

20.87.1.9 gps_to_LOS()

```
def pysar.transect_legacy.gps_to_LOS (
    Ve,
    Vn,
    theta,
    heading )
```

20.87.1.10 line()

```
def pysar.transect_legacy.line (
    x0,
    y0,
    x1,
    y1 )
```

20.87.1.11 main()

```
def pysar.transect_legacy.main (
    argv )
```

20.87.1.12 nanmean()

```
def pysar.transect_legacy.nanmean (
    data,
    args )
```

20.87.1.13 nanstd()

```
def pysar.transect_legacy.nanstd (
    data,
    args )
```

20.87.1.14 nearest()

```
def pysar.transect_legacy.nearest (
    x,
    tbase,
    xstep )
```

20.87.1.15 onclick()

```
def pysar.transect_legacy.onclick (
    event )
```

20.87.1.16 readGPSfile()

```
def pysar.transect_legacy.readGPSfile (
    gpsFile,
    gps_source )
```

20.87.1.17 redGPSfile()

```
def pysar.transect_legacy.redGPSfile (
    gpsFile )
```

20.87.1.18 redGPSfile_cmm4()

```
def pysar.transect_legacy.redGPSfile_cmm4 (
    gpsFile )
```

20.87.1.19 Usage()

```
def pysar.transect_legacy.Usage ( )
```

20.87.2 Variable Documentation**20.87.2.1 alpha**

alpha

20.87.2.2 avgInSAR

avgInSAR

20.87.2.3 ax

ax

20.87.2.4 axes

axes

20.87.2.5 axes2

axes2

20.87.2.6 axis

axis

20.87.2.7 c

c

20.87.2.8 c_prof_edge

c_prof_edge

20.87.2.9 cf

cf

20.87.2.10 check_result

check_result

20.87.2.11 check_result2

check_result2

20.87.2.12 cid

cid

20.87.2.13 color

color

```
ax.plot(D/1000.0, avglnSAR*1000, 'r-')
```

To plot the Fault location on the profile.

20.87.2.14 D

D

20.87.2.15 dataset

dataset

20.87.2.16 df0

df0

20.87.2.17 df0_km

df0_km

20.87.2.18 df1

df1

20.87.2.19 dg

dg

20.87.2.20 DistGPS

DistGPS

20.87.2.21 dset

dset

20.87.2.22 dx

dx

20.87.2.23 DX

DX

20.87.2.24 dy

dy

20.87.2.25 DY

DY

20.87.2.26 earth_radius

earth_radius

20.87.2.27 fault_loc

fault_loc

20.87.2.28 FaultLine

FaultLine

20.87.2.29 fig

fig

20.87.2.30 fig2

fig2

20.87.2.31 figName

figName

Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minorLocator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)

20.87.2.32 fileExtension

fileExtension

20.87.2.33 fileName

fileName

20.87.2.34 fontsize

fontsize

20.87.2.35 GPS

GPS

20.87.2.36 GPS_in_bound

GPS_in_bound

20.87.2.37 GPS_in_bound_st

GPS_in_bound_st

20.87.2.38 GPS_lat

GPS_lat

20.87.2.39 GPS_lon

GPS_lon

20.87.2.40 GPS_station

GPS_station

20.87.2.41 gpsFile

gpsFile

20.87.2.42 gpsLOS

gpsLOS

20.87.2.43 gpsLOS_ref

gpsLOS_ref

20.87.2.44 GPSx

GPSx

20.87.2.45 GPSxx

GPSxx

20.87.2.46 GPSy

GPSy

20.87.2.47 GPSyy

GPSyy

20.87.2.48 gx

gx

20.87.2.49 gy

gy

20.87.2.50 h5file_theta

h5file_theta

20.87.2.51 hbound

hbound

20.87.2.52 heading

heading

20.87.2.53 idx

idx

20.87.2.54 IDX

IDX

20.87.2.55 idxRef

idxRef

20.87.2.56 IDXref

IDXref

20.87.2.57 IDY

IDY

20.87.2.58 IDYref

IDYref

20.87.2.59 Info_aboutFault

Info_aboutFault

20.87.2.60 insarData

insarData

20.87.2.61 Lat

Lat

20.87.2.62 lat

lat

20.87.2.63 lat_all

lat_all

20.87.2.64 lat_step

lat_step

20.87.2.65 lat_transect

lat_transect

20.87.2.66 lbound

lbound

lower and higher bounds for displaying the profile

20.87.2.67 length

length

20.87.2.68 Length

Length

20.87.2.69 linewidth

linewidth

20.87.2.70 Lon

Lon

20.87.2.71 lon

lon

20.87.2.72 lon_all

lon_all

20.87.2.73 lon_step

lon_step

20.87.2.74 lon_transect

lon_transect

20.87.2.75 m

m

20.87.2.76 m1

m1

20.87.2.77 m_prof_edge

m_prof_edge

20.87.2.78 matFile

matFile

20.87.2.79 mf

mf

20.87.2.80 mfc

mfc

20.87.2.81 mp

mp

20.87.2.82 ms

ms

```
ax.fill_between(D/1000.0, (avglnSAR-stdlnSAR)*1000, (avglnSAR+stdlnSAR)*1000,where=(avglnSAR+stdlnSAR)*1000>=(avglnSAR-stdlnSAR)*1000,alpha=1, facecolor='Red')
```

20.87.2.83 NoInSAR

NoInSAR

20.87.2.84 nrows

nrows

20.87.2.85 Se

Se

20.87.2.86 Sn

Sn

20.87.2.87 Stations

Stations

20.87.2.88 stationsList

stationsList

20.87.2.89 stdInSAR

stdInSAR

20.87.2.90 theta

theta

20.87.2.91 transect

transect

20.87.2.92 transect_lat

transect_lat

20.87.2.93 transect_lon

transect_lon

20.87.2.94 unitVec

unitVec

20.87.2.95 Ve

Ve

20.87.2.96 Vn

Vn

20.87.2.97 Width

Width

20.87.2.98 x

x

20.87.2.99 x0

x0

20.87.2.100 X0

X0

20.87.2.101 x1

x1

20.87.2.102 X1

X1

20.87.2.103 xc

xc

20.87.2.104 XX0

XX0

20.87.2.105 XX1

XX1

20.87.2.106 y

y

20.87.2.107 y0

y0

20.87.2.108 Y0

Y0

20.87.2.109 y1

y1

20.87.2.110 Y1

Y1

20.87.2.111 yc

yc

20.87.2.112 ylim

ylim

20.87.2.113 YY0

YY0

20.87.2.114 YY1

YY1

20.87.2.115 zi

zi

20.88 pysar.tropcor_phase_elevation Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [REFERENCE](#)

20.88.1 Function Documentation

20.88.1.1 cmdLineParse()

```
def pysar.tropcor_phase_elevation.cmdLineParse ( )
```

20.88.1.2 main()

```
def pysar.tropcor_phase_elevation.main (
    argv )
```

20.88.2 Variable Documentation

20.88.2.1 EXAMPLE

EXAMPLE

20.88.2.2 REFERENCE

REFERENCE

20.89 pysar.tropcor_pyaps Namespace Reference

Functions

- def [get_delay](#) (grib_file, atr, inps_dict)
- def [date_list2grib_file](#) (date_list, hour, grib_source, grib_dir)
- def [dload_grib](#) (date_list, hour, grib_source='ECMWF', weather_dir='.')
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [REFERENCE](#)
- [TEMPLATE](#)
- [DATA_INFO](#)

20.89.1 Function Documentation

20.89.1.1 cmdLineParse()

```
def pysar.tropcor_pyaps.cmdLineParse ( )
```

20.89.1.2 date_list2grib_file()

```
def pysar.tropcor_pyaps.date_list2grib_file (
    date_list,
    hour,
    grib_source,
    grib_dir )
```

20.89.1.3 dload_grib()

```
def pysar.tropcor_pyaps.dload_grib (
    date_list,
    hour,
    grib_source = 'ECMWF',
    weather_dir = './' )
```

Download weather re-analysis grib files using PyAPS

Inputs:

date_list : list of string in YYYYMMDD format
hour : string in HH:MM or HH format
grib_source : string,
weather_dir : string,

Output:

grib_file_list : list of string

20.89.1.4 get_delay()

```
def pysar.tropcor_pyaps.get_delay (
    grib_file,
    atr,
    inps_dict )
```

Get delay matrix using PyAPS for one acquisition

Inputs:

```
grib_file - strng, grib file path
atr       - dict, including the following attributes:
    dem_file      - string, DEM file path
    grib_source   - string, Weather re-analysis data source
    delay_type    - string, comb/dry/wet
    ref_y/x       - string, reference pixel row/col number
    inc_angle     - np.array, 0/1/2 D
```

Output:

```
phs - 2D np.array, absolute tropospheric phase delay relative to ref_y/x
```

20.89.1.5 main()

```
def pysar.tropcor_pyaps.main (
    argv )
```

20.89.2 Variable Documentation

20.89.2.1 DATA_INFO

DATA_INFO

20.89.2.2 EXAMPLE

EXAMPLE

20.89.2.3 REFERENCE

REFERENCE

20.89.2.4 TEMPLATE

TEMPLATE

20.90 pysar.tsviewer Namespace Reference

Functions

- def [read_timeseries_yx](#) (timeseries_file, y, x, ref_yx=None)
- def [read_timeseries_lalo](#) (timeseries_file, lat, lon)
- def [cmdLineParse](#) ()
- def [format_coord](#) (x, y)
- def [time_slider_update](#) (val)
- def [plot_timeseries_errorbar](#) (ax, dis_ts, inps)
- def [plot_timeseries_scatter](#) (ax, dis_ts, inps)
- def [update_timeseries](#) (y, x)
- def [plot_timeseries_event](#) (event)

Variables

- [EXAMPLE](#)
- [inps](#)

Actual code.

- [atr](#)
- [k](#)
- [h5](#)
- [dateList](#)
- [date_num](#)
- [dates](#)
- [tims](#)
- [input_ex_date](#)
- [ex_date_list](#)
- [ex_date](#)
- [ex_dates](#)
- [ex_idx_list](#)
- [zero_idx](#)

Zero displacement for 1st acquisition.

- [length](#)
- [width](#)
- [ullon](#)
- [ullat](#)
- [lon_step](#)
- [lat_step](#)
- [lrlon](#)
- [lrlat](#)
- [y](#)
- [x](#)
- [yx](#)
- [ref_yx](#)
- [unit_fac](#)
- [flip_ud](#)
- [left_lr](#)
- [file_list](#)
- [mask_file](#)
- [mask](#)
- [epoch](#)
- [d_v](#)

- [timeseries_file](#)
- [ref_d_v](#)
- [data_lim](#)
- [ylim_mat](#)
- [fig_v](#)

Fig 1 - Cumulative Displacement Map.

- [ax_v](#)
- [dem](#)
- [dem_file](#)
- [img](#)
- [cmap](#)
- [colormap](#)
- [clim](#)
- [interpolation](#)
- [ms](#)
- [markeredgecolor](#)
- [format_coord](#)
- [cbar](#)
- [orientation](#)
- [ax_time](#)
- [axisbg](#)
- [yticks](#)
- [tslider](#)
- [valinit](#)
- [facecolor](#)
- [ecolor](#)
- [fig_ts](#)

Fig 2 - Time Series Displacement - Point.

- [figsize](#)
- [ax_ts](#)
- [error_ts](#)
- [error_fileContent](#)
- [error_file](#)
- [dtype](#)
- [e_ts](#)
- [ex_error_ts](#)
- [d_ts](#)
- [fig_base](#)

Output.

- [outName](#) = `inps.fig_base+'_ts.pdf'`
- [header_info](#)
- [lat](#)
- [lon](#)
- [fmt](#)
- `string delimiter = header_info)`
- [bbox_inches](#)
- [transparent](#)
- [True](#)
- [dpi](#)
- `cid = fig_v.canvas.mpl_connect('button_press_event', plot_timeseries_event)`

Final linking of the canvas to the plots.

20.90.1 Function Documentation

20.90.1.1 cmdLineParse()

```
def pysar.tsviewer.cmdLineParse ( )
```

20.90.1.2 format_coord()

```
def pysar.tsviewer.format_coord (
    x,
    y )
```

20.90.1.3 plot_timeseries_errorbar()

```
def pysar.tsviewer.plot_timeseries_errorbar (
    ax,
    dis_ts,
    inps )
```

20.90.1.4 plot_timeseries_event()

```
def pysar.tsviewer.plot_timeseries_event (
    event )
```

Event function to get y/x from button press

20.90.1.5 plot_timeseries_scatter()

```
def pysar.tsviewer.plot_timeseries_scatter (
    ax,
    dis_ts,
    inps )
```

20.90.1.6 read_timeseries_lalo()

```
def pysar.tsviewer.read_timeseries_lalo (
    timeseries_file,
    lat,
    lon )
```

Read time-series displacement on point (y,x) from timeseries_file

Inputs:

timeseries_file : string, name/path of timeseries hdf5 file
lat/lon : float, latitude/longitude of point of interest

Output:

dis_ts : list of float, displacement time-series of point of interest

20.90.1.7 read_timeseries_yx()

```
def pysar.tsviewer.read_timeseries_yx (
    timeseries_file,
    y,
    x,
    ref_yx = None )
```

Read time-series displacement on point (y,x) from timeseries_file

Inputs:

timeseries_file : string, name/path of timeseries hdf5 file
y/x : int, row/column number of point of interest

Output:

dis_ts : list of float, displacement time-series of point of interest

20.90.1.8 time_slider_update()

```
def pysar.tsviewer.time_slider_update (
    val )
```

Update Displacement Map using Slider

20.90.1.9 update_timeseries()

```
def pysar.tsviewer.update_timeseries (
    y,
    x )
```

Plot point time series displacement at pixel [y, x]

20.90.2 Variable Documentation

20.90.2.1 atr

`atr`

20.90.2.2 ax_time

`ax_time`

20.90.2.3 ax_ts

`ax_ts`

20.90.2.4 ax_v

`ax_v`

20.90.2.5 axisbg

`axisbg`

20.90.2.6 bbox_inches

`bbox_inches`

20.90.2.7 cbar

`cbar`

20.90.2.8 cid

```
cid = fig_v.canvas.mpl_connect('button_press_event', plot\_timeseries\_event)
```

Final linking of the canvas to the plots.

20.90.2.9 clim

clim

20.90.2.10 cmap

cmap

20.90.2.11 colormap

colormap

20.90.2.12 d_ts

d_ts

20.90.2.13 d_v

d_v

20.90.2.14 data_lim

data_lim

20.90.2.15 date_num

date_num

20.90.2.16 dateList

dateList

20.90.2.17 dates

dates

20.90.2.18 delimiter

```
string delimiter = header_info)
```

20.90.2.19 dem

```
dem
```

20.90.2.20 dem_file

```
dem_file
```

20.90.2.21 dpi

```
dpi
```

20.90.2.22 dtype

```
dtype
```

20.90.2.23 e_ts

```
e_ts
```

20.90.2.24 ecolor

```
ecolor
```

20.90.2.25 epoch

```
epoch
```

20.90.2.26 error_file

```
error_file
```

20.90.2.27 error_fileContent

error_fileContent

20.90.2.28 error_ts

error_ts

20.90.2.29 ex_date

ex_date

20.90.2.30 ex_date_list

ex_date_list

20.90.2.31 ex_dates

ex_dates

20.90.2.32 ex_error_ts

ex_error_ts

20.90.2.33 ex_idx_list

ex_idx_list

20.90.2.34 EXAMPLE

EXAMPLE

20.90.2.35 facecolor

facecolor

20.90.2.36 fig_base

`fig_base`

Output.

20.90.2.37 fig_ts

`fig_ts`

Fig 2 - Time Series Displacement - Point.

20.90.2.38 fig_v

`fig_v`

Fig 1 - Cumulative Displacement Map.

20.90.2.39 figsize

`figsize`

20.90.2.40 file_list

`file_list`

20.90.2.41 flip_ud

`flip_ud`

20.90.2.42 fmt

`fmt`

20.90.2.43 format_coord

`format_coord`

20.90.2.44 h5

h5

20.90.2.45 header_info

header_info

20.90.2.46 img

img

20.90.2.47 inps

inps

Actual code.

20.90.2.48 input_ex_date

input_ex_date

20.90.2.49 interpolation

interpolation

20.90.2.50 k

k

20.90.2.51 lat

lat

20.90.2.52 lat_step

lat_step

20.90.2.53 left_lr

left_lr

20.90.2.54 length

length

20.90.2.55 lon

lon

20.90.2.56 lon_step

lon_step

20.90.2.57 lrlat

lrlat

20.90.2.58 lrlon

lrlon

20.90.2.59 markeredgecolor

markeredgecolor

20.90.2.60 mask

mask

20.90.2.61 mask_file

mask_file

20.90.2.62 ms

ms

20.90.2.63 orientation

orientation

20.90.2.64 outName

```
string outName = inps.fig_base+'_ts.pdf'
```

20.90.2.65 ref_d_v

ref_d_v

20.90.2.66 ref_yx

ref_yx

20.90.2.67 timeseries_file

timeseries_file

20.90.2.68 tims

tims

20.90.2.69 transparent

transparent

20.90.2.70 True

True

20.90.2.71 tslider

tslider

20.90.2.72 ullat

ullat

20.90.2.73 ullon

ullon

20.90.2.74 unit_fac

unit_fac

20.90.2.75 valinit

valinit

20.90.2.76 width

width

20.90.2.77 x

x

20.90.2.78 y

y

20.90.2.79 ylim_mat

ylim_mat

20.90.2.80 yticks

yticks

20.90.2.81 yx

yx

20.90.2.82 zero_idx

zero_idx

Zero displacement for 1st acquisition.

20.91 pysar.unwrap_error Namespace Reference

Functions

- def [bridging_data](#) (data, mask, x, y)
- def [unwrap_error_correction_phase_closure](#) (ifgram_file, mask_file, ifgram_cor_file=None)
- def [unwrap_error_correction_bridging](#) (ifgram_file, mask_file, y_list, x_list, ramp_type='plane', ifgram_cor_↔
file=None, save_cor_deramp_file=False)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- string [EXAMPLE](#)
- string [TEMPLATE](#)
- string [REFERENCE](#)
- string [DESCRIPTION](#)

20.91.1 Function Documentation

20.91.1.1 bridging_data()

```
def pysar.unwrap_error.bridging_data (
    data,
    mask,
    x,
    y )
```

Phase Jump Correction, using phase continuity on bridge/bonding points in each pair of patches.

Inputs:

```
data : 2D np.array, phase matrix need to be corrected
mask : mask file marks different patches with different positive integers
x/y   : list of int, array of bridge points, listed as: x_ref, x, x_ref, x
```

Output:

```
data : 2D np.array, phase corrected matrix
```

20.91.1.2 cmdLineParse()

```
def pysar.unwrap_error.cmdLineParse ( )
```

20.91.1.3 main()

```
def pysar.unwrap_error.main (
    argv )
```

20.91.1.4 read_template2inps()

```
def pysar.unwrap_error.read_template2inps (
    template_file,
    inps = None )
```

Read input template options into Namespace inps

20.91.1.5 unwrap_error_correction_bridging()

```
def pysar.unwrap_error.unwrap_error_correction_bridging (
    ifgram_file,
    mask_file,
    y_list,
    x_list,
    ramp_type = 'plane',
    ifgram_cor_file = None,
    save_cor_deramp_file = False )
```

Unwrapping error correction with bridging.

Inputs:

```
ifgram_file : string, name/path of interferogram(s) to be corrected
mask_file   : string, name/path of mask file to mark different patches
y/x_list    : list of int, bonding points in y/x
ifgram_cor_file : string, optional, output file name
save_cor_deramp_file : bool, optional
```

Output:

```
ifgram_cor_file
```

Example:

```
y_list = [235, 270, 350, 390]
x_list = [880, 890, 1200, 1270]
unwrap_error_correction_bridging('unwrapIfgram.h5', 'mask_all.h5', y_list, x_list, 'quadratic')
```

20.91.1.6 unwrap_error_correction_phase_closure()

```
def pysar.unwrap_error.unwrap_error_correction_phase_closure (
    ifgram_file,
    mask_file,
    ifgram_cor_file = None )
```

Correct unwrapping errors in network of interferograms using phase closure.

Inputs:

```
ifgram_file      - string, name/path of interferograms file
mask_file        - string, name/path of mask file to mask the pixels to be corrected
ifgram_cor_file  - string, optional, name/path of corrected interferograms file
```

Output:

```
ifgram_cor_file
```

Example:

```
'unwrapIfgram_unwCor.h5' = unwrap_error_correction_phase_closure('Seeded_unwrapIfgram.h5','mask.h5')
```

20.91.2 Variable Documentation

20.91.2.1 DESCRIPTION

```
string DESCRIPTION
```

20.91.2.2 EXAMPLE

```
string EXAMPLE
```

Initial value:

```
1 = '''example:
2 Phase Closure:
3  unwrap_error.py  Seeded_unwrapIfgram.h5  --mask mask.h5
4 Bridging:
5  unwrap_error.py  unwrapIfgram.h5        -t ShikokuT417F650_690AlosA.template
6  unwrap_error.py  unwrapIfgram.h5        --mask mask.h5      -x 283 305 -y 1177 1247
7  unwrap_error.py  081018_090118.unw      --mask mask_all.h5 -x 283 305 -y 1177 1247 --ramp quadratic
8 '''
```

20.91.2.3 REFERENCE

```
string REFERENCE
```

Initial value:

```
1 = '''reference:
2  Fattahi, H. (2015), Geodetic Imaging of Tectonic Deformation with InSAR, 190 pp, University of Miami,
   Miami, FL.
3 '''
```

20.91.2.4 TEMPLATE

string TEMPLATE

Initial value:

```
1 = '''
2 ## 4. Unwrapping Error Correction
3 ## unwrapping error correction based on the following two methods:
4 ## a. phase closure (Fattahi, 2015, PhD Thesis)
5 ## b. connecting bridge
6 pysar.unwrapError.method = auto    #[bridging / phase_closure / no], auto for no
7 pysar.unwrapError.maskFile = auto  #[file name / no], auto for no
8 pysar.unwrapError.ramp = auto      #[plane / quadratic], auto for plane
9 pysar.unwrapError.yx = auto        #[yl_start,xl_start,yl_end,xl_end;y2_start,...], auto for none
10 '''
```

20.92 pysar.view Namespace Reference

Classes

- class [Basemap2](#)

Class #####

Functions

- def [round_to_1](#) (x)
- def [add_inner_title](#) (ax, title, loc, size=None, kwargs)
- def [auto_flip_direction](#) (atr_dict)
- def [auto_figure_title](#) (fname, epoch=[], inps_dict=None)
- def [auto_row_col_num](#) (subplot_num, data_shape, fig_size, fig_num=1)
- def [check_colormap_input](#) (atr_dict, colormap=None)
- def [check_multilook_input](#) (pixel_box, row_num, col_num)
- def [get_epoch_full_list_from_input](#) (all_epoch_list, epoch_input_list=[], epoch_num_input_list=[])
- def [plot_dem_lalo](#) (bmap, dem, box, inps_dict)
- def [plot_dem_yx](#) (ax, dem, inps_dict=dict())
- def [scale_data4disp_unit_and_rewrap](#) (data, atr, disp_unit=None, rewrapping=False)
- def [scale_data2disp_unit](#) (matrix, atr_dict, disp_unit)
- def [update_plot_inps_with_display_setting_file](#) (inps, disp_set_file)
- def [update_plot_inps_with_meta_dict](#) (inps, meta_dict)
- def [update_matrix_with_plot_inps](#) (data, meta_dict, inps)
- def [plot_matrix](#) (ax, data, meta_dict, inps=None)
- def [cmdLineParse](#) (argv)
- def [main](#) (argv)

Main Function #####

Variables

- list [mplColors](#)
- string [EXAMPLE](#)
- string [PLOT_TEMPLATE](#)

20.92.1 Function Documentation

20.92.1.1 add_inner_title()

```
def pysar.view.add_inner_title (
    ax,
    title,
    loc,
    size = None,
    kwargs )
```

20.92.1.2 auto_figure_title()

```
def pysar.view.auto_figure_title (
    fname,
    epoch = [],
    inps_dict = None )
```

Get auto figure title from meta dict and input options

Inputs:

- fname - string, input file name
- epoch - list of string, optional, epoch to read for multi dataset/group files
- inps_dict - dict, optional, processing attributes, including:
 - ref_date
 - pix_box
 - wrap
 - disp_scale
 - opposite

Output:

- fig_title - string, output figure title

Example:

```
'geo_velocity.h5' = auto_figure_title('geo_velocity.h5', None, vars(inps))
'101020-110220_ECMWF_demErr_quadratic' = auto_figure_title('timeseries_ECMWF_demErr_quadratic.h5', '110220')
```

20.92.1.3 auto_flip_direction()

```
def pysar.view.auto_flip_direction (
    atr_dict )
```

Check flip left-right and up-down based on attribute dict, for radar-coded file only

20.92.1.4 auto_row_col_num()

```
def pysar.view.auto_row_col_num (
    subplot_num,
    data_shape,
    fig_size,
    fig_num = 1 )
```

Get optimal row and column number given figure size number of subplots

Inputs:

```
subplot_num : int, total number of subplots
data_shape  : list of 2 float, data size in pixel in row and column direction of each plot
fig_size    : list of 2 float, figure window size in inches
fig_num     : int, number of figure windows, optional, default = 1.
```

Outputs:

```
row_num : number of subplots in row    direction per figure
col_num : number of subplots in column direction per figure
```

20.92.1.5 check_colormap_input()

```
def pysar.view.check_colormap_input (
    atr_dict,
    colormap = None )
```

20.92.1.6 check_multilook_input()

```
def pysar.view.check_multilook_input (
    pixel_box,
    row_num,
    col_num )
```

20.92.1.7 cmdLineParse()

```
def pysar.view.cmdLineParse (
    argv )
```

20.92.1.8 get_epoch_full_list_from_input()

```
def pysar.view.get_epoch_full_list_from_input (
    all_epoch_list,
    epoch_input_list = [],
    epoch_num_input_list = [] )
```

Read/Get input epoch list from input epoch and epoch_num

20.92.1.9 main()

```
def pysar.view.main (
    argv )
```

Main Function #####.

20.92.1.10 plot_dem_lalo()

```
def pysar.view.plot_dem_lalo (
    bmap,
    dem,
    box,
    inps_dict )
```

Plot DEM in geo-coordinate

Inputs:

```
bmap : basemap object
dem   : dem data, 2D np.int16 matrix
box   : geo bounding box, 4-tuple as (urcrnrlon,urcrnrlat,llcrnrlon,llcrnrlat)
inps_dict : dict with the following 5 items:
    'disp_dem_shade' : bool, True/False
    'disp_dem_contour' : bool, True/False
    'dem_contour_step' : float, 200.0
    'dem_contour_smooth' : float, 3.0
```

Examples:

```
dem_disp_dict = {'dem': 'gsil0m_30m.dem', 'disp_dem_shade': True, 'disp_dem_contour': True,\
    'dem_contour_step': 200.0, 'dem_contour_smooth': 3.0}
bmap = plot_dem_lalo(bmap,dem,geo_box,dem_inps_dict)
```

20.92.1.11 plot_dem_yx()

```
def pysar.view.plot_dem_yx (
    ax,
    dem,
    inps_dict = dict() )
```

Plot DEM in radar coordinate

Inputs:

```
ax : matplotlib axes object
dem : dem data, 2D np.int16 matrix
inps_dict : dict with the following 5 items:
    'disp_dem_shade' : bool, True/False
    'disp_dem_contour' : bool, True/False
    'dem_contour_step' : float, 200.0
    'dem_contour_smooth' : float, 3.0
```

Examples:

```
dem_disp_dict = {'dem': 'gsil0m_30m.dem', 'disp_dem_shade': True, 'disp_dem_contour': True,\
    'dem_contour_step': 200.0, 'dem_contour_smooth': 3.0}
ax = plot_dem_yx(ax,dem,dem_disp_dict)
```

20.92.1.12 plot_matrix()

```
def pysar.view.plot_matrix (
    ax,
    data,
    meta_dict,
    inps = None )
```

Plot 2D matrix

Inputs:

```
ax      : matplotlib.pyplot axes object
data    : 2D np.array,
meta_dict : dictionary, attributes of data
inps    : Namespace, optional, input options for display
```

Outputs:

```
ax      : matplotlib.pyplot axes object
```

Example:

```
import matplotlib.pyplot as plt
import pysar._readfile as readfile
import pysar.view as view

data, atr = readfile.read('velocity.h5')
fig = plt.figure()
ax = fig.add_axes([0.1,0.1,0.8,0.8])
ax = view.plot_matrix(ax, data, atr)
plt.show()
```

20.92.1.13 round_to_1()

```
def pysar.view.round_to_1 (
    x )
```

Return the most significant digit of input number

20.92.1.14 scale_data2disp_unit()

```
def pysar.view.scale_data2disp_unit (
    matrix,
    atr_dict,
    disp_unit )
```

Scale data based on data unit and display unit

Inputs:

```
matrix    : 2D np.array
atr_dict  : dictionary, meta data
disp_unit : str, display unit
```

Outputs:

```
matrix    : 2D np.array, data after scaling
disp_unit : str, display unit
```

Default data file units in PySAR are: m, m/yr, radian, 1

20.92.1.15 scale_data4disp_unit_and_rewrap()

```
def pysar.view.scale_data4disp_unit_and_rewrap (
    data,
    atr,
    disp_unit = None,
    rewrapping = False )
```

Scale 2D matrix value according to display unit and re-wrapping flag

Disable rewrapping option 1) for specific data types, which rewrapping has no physical meaning;

2) if disp_unit exists and != 'radian'; priority: disp_unit > rewrapping

Inputs:

data - 2D np.array
atr - dict, including the following attributes:
UNIT
FILE_TYPE
WAVELENGTH
disp_unit - string, optional
rewrapping - bool, optional

Outputs:

data
disp_unit
rewrapping

20.92.1.16 update_matrix_with_plot_inps()

```
def pysar.view.update_matrix_with_plot_inps (
    data,
    meta_dict,
    inps )
```

20.92.1.17 update_plot_inps_with_display_setting_file()

```
def pysar.view.update_plot_inps_with_display_setting_file (
    inps,
    disp_set_file )
```

Update inps using values from display setting file

20.92.1.18 update_plot_inps_with_meta_dict()

```
def pysar.view.update_plot_inps_with_meta_dict (
    inps,
    meta_dict )
```

20.92.2 Variable Documentation

20.92.2.1 EXAMPLE

string EXAMPLE

Initial value:

```
1 = '''example:
2   view.py SanAndreas.dem
3   view.py velocity.h5 -u cm -m -2 -M 2 -c bwr --mask Mask_tempCoh.h5 -d SanAndreas.dem
4
5   view.py timeseries.h5
6   view.py unwrapIfgram.h5 070927-100217
7   view.py Wrapped.h5      -n 5
8   view.py geomap_4rlks.trans range
9
10  # Display in subset:
11  view.py velocity.h5 -x 100 600      -y 200 800
12  view.py velocity.h5 -l 31.05 31.10 -L 130.05 130.10
13
14  # Exclude Dates:
15  view.py timeseries.h5 -ex drop_date.txt
16
17  # Reference:
18  view.py velocity.h5 --ref-yx 210 566
19  view.py timeseries.h5 --ref-date 20101120
20
21  # Save and Output:
22  view.py velocity.h5 --save
23  view.py velocity.h5 -o velocity.pdf
24  view.py velocity.h5 --nodisplay
25 '''
```

20.92.2.2 mplColors

list mplColors

Initial value:

```
1 = ['#1f77b4',\
2     '#ff7f0e',\
3     '#2ca02c',\
4     '#d62728',\
5     '#9467bd',\
6     '#8c564b',\
7     '#e377c2',\
8     '#7f7f7f',\
9     '#bcbd22',\
10    '#17becf']
```

20.92.2.3 PLOT_TEMPLATE

string PLOT_TEMPLATE

Initial value:

```
1 = '''Plot Setting:
2   plot.name      = 'Yunjun et al., 2016, AGU, Fig 4f'
3   plot.type      = LOS_VELOCITY
4   plot.startDate =
5   plot.endDate   =
6   plot.displayUnit = cm/yr
7   plot.displayMin = -2
8   plot.displayMax = 2
9   plot.colormap   = jet
10  plot.subset.lalo = 33.05:33.15, 131.15:131.27
11  plot.seed.lalo  = 33.0651, 131.2076
12 '''
```

20.93 troposphere_uncertainty Namespace Reference

Functions

- def [cmdLineParse](#) ()
- def [velocity_uncertainty_vs_distance](#) (inps)
- def [statistics](#) (inps)
- def [estimate_seasonal](#) (inps)
- def [velocity_uncertainty](#) (relative_std_file, inps)
- def [download](#) (inps)
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

20.93.1 Function Documentation

20.93.1.1 cmdLineParse()

```
def troposphere_uncertainty.cmdLineParse ( )
```

20.93.1.2 download()

```
def troposphere_uncertainty.download (
    inps )
```

20.93.1.3 estimate_seasonal()

```
def troposphere_uncertainty.estimate_seasonal (
    inps )
```

20.93.1.4 main()

```
def troposphere_uncertainty.main (
    argv )
```

20.93.1.5 statistics()

```
def troposphere_uncertainty.statistics (
    inps )
```

20.93.1.6 velocity_uncertainty()

```
def troposphere_uncertainty.velocity_uncertainty (
    relative_std_file,
    inps )
```

20.93.1.7 velocity_uncertainty_vs_distance()

```
def troposphere_uncertainty.velocity_uncertainty_vs_distance (
    inps )
```

20.93.2 Variable Documentation

20.93.2.1 EXAMPLE

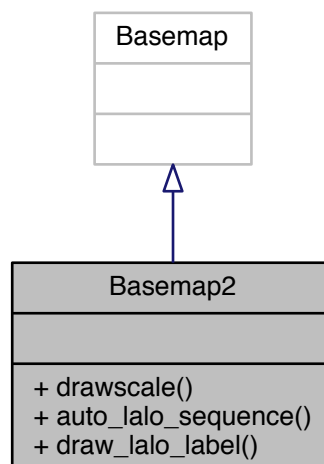
EXAMPLE

21 Class Documentation

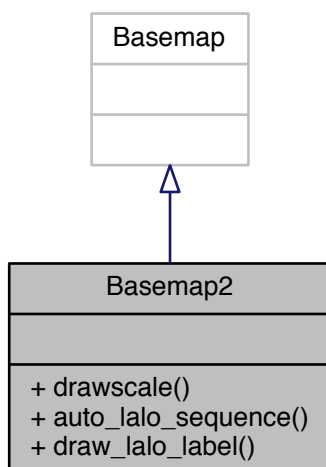
21.1 Basemap2 Class Reference

Class #####.

Inheritance diagram for Basemap2:



Collaboration diagram for Basemap2:



Public Member Functions

- def [drawscale](#) (self, lat_c, lon_c, distance, ax=None, font_size=12, yoffset=None, color='k')
- def [auto_lalo_sequence](#) (self, geo_box, lalo_step=None, max_tick_num=4, step_candidate=[1])
- def [draw_lalo_label](#) (self, geo_box, ax=None, lalo_step=None, labels=[1, font_size=12, color='k')

21.1.1 Detailed Description

Class #####.

21.1.2 Member Function Documentation

21.1.2.1 auto_lalo_sequence()

```

def auto_lalo_sequence (
    self,
    geo_box,
    lalo_step = None,
    max_tick_num = 4,
    step_candidate = [1 ]

```

Auto calculate lat/lon label sequence based on input `geo_box`

Inputs:

`geo_box` : 4-tuple of float, defining UL_lon, UL_lat, LR_lon, LR_lat coordinate
`max_tick_num` : int, rough major tick number along the longer axis
`step_candidate` : list of int, candidate list for the significant number of step

Outputs:

`lats/lons` : np.array of float, sequence of lat/lon auto calculated from input `geo_box`
`lalo_step` : float, lat/lon label step

Example:

```
geo_box = (128.0, 37.0, 138.0, 30.0)
lats, lons, step = m.auto_lalo_sequence(geo_box)
```

21.1.2.2 draw_lalo_label()

```
def draw_lalo_label (
    self,
    geo_box,
    ax = None,
    lalo_step = None,
    labels = [1,
    font_size = 12,
    color = 'k' )
```

Auto draw lat/lon label/tick based on coverage from `geo_box`

Inputs:

`geo_box` : 4-tuple of float, defining UL_lon, UL_lat, LR_lon, LR_lat coordinate
`labels` : list of 4 int, positions where the labels are drawn as in [left, right, top, bottom]
 default: [1,0,0,1]
`ax` : axes object the labels are drawn
`draw` : bool, do not draw if False

Outputs:

Example:

```
geo_box = (128.0, 37.0, 138.0, 30.0)
m.draw_lalo_label(geo_box)
```

21.1.2.3 drawscale()

```
def drawscale (
    self,
    lat_c,
    lon_c,
    distance,
    ax = None,
    font_size = 12,
    yoffset = None,
    color = 'k' )
```

draw a simple map scale from x1,y to x2,y in map projection coordinates, label it with actual distance

Inputs:

`lat_c/lon_c` : float, longitude and latitude of scale bar center, in degree
`distance` : float, distance of scale bar, in m
`yoffset` : float, optional, scale bar length at two ends, in degree

Example:

```
m.drawscale(33.06, 131.18, 2000)
```

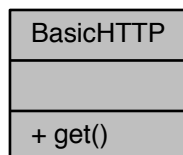
ref_link: <http://matplotliblib.1069221.n5.nabble.com/basemap-scalebar-td14133.html>

The documentation for this class was generated from the following file:

- [view.py](#)

21.2 BasicHTTP Class Reference

Collaboration diagram for BasicHTTP:



Static Public Member Functions

- def [get](#) (url)

21.2.1 Detailed Description

21.2.2 Member Function Documentation

21.2.2.1 get()

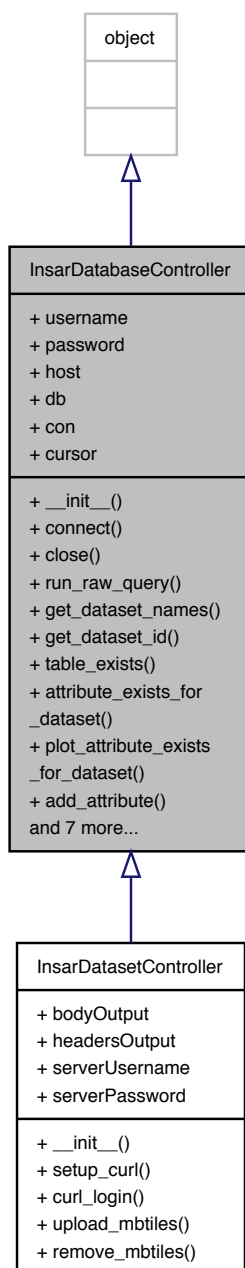
```
def get (
    url ) [static]
```

The documentation for this class was generated from the following file:

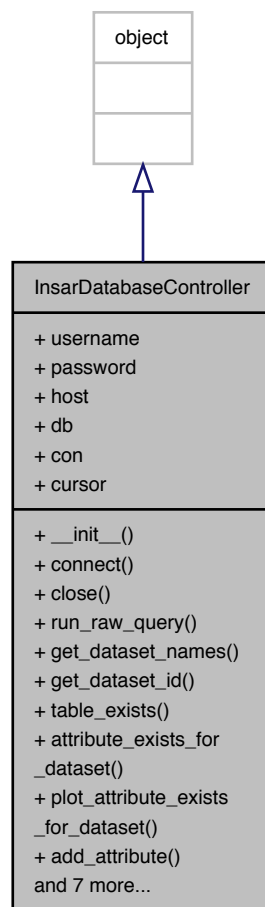
- [insarmaps_query.py](#)

21.3 InsarDatabaseController Class Reference

Inheritance diagram for InsarDatabaseController:



Collaboration diagram for InsarDatabaseController:



Public Member Functions

- `def __init__ (self, username, password, host, db)`
- `def connect (self)`
- `def close (self)`
- `def run_raw_query (self, query)`
- `def get_dataset_names (self)`
- `def get_dataset_id (self, dataset)`
- `def table_exists (self, table)`
- `def attribute_exists_for_dataset (self, dataset, attributekey)`
- `def plot_attribute_exists_for_dataset (self, dataset, attributekey)`
- `def add_attribute (self, dataset, attributekey, attributevalue)`
- `def add_plot_attribute (self, dataset, attributekey, plotAttributeJSON)`
- `def index_table_on (self, table, on, index_name)`
- `def cluster_table_using (self, table, index_name)`
- `def remove_point_table_if_there (self, table_name)`
- `def create_area_table_if_not_exists (self)`

- def [insert_dataset_into_area_table](#) (self, area, project_name, mid_long, mid_lat, country, region, chunk_num, attribute_keys, attribute_values, string_dates_sql, decimal_dates_sql)
- def [remove_dataset_if_there](#) (self, unavco_name)

Public Attributes

- [username](#)
- [password](#)
- [host](#)
- [db](#)
- [con](#)
- [cursor](#)

21.3.1 Detailed Description

21.3.2 Constructor & Destructor Documentation

21.3.2.1 `__init__()`

```
def __init__ (
    self,
    username,
    password,
    host,
    db )
```

21.3.3 Member Function Documentation

21.3.3.1 `add_attribute()`

```
def add_attribute (
    self,
    dataset,
    attributekey,
    attributevalue )
```

21.3.3.2 `add_plot_attribute()`

```
def add_plot_attribute (
    self,
    dataset,
    attributekey,
    plotAttributeJSON )
```

21.3.3.3 attribute_exists_for_dataset()

```
def attribute_exists_for_dataset (
    self,
    dataset,
    attributekey )
```

21.3.3.4 close()

```
def close (
    self )
```

21.3.3.5 cluster_table_using()

```
def cluster_table_using (
    self,
    table,
    index_name )
```

21.3.3.6 connect()

```
def connect (
    self )
```

21.3.3.7 create_area_table_if_not_exists()

```
def create_area_table_if_not_exists (
    self )
```

21.3.3.8 get_dataset_id()

```
def get_dataset_id (
    self,
    dataset )
```

21.3.3.9 get_dataset_names()

```
def get_dataset_names (
    self )
```

21.3.3.10 index_table_on()

```
def index_table_on (
    self,
    table,
    on,
    index_name )
```

21.3.3.11 insert_dataset_into_area_table()

```
def insert_dataset_into_area_table (
    self,
    area,
    project_name,
    mid_long,
    mid_lat,
    country,
    region,
    chunk_num,
    attribute_keys,
    attribute_values,
    string_dates_sql,
    decimal_dates_sql )
```

21.3.3.12 plot_attribute_exists_for_dataset()

```
def plot_attribute_exists_for_dataset (
    self,
    dataset,
    attributekey )
```

21.3.3.13 remove_dataset_if_there()

```
def remove_dataset_if_there (
    self,
    unavco_name )
```

21.3.3.14 remove_point_table_if_there()

```
def remove_point_table_if_there (
    self,
    table_name )
```

21.3.3.15 run_raw_query()

```
def run_raw_query (
    self,
    query )
```

21.3.3.16 table_exists()

```
def table_exists (
    self,
    table )
```

21.3.4 Member Data Documentation

21.3.4.1 con

con

21.3.4.2 cursor

cursor

21.3.4.3 db

db

21.3.4.4 host

host

21.3.4.5 password

password

21.3.4.6 username

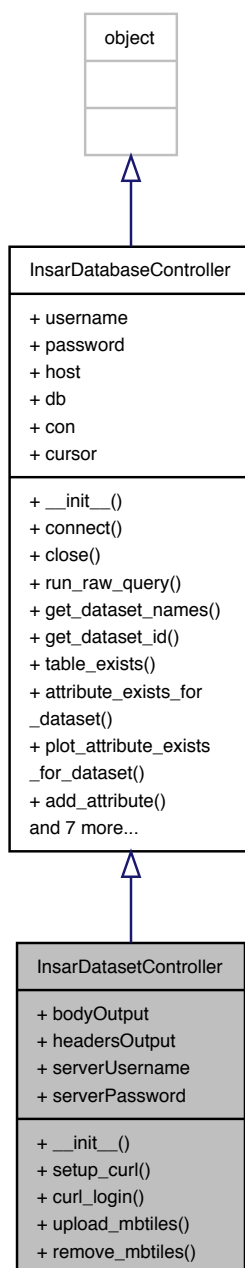
username

The documentation for this class was generated from the following file:

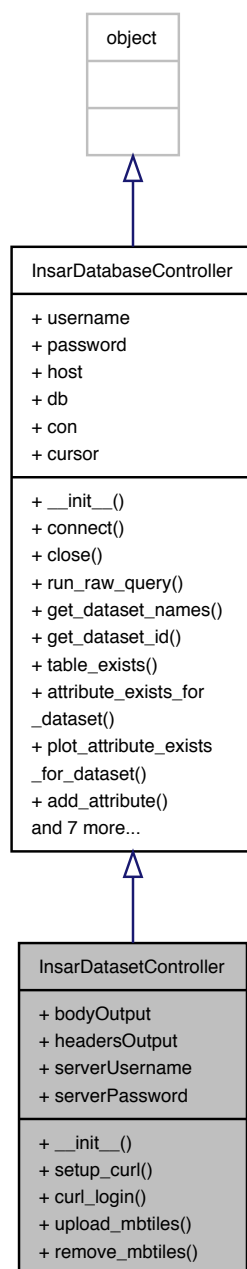
- [add_attribute_insarmaps.py](#)

21.4 InsarDatasetController Class Reference

Inheritance diagram for InsarDatasetController:



Collaboration diagram for InsarDatasetController:



Public Member Functions

- def `__init__` (self, `username`, `password`, `host`, `db`, `serverUsername`, `serverPassword`)
- def `setup_curl` (self)
- def `curl_login` (self, `username`, `password`)
- def `upload_mbtilles` (self, `fileName`)
- def `remove_mbtilles` (self, `fileName`)

Public Attributes

- [bodyOutput](#)
- [headersOutput](#)
- [serverUsername](#)
- [serverPassword](#)

21.4.1 Detailed Description

21.4.2 Constructor & Destructor Documentation

21.4.2.1 `__init__()`

```
def __init__ (
    self,
    username,
    password,
    host,
    db,
    serverUsername,
    serverPassword )
```

21.4.3 Member Function Documentation

21.4.3.1 `curl_login()`

```
def curl_login (
    self,
    username,
    password )
```

21.4.3.2 `remove_mbtiles()`

```
def remove_mbtiles (
    self,
    fileName )
```

21.4.3.3 `setup_curl()`

```
def setup_curl (
    self )
```


21.4.3.4 upload_mbtiles()

```
def upload_mbtiles (
    self,
    fileName )
```

21.4.4 Member Data Documentation

21.4.4.1 bodyOutput

bodyOutput

21.4.4.2 headersOutput

headersOutput

21.4.4.3 serverPassword

serverPassword

21.4.4.4 serverUsername

serverUsername

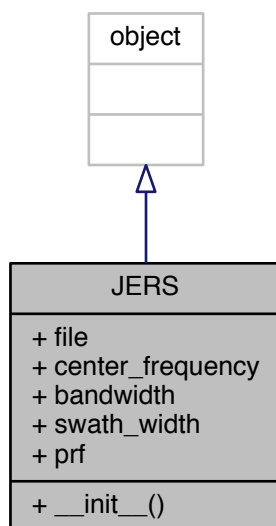
The documentation for this class was generated from the following file:

- [add_attribute_insarmaps.py](#)

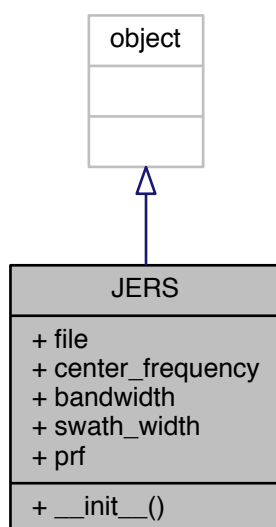
21.5 JERS Class Reference

Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang #.

Inheritance diagram for JERS:



Collaboration diagram for JERS:



Public Member Functions

- `def __init__ (self, file=None)`

Public Attributes

- `file`
- `center_frequency`
- `bandwidth`
- `swath_width`
- `prf`

21.5.1 Detailed Description

Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang #.

Recommended Usage: `import pysar._sensor` as `sensor`

21.5.2 Constructor & Destructor Documentation

21.5.2.1 `__init__()`

```
def __init__ (
    self,
    file = None )
```

21.5.3 Member Data Documentation

21.5.3.1 `bandwidth`

`bandwidth`

21.5.3.2 `center_frequency`

`center_frequency`

21.5.3.3 `file`

`file`

21.5.3.4 prf

prf

21.5.3.5 swath_width

swath_width

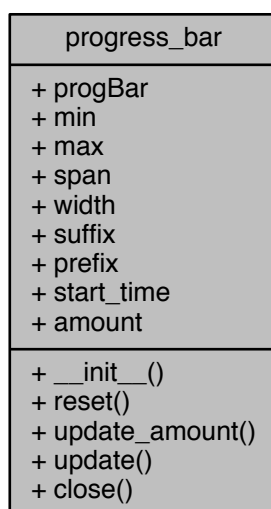
The documentation for this class was generated from the following file:

- [_sensor.py](#)

21.6 progress_bar Class Reference

Simple progress bar#####.

Collaboration diagram for progress_bar:



Public Member Functions

- def [__init__](#) (self, max**Value**=100, [prefix](#)="", min**Value**=0, total**Width**=60)
- def [reset](#) (self)
- def [update_amount](#) (self, new**Amount**=0, [suffix](#)="")
- def [update](#) (self, value, every=1, [suffix](#)="")
- def [close](#) (self)

Public Attributes

- [progBar](#)
- [min](#)
- [max](#)
- [span](#)
- [width](#)
- [suffix](#)
- [prefix](#)
- [start_time](#)
- [amount](#)

21.6.1 Detailed Description

Simple progress bar#####.

Creates a text-based progress bar. Call the object with the simple 'print' command to see the progress bar, which looks something like this:
 [=====> 22%]
 You may specify the progress bar's width, min and max values on init.

note:
 modified from PyAPS release 1.0 (<http://earthdef.caltech.edu/projects/pyaps/wiki/Main>)
 Code originally from <http://code.activestate.com/recipes/168639/>

```
example:
import pysar._datetime as ptime
date12_list = ptime.list_ifgram2date12(ifgram_list)
prog_bar = ptime.progress_bar(maxValue=1000, prefix='calculating:')
for i in range(1000):
    prog_bar.update(i+1, suffix=date)
    prog_bar.update(i+1, suffix=date12_list[i])
prog_bar.close()
```

21.6.2 Constructor & Destructor Documentation**21.6.2.1 __init__()**

```
def __init__ (
    self,
    maxValue = 100,
    prefix = '',
    minValue = 0,
    totalWidth = 60 )
```

21.6.3 Member Function Documentation

21.6.3.1 close()

```
def close (
    self )
```

Prints a blank space at the end to ensure proper printing of future statements.

21.6.3.2 reset()

```
def reset (
    self )
```

21.6.3.3 update()

```
def update (
    self,
    value,
    every = 1,
    suffix = '' )
```

Updates the amount, and writes to stdout. Prints a carriage return first, so it will overwrite the current line in stdout.

21.6.3.4 update_amount()

```
def update_amount (
    self,
    newAmount = 0,
    suffix = '' )
```

Update the progress bar with the new amount (with min and max values set at initialization; if it is over or under, it takes the min or max value as a default).

21.6.4 Member Data Documentation

21.6.4.1 amount

amount

21.6.4.2 max

max

21.6.4.3 min

min

21.6.4.4 prefix

prefix

21.6.4.5 progBar

progBar

21.6.4.6 span

span

21.6.4.7 start_time

start_time

21.6.4.8 suffix

suffix

21.6.4.9 width

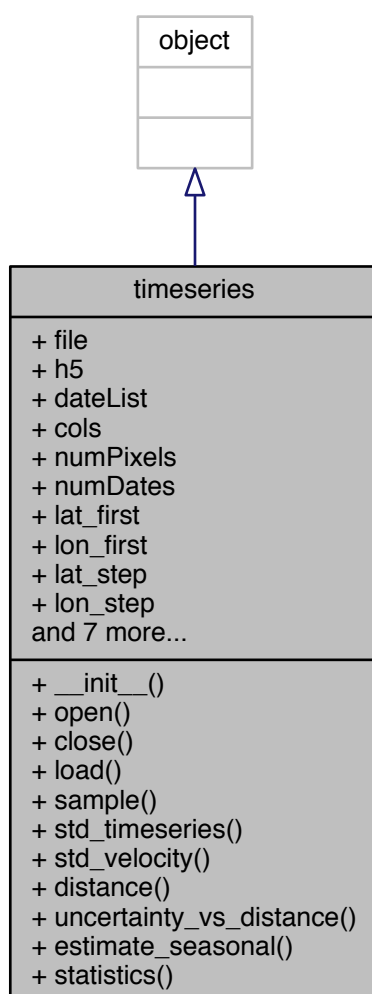
width

The documentation for this class was generated from the following file:

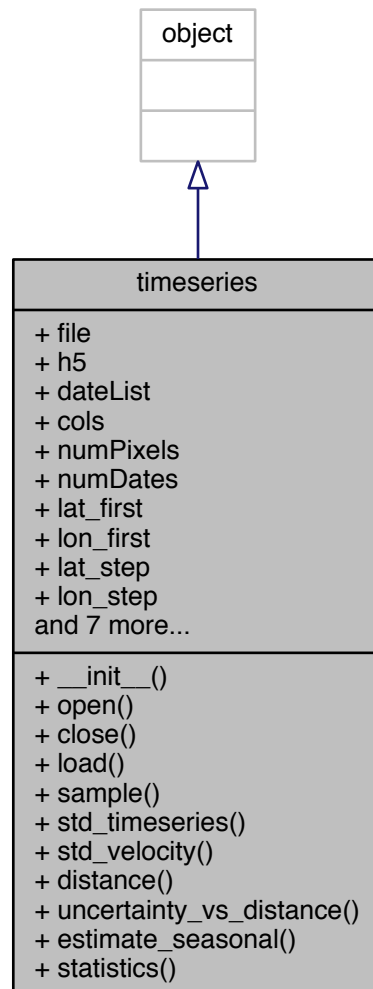
- [_datetime.py](#)

21.7 timeseries Class Reference

Inheritance diagram for timeseries:



Collaboration diagram for timeseries:



Public Member Functions

- `def __init__ (self, file=None)`
- `def open (self)`
- `def close (self)`
- `def load (self)`
- `def sample (self, numSamples=500, mask=None)`
- `def std_timeseries (self, ref)`
- `def std_velocity (self, sar_dates)`
- `def distance (self, i)`
- `def uncertainty_vs_distance (self, sar_dates)`
- `def estimate_seasonal (self, inps)`
- `def statistics (self, inps)`

Public Attributes

- [file](#)
- [h5](#)
- [dateList](#)
- [cols](#)
- [numPixels](#)
- [numDates](#)
- [lat_first](#)
- [lon_first](#)
- [lat_step](#)
- [lon_step](#)
- [lat](#)
- [lon](#)
- [Data](#)
- [idx](#)
- [relative_std](#)
- [relative_std_velocity](#)
- [dist](#)

21.7.1 Detailed Description

21.7.2 Constructor & Destructor Documentation

21.7.2.1 `__init__()`

```
def __init__ (
    self,
    file = None )
```

21.7.3 Member Function Documentation

21.7.3.1 `close()`

```
def close (
    self )
```

21.7.3.2 `distance()`

```
def distance (
    self,
    i )
```

21.7.3.3 estimate_seasonal()

```
def estimate_seasonal (
    self,
    inps )
```

21.7.3.4 load()

```
def load (
    self )
```

21.7.3.5 open()

```
def open (
    self )
```

21.7.3.6 sample()

```
def sample (
    self,
    numSamples = 500,
    mask = None )
```

21.7.3.7 statistics()

```
def statistics (
    self,
    inps )
```

21.7.3.8 std_timeseries()

```
def std_timeseries (
    self,
    ref )
```

21.7.3.9 std_velocity()

```
def std_velocity (
    self,
    sar_dates )
```

21.7.3.10 uncertainty_vs_distance()

```
def uncertainty_vs_distance (
    self,
    sar_dates )
```

21.7.4 Member Data Documentation**21.7.4.1 cols**

cols

21.7.4.2 Data

Data

21.7.4.3 dateList

dateList

21.7.4.4 dist

dist

21.7.4.5 file

file

21.7.4.6 h5

h5

21.7.4.7 idx

idx

21.7.4.8 lat

lat

21.7.4.9 lat_first

lat_first

21.7.4.10 lat_step

lat_step

21.7.4.11 lon

lon

21.7.4.12 lon_first

lon_first

21.7.4.13 lon_step

lon_step

21.7.4.14 numDates

numDates

21.7.4.15 numPixels

numPixels

21.7.4.16 relative_std

relative_std

21.7.4.17 `relative_std_velocity`

`relative_std_velocity`

The documentation for this class was generated from the following file:

- [delayTimeseries.py](#)

22 File Documentation

22.1 `__init__.py` File Reference

Namespaces

- [pysar](#)

Variables

- bool [miami_path](#) = True
- int [parallel_num](#) = 8
- float [figsize_single_min](#) = 6.0
- float [figsize_single_max](#) = 12.0
- list [figsize_multi](#) = [15.0, 8.0]

22.2 `_datetime.py` File Reference

Classes

- class [progress_bar](#)
Simple progress bar#####.

Namespaces

- [pysar._datetime](#)

Functions

- def [yyyymmdd2years](#) (dates)
- def [yymmdd2yyyymmdd](#) (date)
- def [yyyymmdd](#) (dates)
- def [yymmdd](#) (dates)
- def [ifgram_date_list](#) (ifgramFile, fmt='YYYYMMDD')
- def [read_date_list](#) (date_list_file)
- def [date_index](#) (dateList)
- def [date_list2tbase](#) (dateList)
- def [date_list2vector](#) (dateList)
- def [auto_adjust_xaxis_date](#) (ax, datevector, fontSize=12, every_year=1)
- def [list_ifgram2date12](#) (ifgram_list)
- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')

22.3 `_gmt.py` File Reference

Namespaces

- `pysar._gmt`

Functions

- def `write_gmt_simple` (lons, lats, z, fname, title='default', name='z', scale=1.0, offset=0, units='meters')

22.4 `_network.py` File Reference

Namespaces

- `pysar._network`

Functions

- def `read_pairs_list` (date12ListFile, dateList=[])
- def `write_pairs_list` (pairs, dateList, outName)
- def `read_igram_pairs` (igramFile)
- def `read_baseline_file` (baselineFile, exDateList=[])
- def `date12_list2index` (date12_list, date_list=[])
- def `get_date12_list` (File, check_drop_ifgram=False)
- def `igram_perp_baseline_list` (File)
- def `azimuth_bandwidth` (sensor)
- def `range_bandwidth` (sensor)
- def `wavelength` (sensor)
- def `incidence_angle` (sensor, inc_angle=None)
- def `signal2noise_ratio` (sensor)
- def `critical_perp_baseline` (sensor, inc_angle=None, print_msg=False)
- def `calculate_doppler_overlap` (dop_a, dop_b, bandwidth_az)
- def `simulate_coherence` (date12_list, baselineFile='bl_list.txt', sensor='Env', inc_angle=22.8, decor_↵
time=200.0, coh_resid=0.2, display=False)
- def `threshold_doppler_overlap` (date12_list, date_list, dop_list, bandwidth_az, dop_overlap_min=0.15)
- def `threshold_perp_baseline` (date12_list, date_list, pbase_list, pbase_max, pbase_min=0.0)
- def `threshold_temporal_baseline` (date12_list, btemp_max, keep_seasonal=True, btemp_min=0.0)
- def `coherence_matrix` (date12_list, coh_list, diagValue=np.nan)
- def `threshold_coherence_based_mst` (date12_list, coh_list)
- def `pair_sort` (pairs)
- def `pair_merge` (pairs1, pairs2)
- def `select_pairs_all` (date_list)
- def `select_pairs_sequential` (date_list, increment_num=2)
- def `select_pairs_hierarchical` (date_list, pbase_list, temp_perp_list)
- def `select_pairs_delaunay` (date_list, pbase_list, norm=True)
- def `select_pairs_mst` (date_list, pbase_list)
- def `select_pairs_star` (date_list, m_date=None, pbase_list=[])
- def `select_master_date` (date_list, pbase_list=[])
- def `select_master_interferogram` (date12_list, date_list, pbase_list, m_date=None)
- def `plot_network` (ax, date12_list, date_list, pbase_list, plot_dict={}, date12_list_drop=[], print_msg=True)
- def `plot_perp_baseline_hist` (ax, date8_list, pbase_list, plot_dict={}, date8_list_drop=[])
- def `plot_coherence_matrix` (ax, date12_list, coherence_list, date12_list_drop=[], plot_dict={})
- def `mode` (thelist)
- def `plot_coherence_history` (ax, date12_list, coherence_list, plot_dict={})
- def `auto_adjust_yaxis` (ax, dataList, fontSize=12, ymin=None, ymax=None)

Variables

- string [BASELINE_LIST_FILE](#)
- string [IFGRAM_LIST_FILE](#)

22.5 [_plot.py](#) File Reference

Namespaces

- [pysar._plot](#)

Functions

- def [plot_bar_std](#) (ax, date_list, std_list, fig_name=None, ref_date=None)

22.6 [_pysar_utilities.py](#) File Reference

Namespaces

- [pysar._pysar_utilities](#)

Functions

- def [touch](#) (fname_list, times=None)
- def [get_lookup_file](#) (filePattern=None, abspath=False, print_msg=True)
- def [get_geometry_file](#) (dset, coordType=None, filePattern=None, abspath=False, print_msg=True)
- def [check_loaded_dataset](#) (work_dir='.', inps=None, print_msg=True)
- def [is_file_exist](#) (file_list, abspath=True)
- def [four_corners](#) (atr)
- def [circle_index](#) (atr, circle_par)
- def [update_template_file](#) (template_file, extra_dict)
- def [get_residual_std](#) (timeseries_resid_file, mask_file='maskTempCoh.h5', ramp_type='quadratic')
- def [timeseries_std](#) (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def [get_residual_rms](#) (timeseries_resid_file, mask_file='maskTempCoh.h5', ramp_type='quadratic')
- def [timeseries_rms](#) (inFile, maskFile='maskTempCoh.h5', outFile=None, dimension=2)
- def [timeseries_coherence](#) (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def [normalize_timeseries](#) (ts_mat, nanValue=0)
- def [normalize_timeseries_old](#) (ts_mat, nanValue=0)
- def [update_file](#) (outFile, inFile=None, overwrite=False, check_readable=True)
- def [update_attribute_or_not](#) (atr_new, atr_orig, update=False)
- def [add_attribute](#) (File, atr_new=dict())
- def [check_parallel](#) (file_num=1, print_msg=True)
- def [perp_baseline_timeseries](#) (atr, dimension=1)
- def [range_distance](#) (atr, dimension=2)
- def [incidence_angle](#) (atr, dimension=2, print_msg=True)
- def [which](#) (program)
- def [check_drop_ifgram](#) (h5, print_msg=True)
- def [nonzero_mask](#) (File, outFile='mask.h5')
- def [spatial_average](#) (File, maskFile=None, box=None, saveList=False, checkAoi=True)
- def [temporal_average](#) (File, outFile=None)

- def [get_file_list](#) (fileList, abspath=False, coord=None)
- def [check_file_size](#) (fname_list, mode_width=None, mode_length=None)
- def [mode](#) (thelist)
- def [range_ground_resolution](#) (atr, print_msg=False)
- def [azimuth_ground_resolution](#) (atr)
- def [get_lookup_row_col](#) (y, x, lut_y, lut_x, y_factor=10, x_factor=10, geoCoord=False)
Use geomap.trans file for precious (pixel-level) coord conversion.*
- def [glob2radar](#) (lat, lon, lookupFile=None, atr_rdr=dict(), print_msg=True)
- def [radar2glob](#) (az, rg, lookupFile=None, atr_rdr=dict(), print_msg=True)
- def [check_variable_name](#) (path)
- def [hillshade](#) (data, scale)
- def [date_list](#) (h5file)
- def [design_matrix](#) (ifgramFile=None, date12_list=[], referenceDate=None, zero_first=True)
- def [timeseries_inversion_FGLS](#) (h5flat, h5timeseries)
- def [timeseries_inversion_L1](#) (h5flat, h5timeseries)
- def [perp_baseline_ifgram2timeseries](#) (ifgramFile, ifgram_list=[])
- def [dBh_dBv_timeseries](#) (ifgramFile)
- def [Bh_Bv_timeseries](#) (ifgramFile)
- def [get_file_stack](#) (File, maskFile=None)
- def [stacking](#) (File)
- def [yymmdd2YYYYMMDD](#) (date)
- def [yyyymmdd](#) (dates)
- def [yymmdd](#) (dates)
- def [make_triangle](#) (dates12, igram1, igram2, igram3)
- def [get_triangles](#) (h5file)
- def [generate_curls](#) (curlfile, h5file, Triangles, curls)

22.7 _readfile.py File Reference

Namespaces

- [pysar._readfile](#)

Functions

- def [read](#) (File, box=None, epoch=None, print_msg=True)
- def [read_attribute](#) (File, epoch=None)
- def [check_variable_name](#) (path)
- def [is_plot_attribute](#) (attribute)
- def [read_template](#) (File, delimiter='=')
- def [read_roipac_rsc](#) (File)
- def [read_gamma_par](#) (fname, delimiter=':', skiprows=3, convert2roipac=True)
- def [read_isce_xml](#) (File)
- def [attribute_gamma2roipac](#) (par_dict_in)
- def [attribute_isce2roipac](#) (metaDict, dates=[], baselineDict={})
- def [attribute_envi2roipac](#) (metaDict)
- def [read_float32](#) (File, box=None, byte_order='I')
- def [read_real_float64](#) (fname, box=None, byte_order='I')
- def [read_complex_float32](#) (fname, box=None, byte_order='I', cpx=False)
- def [read_real_float32](#) (fname, box=None, byte_order='I')
- def [read_complex_int16](#) (File, box=None, byte_order='I', cpx=False)
- def [read_real_int16](#) (File, box=None, byte_order='I')
- def [read_bool](#) (File, box=None)
- def [read_GPS_USGS](#) (File)
- def [read_multiple](#) (File, box='')

Variables

- list [multi_group_hdf5_file](#) = ['interferograms','coherence','wrapped','snaphu_connect_component']
- list [multi_dataset_hdf5_file](#) = ['timeseries','geometry']
- list [single_dataset_hdf5_file](#) = ['dem','mask','rmse','temporal_coherence', 'velocity']
- list [geometry_dataset](#)

22.8 [_remove_surface.py](#) File Reference

Namespaces

- [pysar._remove_surface](#)

Functions

- def [remove_data_surface](#) (data, mask, surf_type='plane')
- def [remove_data_multiple_surface](#) (data, mask, surf_type, ysub)
- def [remove_surface](#) (File, surf_type, maskFile=None, outFile=None, ysub=None)

22.9 [_sensor.py](#) File Reference

Classes

- class [JERS](#)

Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang #.

Namespaces

- [pysar._sensor](#)

22.10 [_Sidebar.md](#) File Reference22.11 [_variance.py](#) File Reference

Namespaces

- [pysar._variance](#)

Functions

- def [get_lat_lon](#) (atr)
- def [sample_data](#) (lat, lon, mask=None, num_sample=500)
- def [get_distance](#) (lat, lon, i)
- def [structure_function](#) (data, lat, lon, step=5e3, min_pair_num=100e3, print_msg=True)
- def [bin_variance](#) (distance, variance, step=5e3, min_pair_num=100e3, print_msg=True)

22.12 `_writefile.py` File Reference

Namespaces

- [pysar._writefile](#)

Functions

- def [write](#) (args)
- def [write_roipac_rsc](#) (atr, outname, sorting=True)
- def [write_float32](#) (args)
- def [write_complex64](#) (data, outname)
- def [write_real_int16](#) (data, outname)
- def [write_dem](#) (data, outname)
- def [write_real_float32](#) (data, outname)
- def [write_complex_int16](#) (data, outname)

22.13 `add.py` File Reference

Namespaces

- [pysar.add](#)

Functions

- def [add_matrix](#) (data1, data2)
- def [add_files](#) (fname_list, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- string [EXAMPLE](#)

22.14 `add_attribute.py` File Reference

Namespaces

- [pysar.add_attribute](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.15 add_attribute_insarmaps.py File Reference

Classes

- class [InsarDatabaseController](#)
- class [InsarDatasetController](#)

Namespaces

- [pysar.add_attribute_insarmaps](#)

Functions

- def [build_parser](#) ()
- def [main](#) (argv)

22.16 animation.py File Reference

Namespaces

- [animation](#)

Functions

- def [updatefig](#) (args)

Variables

- string [work_dir](#) = '/Users/yunjunz/insarlab/Galapagos/AlcedoEnvA2T061/PIC'
- list [fileList](#) = []
- list [titleList](#) = []
- list [imgs](#) = []
- [img](#) = mpimg.imread(fname)
- [fig](#) = plt.figure(figsize=[10, 5.4])
- [ax](#) = fig.add_axes([0.05, 0.05, 0.9, 0.8])
- int [i](#) = -2
- [im](#) = ax.imshow(imgs[i], animated=True)
- [ttl](#) = ax.text(200, -150, titleList[i], ha='left', fontsize=32)
- [ani](#) = animation.FuncAnimation(fig, updatefig, interval=1000, blit=True)
- [savefigDict](#) = dict()
- string [outName](#) = 'timeseries_animation.gif'
- [writer](#)
- [dpi](#)
- [savefig_kwargs](#)

22.17 asc_desc.py File Reference

Namespaces

- [pysar.asc_desc](#)

Functions

- def [get_overlap_lalo](#) (atr1, atr2)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [REFERENCE](#)
- [EXAMPLE](#)

22.18 Attributes.md File Reference

22.19 baseline_error.py File Reference

Namespaces

- [pysar.baseline_error](#)

Functions

- def [to_percent](#) (y, position)
- def [usage](#) ()
- def [main](#) (argv)

22.20 baseline_trop.py File Reference

Namespaces

- [pysar.baseline_trop](#)

Functions

- def [to_percent](#) (y, position)
- def [usage](#) ()
- def [main](#) (argv)

22.21 Bibliography.md File Reference

22.22 coord_glob2radar.py File Reference

Namespaces

- [pysar.coord_glob2radar](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.23 coord_radar2glob.py File Reference

Namespaces

- [pysar.coord_radar2glob](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.24 Coordinate.md File Reference

22.25 correct_dem.py File Reference

Namespaces

- [pysar.correct_dem](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.26 correlation_with_dem.py File Reference

Namespaces

- [pysar.correlation_with_dem](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.27 delayTimeseries.py File Reference

Classes

- class [timeseries](#)

Namespaces

- [delayTimeseries](#)

Functions

- def [write_to_h5](#) (dataset, outName, groupName, h5withAttributes)
- def [nearest_valid](#) (xr, yr, data_flat, rows, cols)

22.28 DEM.md File Reference

22.29 dem_error.py File Reference

Namespaces

- [pysar.dem_error](#)

Functions

- def [topographic_residual_inversion](#) (ts0, A0, inps)
- def [read_template2inps](#) (template_file, inps=None)
- def [check_exclude_date](#) (exDateIn, dateList)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)
- [REFERENCE](#)

22.30 diff.py File Reference

Namespaces

- [pysar.diff](#)

Functions

- def [diff_data](#) (data1, data2)
- def [diff_file](#) (file1, file2, outName=None, force=False)
- def [usage](#) ()
- def [cmdLineParse](#) ()
- def [main](#) (argv)

22.31 dloadUtil.py File Reference

Namespaces

- [dloadUtil](#)

Functions

- def [download_modis](#) (inps)
- def [download_atmosphereModel](#) (inps)
- def [daterange](#) (start_date, end_date)
- def [get_date](#) (f)
- def [pwv2zwd](#) (pwv)
- def [zwd2swd](#) (zwd, theta)
- def [read_modis](#) (file)

22.32 Documentation-Generation.md File Reference

22.33 Example.md File Reference

22.34 File-Descriptions.md File Reference

22.35 Gamma-File-Decription.md File Reference

22.36 gamma_view.py File Reference

Namespaces

- [pysar.gamma_view](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.37 generate_mask.py File Reference

Namespaces

- [pysar.generate_mask](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.38 geocode.py File Reference

Namespaces

- [pysar.geocode](#)

Functions

- def [geocode_output_filename](#) (fname)
- def [update_attribute_geo_lut](#) (atr_rdr, atr_lut, print_msg=True)
Geocoded with lut in geo coord #####
- def [geocode_file_geo_lut](#) (fname, lookup_file, fname_out, inps)
- def [interp_weights](#) (xy, uv, d=2)
Geocoded with lut in radar coord ##### Reference: <https://stackoverflow.com/questions/20915502/speedup-scipy-griddata-for-multiple-interpolations-between-two-irregular-grids>
- def [interpolate](#) (values, vtx, wts, fill_value=np.nan)
- def [update_attribute_radar_lut](#) (atr_rdr, inps, lat=None, lon=None, print_msg=True)
- def [geocode_file_radar_lut](#) (fname, lookup_file, fname_out=None, inps=None)
- def [geocode_file](#) (fname, lookup_file, fname_out, inps)
- def [read_template2inps](#) (template_file, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

22.39 get_modis_v3.py File Reference

Namespaces

- [get_modis_v3](#)

Functions

- def [usage](#) ()
- def [main](#) ()

Variables

- [out](#)
- [start_time_main](#)
- [time_elapsed](#)

22.40 Google-Earth.md File Reference

22.41 hdfsos5_2insarmaps.py File Reference

Namespaces

- [pysar.hdfsos5_2insarmaps](#)

Functions

- def [get_H5_filename](#) (path)
- def [build_parser](#) ()
- def [main](#) ()

22.42 hdfsos5_2json_mbtiles.py File Reference

Namespaces

- [pysar.hdfsos5_2json_mbtiles](#)

Functions

- def [get_date](#) (date_string)
- def [get_decimal_date](#) (d)
- def [region_name_from_project_name](#) (project_name)
- def [serialize_dictionary](#) (dictionary, fileName)
- def [convert_data](#) (attributes, decimal_dates, timeseries_datasets, dates, json_path, folder_name)
- def [make_json_file](#) (chunk_num, points, dates, json_path, folder_name)
- def [build_parser](#) ()
- def [main](#) ()

Variables

- [needed_attributes](#)

22.43 Home.md File Reference

22.44 ifgram_closure.py File Reference

Namespaces

- [pysar.ifgram_closure](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.45 ifgram_inversion.py File Reference

Namespaces

- [pysar.ifgram_inversion](#)

Functions

- def [phase_pdf_ds](#) (L, coherence=None, phiNum=1000)
- def [phase_variance_ds](#) (L, coherence=None)
- def [phase_variance_ps](#) (L, coherence=None)
- def [coherence2phase_variance_ds](#) (coherence, L=32, print_msg=False)
- def [coherence2fisher_info_index](#) (coherence, L=32, epsilon=1e-4)
- def [round_to_1](#) (x)
- def [ceil_to_1](#) (x)
- def [network_inversion_sbas](#) (B, ifgram, tbase_diff, skipZeroPhase=True)
- def [network_inversion_wls](#) (A, ifgram, weight, skipZeroPhase=True, Astd=None)
- def [temporal_coherence](#) (A, ts, ifgram, weight=None, chunk_size=500)
- def [ifgram_inversion_patch](#) (ifgramFile, coherenceFile, meta, box=None)
- def [ifgram_inversion](#) (ifgramFile='unwrapIfgram.h5', coherenceFile='coherence.h5', meta=None)
- def [write_timeseries_hdf5_file](#) (timeseries, date8_list, atr, timeseriesFile=None)
- def [read_template2inps](#) (template_file, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [TEMPLATE](#)
- [REFERENCE](#)

22.46 ifgram_reconstruction.py File Reference

Namespaces

- [pysar.ifgram_reconstruction](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.47 ifgram_simulation.py File Reference

Namespaces

- [pysar.ifgram_simulation](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.48 image_math.py File Reference

Namespaces

- [pysar.image_math](#)

Functions

- def [data_operation](#) (data, operator, operand)
- def [file_operation](#) (fname, operator, operand, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.49 incidence_angle.py File Reference

Namespaces

- [pysar.incidence_angle](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.50 info.py File Reference

Namespaces

- [pysar.info](#)

Functions

- def `print_attributes` (atr, sorting=True)
- def `print_hdf5_structure` (File)
By andrewcollette at <https://github.com/h5py/h5py/issues/406>.
- def `print_timseries_date_info` (dateList)
- def `usage` ()
- def `main` (argv)

22.51 insar_vs_gps.py File Reference

Namespaces

- `pysar.insar_vs_gps`

Functions

- def `readGPSfile` (gpsFile, gps_source)
- def `nearest` (x, tbase, xstep)
- def `find_row_column` (Lon, Lat, lon, lat, lon_step, lat_step)
- def `usage` ()
- def `main` (argv)

22.52 insarmaps_query.py File Reference

Classes

- class `BasicHTTP`

Namespaces

- `pysar.insarmaps_query`

Functions

- def `buildURL` (args)
- def `build_parser` ()
- def `main` ()

22.53 json_mbtiles2insarmaps.py File Reference

Namespaces

- `pysar.json_mbtiles2insarmaps`

Functions

- def `get_unavco_name` (json_path)
- def `upload_insarmaps_metadata` (fileName)
- def `upload_json` (folder_path)
- def `build_parser` ()
- def `main` ()

Variables

- `dbUsername`
- `dbPassword`
- `dbHost`

22.54 l1.py File Reference

Namespaces

- `pysar.l1`

Functions

- def `l1mosek` (P, q)
- def `l1mosek2` (P, q)
- def `l1` (P, q)
- def `l1blas` (P, q)

Variables

- `__MOSEK`
- `task`
- `x`

22.55 load_data.py File Reference

Namespaces

- `pysar.load_data`

Functions

- def `project_name2sensor` (projectName)
Sub Functions #####.
- def `auto_path_miami` (inps, template={})
- def `mode` (thelist)
- def `check_file_size` (fileList, mode_width=None, mode_length=None)
- def `check_existed_hdf5_file` (inFiles, hdf5File)
- def `load_multi_group_hdf5` (fileType, fileList, outfile='unwrapIgram.h5', exDict=dict())
- def `load_geometry_hdf5` (fileType, fileList, outfile=None, exDict=dict())
- def `load_single_dataset_hdf5` (file_type, infile, outfile=None, exDict=dict())
- def `copy_file` (targetFile, destDir)
- def `load_file` (fileList, inps_dict=dict(), outfile=None, file_type=None)
- def `load_data_from_template` (inps)
- def `cmdLineParse` ()
- def `main` (argv)
Main Function #####.

Variables

- [sensorList](#)
 - [EXAMPLE](#)
- Usage #####.*
- [TEMPLATE](#)

22.56 load_dem.py File Reference

Namespaces

- [pysar.load_dem](#)

Variables

- [demFile](#)
- [ext](#)
- [amp](#)
- [dem](#)
- [demRsc](#)
- [outName](#)
- [h5](#)
- [group](#)
- [dset](#)
- [data](#)
- [compression](#)

22.57 lod.py File Reference

Namespaces

- [pysar.lod](#)

Functions

- def [correct_lod_file](#) (File, rangeDistFile=None, outFile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [REFERENCE](#)
- [EXAMPLE](#)

22.58 look_angle.py File Reference

Namespaces

- [pysar.look_angle](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.59 los2enu.py File Reference

Namespaces

- [pysar.los2enu](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.60 mask.py File Reference

Namespaces

- [pysar.mask](#)

Functions

- def [mask_matrix](#) (data_mat, mask_mat, fill_value=None)
- def [update_mask](#) (mask, inps_dict, print_msg=True)
- def [mask_file](#) (File, maskFile, outFile=None, inps_dict=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.61 match.py File Reference

Namespaces

- [pysar.match](#)

Functions

- def [corners](#) (atr)
- def [nearest](#) (x, X)
- def [manual_offset_estimate](#) (matrix1, matrix2)
- def [match_two_files](#) (File1, File2, outName=None, manual_match=False, disp_fig=False)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.62 modify_network.py File Reference

Namespaces

- [pysar.modify_network](#)

Functions

- `def nearest_neighbor (x, y, x_array, y_array)`
Sub Function #####.
- `def reset_pairs (File)`
- `def manual_select_pairs_to_remove (File)`
- `def modify_file_date12_list (File, date12_to_rmv, mark_attribute=False, outFile=None)`
- `def read_template2inps (template_file, inps=None)`
- `def cmdLineParse ()`
- `def main (argv)`
Main Function #####.

Variables

- [EXAMPLE](#)
Usage #####.
- [TEMPLATE](#)

22.63 multi_transect.py File Reference

Namespaces

- [pysar.multi_transect](#)

Functions

- `def usage ()`
- `def dms2d (Coord)`
- `def gps_to_LOS (Ve, Vn, theta, heading)`
- `def check_st_in_box (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)`
- `def check_st_in_box2 (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)`
- `def line (x0, y0, x1, y1)`
- `def dist_point_from_line (m, c, x, y, dx, dy)`
- `def get_intersect (m, c, x, y)`
- `def readGPSfile (gpsFile, gps_source)`
- `def redGPSfile (gpsFile)`
- `def redGPSfile_cmm4 (gpsFile)`
- `def nearest (x, tbase, xstep)`
- `def find_row_column (Lon, Lat, lon, lat, lon_step, lat_step)`
- `def get_lat_lon (h5file)`
- `def nanmean (data, args)`
- `def nanstd (data, args)`
- `def get_transect (z, x0, y0, x1, y1)`
- `def get_start_end_point (Xf0, Yf0, Xf1, Yf1, L, dx, dy)`
- `def point_with_distance_from_line (Xf0, Yf0, Xf1, Yf1, L)`
- `def point_on_line_with_distance_from_beginning (Xf0, Yf0, Xf1, Yf1, L)`
- `def read_fault_coords (Fault_coord_file, Dp)`
- `def main (argv)`
- `def onclick (event)`

Variables

- lat
- lon
- lat_step
- lon_step
- lat_all
- lon_all
- Fault_lon
- Fault_lat
- Num_profiles
- FaultCoords
- Lat0
- Lon0
- Lat1
- Lon1
- Length
- Width
- Yf0
- Xf0
- Yf1
- Xf1
- y0
- x0
- y1
- x1
- fig
- ax
- xc
- yc
- cid
- length

*try: mf=float(Yf1-Yf0)/float((Xf1-Xf0)) # slope of the fault line cf=float(Yf0-mf*Xf0) # intercept of the fault line df0=dist_↔
_point_from_line(mf,cf,x0,y0,1,1) #distance of the profile start point from the Fault line df1=dist_point_from_↔
line(mf,cf,x1,y1,1,1) #distance of the profile end point from the Fault line*

- x
- y
- zi
- lat_transect
- lon_transect
- dx
- dy
- DX
- DY
- D
- mf
- cf
- df0_km
- transect
- XX0
- XX1
- YY0
- YY1
- m
- c

- [m1](#)
- [dp](#)
- [X0](#)
- [Y0](#)
- [X1](#)
- [Y1](#)
- [transect_lat](#)
- [transect_lon](#)
- [m_prof_edge](#)
- [c_prof_edge](#)
- [gpsFile](#)
- [insarData](#)
- [fileName](#)
- [fileExtension](#)
- [Stations](#)
- [Lat](#)
- [Lon](#)
- [Ve](#)
- [Se](#)
- [Vn](#)
- [Sn](#)
- [idxRef](#)
- [IDYref](#)
- [IDXref](#)
- [stationsList](#)
- [h5file_theta](#)
- [dset](#)
- [theta](#)
- [heading](#)
- [unitVec](#)
- [gpsLOS_ref](#)
- [GPS](#)
- [GPS_station](#)
- [GPSx](#)
- [GPSy](#)
- [GPS_lat](#)
- [GPS_lon](#)
- [idx](#)
- [IDY](#)
- [IDX](#)
- [gpsLOS](#)
- [NoInSAR](#)
- [DistGPS](#)
- [GPS_in_bound](#)
- [GPS_in_bound_st](#)
- [GPSxx](#)
- [GPSyy](#)
- [gx](#)
- [gy](#)
- [check_result](#)
- [check_result2](#)
- [dg](#)
- [axes](#)
- [nrows](#)
- [ms](#)

```
ax.fill_between(D/1000.0, (avglnSAR-stdlnSAR)*1000, (avglnSAR+stdlnSAR)*1000,where=(avglnSAR+stdlnSAR)*1000>=(avglnSAR-stdlnSAR)*1000,alpha=1, facecolor='Red')
```

- [avglnSAR](#)
- [axis](#)
- [stdlnSAR](#)
- [fig2](#)
- [axes2](#)
- [FaultLine](#)
- [figName](#)

Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minorLocator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)

- [mfc](#)
- [linewidth](#)
- [matFile](#)
- [dataset](#)
- [color](#)

```
ax.plot(D/1000.0, avglnSAR*1000, 'r-')
```

- [alpha](#)
- [fontsize](#)
- [lbound](#)

lower and higher bounds for displaying the profile

- [hbound](#)
- [ylim](#)
- [xlim](#)

22.64 multilook.py File Reference

Namespaces

- [pysar.multilook](#)

Functions

- def [multilook_matrix](#) (matrix, lks_y, lks_x)
Sub Functions #####
- def [multilook_attribute](#) (atr_dict, lks_y, lks_x, print_msg=True)
- def [multilook_file](#) (infile, lks_y, lks_x, outfile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.65 perp_baseline.py File Reference

Namespaces

- [pysar.perp_baseline](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.66 plot_network.py File Reference

Namespaces

- [pysar.plot_network](#)

Functions

- def [read_template2inps](#) (template_file, inps=None)
Sub Function #####.
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [BL_LIST](#)
Sub Function #####.
- [DATE12_LIST](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

22.67 plot_tropcor_phase_elevation.py File Reference

Namespaces

- [plot_tropcor_phase_elevation](#)

Variables

- [workDir](#)
- [demFile](#)
- [timeseriesFile](#)
- [timeseriesFile2](#)
- [maskFile](#)
- [tropHgtFile](#)
- [ecmwfFile](#)
- [epoch](#)
- [dem](#)
- [dem_atr](#)
- [data](#)
- [atr](#)
- [data2](#)
- [atr2](#)

- [tropHgt](#)
- [atr3](#)
- [ecmwf](#)
- [atr4](#)
- [mask](#)
- [msk_atr](#)
- [ndx](#)
- [dataList](#)
- [fig](#)
- [axes](#)
- [nrows](#)
- [ncols](#)
- [sharex](#)
- [True](#)
- [sharey](#)
- [figsize](#)
- [i](#)
- [ms](#)
- [bbox_inches](#)
- [dpi](#)

22.68 prep4timeseries.py File Reference

Namespaces

- [pysar.prep4timeseries](#)

Functions

- def [createParser](#) ()
- def [cmdLineParse](#) (iargs=None)
- def [extractIsceMetadata](#) (xmlFile)
- def [read_baseline](#) (baselineFile)
- def [baselineTimeseries](#) (baselineDir)
- def [read_rsc](#) (rscFile)
- def [write_rsc](#) (rscDict, rscFile)
- def [attribute_isce2roipac](#) (metaDict, dates=[], baselineDict={})
- def [prepare_stack](#) (inputDir, filePattern, metaDictIn, baselineDict)
- def [prepare_geometry](#) (geometryDir, exDict=None)
- def [read_template](#) (File, delimiter='=')
- *from _read_file.py Need to be removed once we can import _readfile.py*
- def [check_variable_name](#) (path)
- def [main](#) (iargs=None)

Variables

- [GDAL2NUMPY_DATATYPE](#)
- [EXAMPLE](#)

22.69 prep_gamma.py File Reference

Namespaces

- [pysar.prep_gamma](#)

Functions

- def [get_perp_baseline](#) (m_par_file, s_par_file, off_file, atr_dict={})
 Sub Functions #####
- def [get_lalo_ref](#) (m_par_file, atr_dict={})
- def [extract_attribute_interferogram](#) (fname)
- def [extract_attribute_lookup_table](#) (fname)
- def [extract_attribute_dem_geo](#) (fname)
- def [extract_attribute_dem_radar](#) (fname)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

22.70 prep_giant_ifg_list.py File Reference

Namespaces

- [pysar.prep_giant_ifg_list](#)

Functions

- def [get_mission_name](#) (meta_dict)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.71 prep_isce.py File Reference

Namespaces

- [pysar.prep_isce](#)

Functions

- def [createParser](#) ()
- def [cmdLineParse](#) (iargs=None)
- def [extractIsceMetadata](#) (xmlFile)
- def [write_rsc](#) (isceFile, dates, metadata, baselineDict)
- def [prepare_stack](#) (inputDir, filePattern, metadata, baselineDict)
- def [read_baseline](#) (baselineFile)
- def [baselineTimeseries](#) (baselineDir)
- def [prepare_geometry](#) (geometryDir)
- def [main](#) (iargs=None)

Variables

- [GDAL2NUMPY_DATATYPE](#)

22.72 prep_roipac.py File Reference

Namespaces

- [pysar.prep_roipac](#)

Functions

- def [extract_attribute](#) (fname)
Sub Functions #####
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

22.73 pysarApp.py File Reference

Namespaces

- [pysar.pysarApp](#)

Functions

- def [check_geocode_file](#) (lookupFile, File, templateFile=None, outFile=None)
- def [check_subset_file](#) (File, inps_dict, outFile=None, overwrite=False)
- def [subset_dataset](#) (inps, template_file)
- def [multilook_dataset](#) (inps, lks_y=None, lks_x=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [LOGO](#)
- [TEMPLATE](#)
- [EXAMPLE](#)
- [UM_FILE_STRUCT](#)

22.74 quality_map.py File Reference

Namespaces

- [pysar.quality_map](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.75 range_distance.py File Reference

Namespaces

- [pysar.range_distance](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.76 README.md File Reference

22.77 reference_epoch.py File Reference

Namespaces

- [pysar.reference_epoch](#)

Functions

- def [ref_date_attribute](#) (atr_in, ref_date, date_list)
- def [ref_date_file](#) (inFile, ref_date, outFile=None)
- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

22.78 remove_plane.py File Reference

Namespaces

- [pysar.remove_plane](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.79 rewrap.py File Reference

Namespaces

- [pysar.rewrap](#)

Functions

- def [usage](#) ()
- def [rewrap](#) (unw, cycle=2 *np.pi)
- def [main](#) (argv)

22.80 SAR-Sensor-Parameter.md File Reference

22.81 save_gmt.py File Reference

Namespaces

- [pysar.save_gmt](#)

Functions

- def [get_geo_lat_lon](#) (atr)
- def [write_grd_file](#) (data, atr, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.82 save_hdfeos5.py File Reference

Namespaces

- [pysar.save_hdfeos5](#)

Functions

- def [get_mission_name](#) (meta_dict)
- def [metadata_pysar2unavco](#) (pysar_meta_dict, dateList)
- def [get_hdfeos5_filename](#) (timeseriesFile)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [BOOL_ZERO](#)
- [INT_ZERO](#)
- [FLOAT_ZERO](#)
- [CPX_ZERO](#)
- [TEMPALTE](#)
- [EXAMPLE](#)

22.83 save_kml.py File Reference

Namespaces

- [pysar.save_kml](#)

Functions

- def [write_kmz_file](#) (data, atr, out_name_base, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.84 save_mat.py File Reference

Namespaces

- [pysar.save_mat](#)

Functions

- def [usage](#) ()
- def [yyyymmdd2years](#) (date)
- def [main](#) (argv)

22.85 [save_roipac.py](#) File Reference

Namespaces

- [pysar.save_roipac](#)

Functions

- def [usage](#) ()
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.86 [seed_data.py](#) File Reference

Namespaces

- [pysar.seed_data](#)

Functions

- def [nearest](#) (x, tbase, xstep)
Sub Functions #####.
- def [seed_file_reference_value](#) (File, outName, refList, ref_y=", ref_x=")
- def [seed_file_inps](#) (File, inps=None, outFile=None)
- def [seed_attributes](#) (atr_in, x, y)
- def [manual_select_reference_yx](#) (stack, inps)
- def [select_max_coherence_yx](#) (cohFile, mask=None, min_coh=0.85)
- def [random_select_reference_yx](#) (data_mat, print_msg=True)
- def [print_warning](#) (next_method)
- def [read_seed_template2inps](#) (template_file, inps=None)
- def [read_seed_reference2inps](#) (reference_file, inps=None)
- def [remove_reference_pixel](#) (File)
- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [TEMPLATE](#)
Usage #####.
- [NOTE](#)
- [EXAMPLE](#)

22.87 select_network.py File Reference

Namespaces

- [pysar.select_network](#)

Functions

- def [log](#) (msg)
- def [project_name2sensor](#) (projectName)
- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [sar_sensor_list](#)
- [REFERENCE](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

22.88 spatial_average.py File Reference

Namespaces

- [pysar.spatial_average](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [EXAMPLE](#)
Usage #####.

22.89 spatial_filter.py File Reference

Namespaces

- [pysar.spatial_filter](#)

Functions

- def [filter_data](#) (data, filter_type, filter_par=None)
- def [filter_file](#) (fname, filter_type, filter_par=None, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.90 stacking.py File Reference

Namespaces

- [pysar.stacking](#)

Functions

- def [cmdLineParse](#) ()
 - def [main](#) (argv)
- Main Function #####.*

Variables

- [EXAMPLE](#)
- Usage #####.*

22.91 subset.py File Reference

Namespaces

- [pysar.subset](#)

Functions

- def [coord_geo2radar](#) (geoCoordIn, atr, coordType)
Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')
- def [coord_radar2geo](#) (radarCoordIn, atr, coordType)
Inputs: radarCoord : coordinate (list) in row/col in int atr : dictionary of file attributes coordType : coordinate type: row, col, y, x.
- def [check_box_within_data_coverage](#) (pixel_box, atr_dict)
- def [subset_attribute](#) (atr_dict, subset_box, print_msg=True)
- def [get_coverage_box](#) (atr)
- def [read_subset_template2box](#) (templateFile)
- def [bbox_geo2radar](#) (geo_box, atr_rdr=dict(), lookupFile=None)
- def [bbox_radar2geo](#) (pix_box, atr_rdr=dict(), lookupFile=None)
- def [subset_box2inps](#) (inps, pix_box, geo_box)
- def [get_box_overlap_index](#) (box1, box2)
- def [subset_input_dict2box](#) (subset_dict, meta_dict)
- def [box_pixel2geo](#) (pixel_box, meta_dict)
- def [box_geo2pixel](#) (geo_box, meta_dict)
- def [subset_file](#) (File, subset_dict_input, outFile=None)
- def [subset_file_list](#) (fileList, inps)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.92 sum_epochs.py File Reference

Namespaces

- [pysar.sum_epochs](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.93 temporal_average.py File Reference

Namespaces

- [pysar.temporal_average](#)

Functions

- def [usage](#) ()
Usage #####
- def [main](#) (argv)
Main Function #####

22.94 temporal_coherence.py File Reference

Namespaces

- [pysar.temporal_coherence](#)

Functions

- def [temporal_coherence](#) (timeseriesFile, ifgramFile)
- def [usage](#) ()
- def [main](#) (argv)

Variables

- [USAGE](#)
- [DESCRIPTION](#)
- [REFERENCE](#)
- [EXAMPLE](#)

22.95 temporal_derivative.py File Reference

Namespaces

- [pysar.temporal_derivative](#)

Functions

- def [usage](#) ()
- def [main](#) (argv)

22.96 temporal_filter.py File Reference

Namespaces

- [pysar.temporal_filter](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.97 timeseries2velocity.py File Reference

Namespaces

- [pysar.timeseries2velocity](#)

Functions

- def [get_exclude_date](#) (inps, date_list_all)
- def [get_velocity_filename](#) (timeseries_file, template_file=None, vel_file='velocity.h5', inps=None)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [TEMPLATE](#)
- [DROP_DATE_TXT](#)

22.98 timeseries_rms.py File Reference

Namespaces

- [pysar.timeseries_rms](#)

Functions

- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [TEMPLATE](#)
- [EXAMPLE](#)

22.99 transect.py File Reference

Namespaces

- [pysar.transect](#)

Functions

- def [get_scale_from_disp_unit](#) (disp_unit, data_unit)
 - def [read_lonlat_file](#) (lonlat_file)
 - def [manual_select_start_end_point](#) (File)
 - def [transect_yx](#) (z, atr, start_yx, end_yx, interpolation='nearest')
 - def [transect_lalo](#) (z, atr, start_lalo, end_lalo, interpolation='nearest')
 - def [transect_list](#) (fileList, inps)
 - def [cmdLineParse](#) ()
 - def [main](#) (argv)
- Main #####*

Variables

- [EXAMPLE](#)

22.100 transect_legacy.py File Reference

Namespaces

- [pysar.transect_legacy](#)

Functions

- def [dms2d](#) (Coord)
- def [gps_to_LOS](#) (Ve, Vn, theta, heading)
- def [check_st_in_box](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [check_st_in_box2](#) (x, y, x0, y0, x1, y1, X0, Y0, X1, Y1)
- def [line](#) (x0, y0, x1, y1)
- def [dist_point_from_line](#) (m, c, x, y, dx, dy)
- def [get_intersect](#) (m, c, x, y)
- def [readGPSfile](#) (gpsFile, gps_source)
- def [redGPSfile](#) (gpsFile)
- def [redGPSfile_cmm4](#) (gpsFile)
- def [nearest](#) (x, tbase, xstep)
- def [find_row_column](#) (Lon, Lat, lon, lat, lon_step, lat_step)
- def [get_lat_lon](#) (atr)
- def [nanmean](#) (data, args)
- def [nanstd](#) (data, args)
- def [get_transect](#) (z, x0, y0, x1, y1, interpolation='nearest')

Option: interpolation : sampling/interpolation method, including: 'nearest' - nearest neighbour, by default 'cubic' - cubic interpolation 'bilinear' - bilinear interpolation.

- def [Usage](#) ()
- def [main](#) (argv)
- def [onclick](#) (event)

Variables

- [fig](#)
- [ax](#)
- [xc](#)
- [yc](#)
- [cid](#)
- [x0](#)
- [x1](#)
- [y0](#)
- [y1](#)
- [mf](#)
- [cf](#)
- [df0](#)
- [df1](#)
- [mp](#)
- [Info_aboutFault](#)
- [length](#)
- [x](#)
- [y](#)
- [zi](#)
- [lat_transect](#)
- [lon_transect](#)
- [earth_radius](#)
- [dx](#)
- [dy](#)
- [DX](#)
- [DY](#)
- [D](#)
- [df0_km](#)
- [transect](#)
- [XX0](#)
- [XX1](#)
- [YY0](#)
- [YY1](#)
- [m](#)
- [c](#)
- [m1](#)
- [X0](#)
- [Y0](#)
- [X1](#)
- [Y1](#)
- [transect_lat](#)
- [transect_lon](#)
- [m_prof_edge](#)
- [c_prof_edge](#)
- [gpsFile](#)
- [insarData](#)
- [fileName](#)
- [fileExtension](#)
- [Stations](#)
- [Lat](#)
- [Lon](#)
- [Ve](#)
- [Se](#)

- Vn
- Sn
- idxRef
- Length
- Width
- lat
- lon
- lat_step
- lon_step
- lat_all
- lon_all
- IDYref
- IDXref
- stationsList
- h5file__theta
- dset
- theta
- heading
- unitVec
- gpsLOS_ref
- GPS
- GPS_station
- GPSx
- GPSy
- GPS_lat
- GPS_lon
- idx
- IDY
- IDX
- gpsLOS
- NoInSAR
- DistGPS
- GPS_in_bound
- GPS_in_bound_st
- GPSxx
- GPSyy
- gx
- gy
- check_result
- check_result2
- dg
- axes
- nrows
- ms
- *ax.fill_between(D/1000.0, (avgInSAR-stdInSAR)*1000, (avgInSAR+stdInSAR)*1000,where=(avgInSAR+stdInSAR)*1000>=(avgInSAR-stdInSAR)*1000,alpha=1, facecolor='Red')*
- avgInSAR
- axis
- stdInSAR
- fig2
- axes2
- FaultLine
- figName

*Temporary To plot DEM try: majorLocator = MultipleLocator(5) ax.yaxis.set_major_locator(majorLocator) minor←
Locator = MultipleLocator(1) ax.yaxis.set_minor_locator(minorLocator)*

- [mfc](#)
- [linewidth](#)
- [matFile](#)
- [dataset](#)
- [color](#)

*ax.plot(D/1000.0, avgInSAR*1000, 'r-')*

- [alpha](#)
- [fontsize](#)
- [lbound](#)

lower and higher bounds for displaying the profile

- [hbound](#)
- [fault_loc](#)
- [ylim](#)

22.101 tropcor_phase_elevation.py File Reference

Namespaces

- [pysar.tropcor_phase_elevation](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [REFERENCE](#)

22.102 tropcor_pyaps.py File Reference

Namespaces

- [pysar.tropcor_pyaps](#)

Functions

- def [get_delay](#) (grib_file, atr, inps_dict)
- def [date_list2grib_file](#) (date_list, hour, grib_source, grib_dir)
- def [dload_grib](#) (date_list, hour, grib_source='ECMWF', weather_dir='.')
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [REFERENCE](#)
- [TEMPLATE](#)
- [DATA_INFO](#)

22.103 troposphere_uncertainty.py File Reference

Namespaces

- [troposphere_uncertainty](#)

Functions

- def [cmdLineParse](#) ()
- def [velocity_uncertainty_vs_distance](#) (inps)
- def [statistics](#) (inps)
- def [estimate_seasonal](#) (inps)
- def [velocity_uncertainty](#) (realtime_std_file, inps)
- def [download](#) (inps)
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

22.104 tsviewer.py File Reference

Namespaces

- [pysar.tsviewer](#)

Functions

- def [read_timeseries_yx](#) (timeseries_file, y, x, ref_yx=None)
- def [read_timeseries_lalo](#) (timeseries_file, lat, lon)
- def [cmdLineParse](#) ()
- def [format_coord](#) (x, y)
- def [time_slider_update](#) (val)
- def [plot_timeseries_errorbar](#) (ax, dis_ts, inps)
- def [plot_timeseries_scatter](#) (ax, dis_ts, inps)
- def [update_timeseries](#) (y, x)
- def [plot_timeseries_event](#) (event)

Variables

- [EXAMPLE](#)
- [inps](#)

Actual code.

- [atr](#)
- [k](#)
- [h5](#)
- [dateList](#)
- [date_num](#)
- [dates](#)
- [tims](#)
- [input_ex_date](#)
- [ex_date_list](#)
- [ex_date](#)
- [ex_dates](#)
- [ex_idx_list](#)
- [zero_idx](#)

Zero displacement for 1st acquisition.

- [length](#)
- [width](#)
- [ullon](#)
- [ullat](#)
- [lon_step](#)
- [lat_step](#)
- [lrlon](#)
- [lrlat](#)
- [y](#)
- [x](#)
- [yx](#)
- [ref_yx](#)
- [unit_fac](#)
- [flip_ud](#)
- [left_lr](#)
- [file_list](#)
- [mask_file](#)
- [mask](#)
- [epoch](#)
- [d_v](#)
- [timeseries_file](#)
- [ref_d_v](#)
- [data_lim](#)
- [ylim_mat](#)
- [fig_v](#)

Fig 1 - Cumulative Displacement Map.

- [ax_v](#)
- [dem](#)
- [dem_file](#)
- [img](#)
- [cmap](#)
- [colormap](#)
- [clim](#)
- [interpolation](#)
- [ms](#)

- [markeredgecolor](#)
- [format_coord](#)
- [cbar](#)
- [orientation](#)
- [ax_time](#)
- [axisbg](#)
- [yticks](#)
- [tslider](#)
- [valinit](#)
- [facecolor](#)
- [ecolor](#)
- [fig_ts](#)

Fig 2 - Time Series Displacement - Point.

- [figsize](#)
- [ax_ts](#)
- [error_ts](#)
- [error_fileContent](#)
- [error_file](#)
- [dtype](#)
- [e_ts](#)
- [ex_error_ts](#)
- [d_ts](#)
- [fig_base](#)

Output.

- [outName](#) = inps.fig_base+'_ts.pdf'
- [header_info](#)
- [lat](#)
- [lon](#)
- [fmt](#)
- string [delimiter](#) = header_info)
- [bbox_inches](#)
- [transparent](#)
- [True](#)
- [dpi](#)
- [cid](#) = fig_v.canvas.mpl_connect('button_press_event', plot_timeseries_event)

Final linking of the canvas to the plots.

22.105 UNAVCO-InSAR-Archive.md File Reference

22.106 unwrap_error.py File Reference

Namespaces

- [pysar.unwrap_error](#)

Functions

- def [bridging_data](#) (data, mask, x, y)
- def [unwrap_error_correction_phase_closure](#) (ifgram_file, mask_file, ifgram_cor_file=None)
- def [unwrap_error_correction_bridging](#) (ifgram_file, mask_file, y_list, x_list, ramp_type='plane', ifgram_cor_↵
file=None, save_cor_deramp_file=False)
- def [read_template2inps](#) (template_file, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- string [EXAMPLE](#)
- string [TEMPLATE](#)
- string [REFERENCE](#)
- string [DESCRIPTION](#)

22.107 view.py File Reference

Classes

- class [Basemap2](#)
Class #####.

Namespaces

- [pysar.view](#)

Functions

- def [round_to_1](#) (x)
- def [add_inner_title](#) (ax, title, loc, size=None, kwargs)
- def [auto_flip_direction](#) (atr_dict)
- def [auto_figure_title](#) (fname, epoch=[], inps_dict=None)
- def [auto_row_col_num](#) (subplot_num, data_shape, fig_size, fig_num=1)
- def [check_colormap_input](#) (atr_dict, colormap=None)
- def [check_multilook_input](#) (pixel_box, row_num, col_num)
- def [get_epoch_full_list_from_input](#) (all_epoch_list, epoch_input_list=[], epoch_num_input_list=[])
- def [plot_dem_lalo](#) (bmap, dem, box, inps_dict)
- def [plot_dem_yx](#) (ax, dem, inps_dict=dict())
- def [scale_data4disp_unit_and_rewrap](#) (data, atr, disp_unit=None, rewrapping=False)
- def [scale_data2disp_unit](#) (matrix, atr_dict, disp_unit)
- def [update_plot_inps_with_display_setting_file](#) (inps, disp_set_file)
- def [update_plot_inps_with_meta_dict](#) (inps, meta_dict)
- def [update_matrix_with_plot_inps](#) (data, meta_dict, inps)
- def [plot_matrix](#) (ax, data, meta_dict, inps=None)
- def [cmdLineParse](#) (argv)
- def [main](#) (argv)
Main Function #####.

Variables

- list [mplColors](#)
- string [EXAMPLE](#)
- string [PLOT_TEMPLATE](#)

22.108 Web-Viewer.md File Reference

Index

- [_Sidebar.md](#), 264
- [_MOSEK](#)
 - [pysar::l1](#), 105
- [__init__](#)
 - [delayTimeseries::timeseries](#), 256
 - [pysar::_datetime::progress_bar](#), 251
 - [pysar::_sensor::JERS](#), 249
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), 240
 - [pysar::add_attribute_insarmaps::InsarDataset](#)↔
[Controller](#), 246
- [__init__.py](#), 260
- [_datetime.py](#), 260
- [_gmt.py](#), 261
- [_network.py](#), 261
- [_plot.py](#), 262
- [_pysar_utilities.py](#), 262
- [_readfile.py](#), 263
- [_remove_surface.py](#), 264
- [_sensor.py](#), 264
- [_variance.py](#), 264
- [_writefile.py](#), 265
- [add.py](#), 265
- [add_attribute](#)
 - [pysar::_pysar_utilities](#), 49
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), 240
- [add_attribute.py](#), 265
- [add_attribute_insarmaps.py](#), 266
- [add_files](#)
 - [pysar::add](#), 74
- [add_inner_title](#)
 - [pysar::view](#), 227
- [add_matrix](#)
 - [pysar::add](#), 74
- [add_plot_attribute](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), 240
- [alpha](#)
 - [pysar::multi_transect](#), 126
 - [pysar::transect_legacy](#), 193
- [amount](#)
 - [pysar::_datetime::progress_bar](#), 252
- [amp](#)
 - [pysar::load_dem](#), 110
- [ani](#)
 - [animation](#), 20
- [animation](#), 19
 - [ani](#), 20
 - [ax](#), 20
 - [dpi](#), 20
 - [fig](#), 20
 - [fileList](#), 20
 - [i](#), 20
 - [im](#), 20
 - [img](#), 20
 - [imgs](#), 21
 - [outName](#), 21
 - [savefig_kwargs](#), 21
 - [savefigDict](#), 21
 - [titleList](#), 21
 - [ttl](#), 21
 - [updatefig](#), 19
 - [work_dir](#), 21
 - [writer](#), 21
- [animation.py](#), 266
- [asc_desc.py](#), 266
- [atr](#)
 - [plot_tropcor_phase_elevation](#), 25
 - [pysar::tsviewer](#), 213
- [atr2](#)
 - [plot_tropcor_phase_elevation](#), 25
- [atr3](#)
 - [plot_tropcor_phase_elevation](#), 25
- [atr4](#)
 - [plot_tropcor_phase_elevation](#), 26
- [attribute_envi2roipac](#)
 - [pysar::_readfile](#), 62
- [attribute_exists_for_dataset](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), 240
- [attribute_gamma2roipac](#)
 - [pysar::_readfile](#), 63
- [attribute_isce2roipac](#)
 - [pysar::_readfile](#), 63
 - [pysar::prep4timeseries](#), 144
- [Attributes.md](#), 267
- [auto_adjust_xaxis_date](#)
 - [pysar::_datetime](#), 32
- [auto_adjust_yaxis](#)
 - [pysar::_network](#), 36
- [auto_figure_title](#)
 - [pysar::view](#), 227
- [auto_flip_direction](#)
 - [pysar::view](#), 227
- [auto_lalo_sequence](#)
 - [pysar::view::Basemap2](#), 235
- [auto_path_miami](#)
 - [pysar::load_data](#), 106
- [auto_row_col_num](#)
 - [pysar::view](#), 227
- [avglnSAR](#)
 - [pysar::multi_transect](#), 126
 - [pysar::transect_legacy](#), 193
- [ax](#)
 - [animation](#), 20
 - [pysar::multi_transect](#), 126
 - [pysar::transect_legacy](#), 193
- [ax_time](#)

- pysar::tsviewer, 214
- ax_ts
 - pysar::tsviewer, 214
- ax_v
 - pysar::tsviewer, 214
- axes
 - plot_tropcor_phase_elevation, 26
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 194
- axes2
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 194
- axis
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 194
- axisbg
 - pysar::tsviewer, 214
- azimuth_bandwidth
 - pysar::_network, 37
- azimuth_ground_resolution
 - pysar::_pysar_utilities, 49
- BASELINE_LIST_FILE
 - pysar::_network, 47
- BL_LIST
 - pysar::plot_network, 143
- BOOL_ZERO
 - pysar::save_hdfeos5, 161
- bandwidth
 - pysar::_sensor::JERS, 249
- baseline_error.py, 267
- baseline_trop.py, 267
- baselineTimeseries
 - pysar::prep4timeseries, 144
 - pysar::prep_isce, 150
- Basemap2, 234
- BasicHTTP, 237
- bbox_geo2radar
 - pysar::subset, 174
- bbox_inches
 - plot_tropcor_phase_elevation, 26
 - pysar::tsviewer, 214
- bbox_radar2geo
 - pysar::subset, 174
- Bh_Bv_timeseries
 - pysar::_pysar_utilities, 49
- Bibliography.md, 267
- bin_variance
 - pysar::_variance, 70
- bodyOutput
 - pysar::add_attribute_insarmaps::InsarDataset←
Controller, 247
- box_geo2pixel
 - pysar::subset, 174
- box_pixel2geo
 - pysar::subset, 174
- bridging_data
 - pysar::unwrap_error, 223
- build_parser
 - pysar::add_attribute_insarmaps, 75
 - pysar::hdfeos5_2insarmaps, 88
 - pysar::hdfeos5_2json_mbtiles, 89
 - pysar::insarmaps_query, 102
 - pysar::json_mbtiles2insarmaps, 103
- buildURL
 - pysar::insarmaps_query, 102
- c
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- c_prof_edge
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- CPX_ZERO
 - pysar::save_hdfeos5, 161
- calculate_doppler_overlap
 - pysar::_network, 37
- cbar
 - pysar::tsviewer, 214
- ceil_to_1
 - pysar::ifgram_inversion, 92
- center_frequency
 - pysar::_sensor::JERS, 249
- cf
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- check_box_within_data_coverage
 - pysar::subset, 175
- check_colormap_input
 - pysar::view, 228
- check_drop_ifgram
 - pysar::_pysar_utilities, 50
- check_exclude_date
 - pysar::dem_error, 80
- check_existed_hdf5_file
 - pysar::load_data, 106
- check_file_size
 - pysar::_pysar_utilities, 50
 - pysar::load_data, 107
- check_geocode_file
 - pysar::pysarApp, 153
- check_loaded_dataset
 - pysar::_pysar_utilities, 50
- check_multilook_input
 - pysar::view, 228
- check_parallel
 - pysar::_pysar_utilities, 50
- check_result
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- check_result2
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- check_st_in_box
 - pysar::multi_transect, 122
 - pysar::transect_legacy, 190
- check_st_in_box2
 - pysar::multi_transect, 122

- pysar::transect_legacy, 190
- check_subset_file
 - pysar::pysarApp, 153
- check_variable_name
 - pysar::_pysar_utilities, 51
 - pysar::_readfile, 63
 - pysar::prep4timeseries, 144
- cid
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
 - pysar::tsviewer, 214
- circle_index
 - pysar::_pysar_utilities, 51
- clim
 - pysar::tsviewer, 214
- close
 - delayTimeseries::timeseries, 256
 - pysar::_datetime::progress_bar, 251
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- closest_weather_product_time
 - pysar::_datetime, 32
- cluster_table_using
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- cmap
 - pysar::tsviewer, 215
- cmdLineParse
 - pysar::add, 74
 - pysar::asc_desc, 76
 - pysar::dem_error, 81
 - pysar::diff, 82
 - pysar::generate_mask, 84
 - pysar::geocode, 85
 - pysar::ifgram_inversion, 92
 - pysar::ifgram_simulation, 98
 - pysar::image_math, 98
 - pysar::load_data, 107
 - pysar::lod, 112
 - pysar::mask, 114
 - pysar::match, 115
 - pysar::modify_network, 117
 - pysar::multilook, 141
 - pysar::plot_network, 142
 - pysar::prep4timeseries, 144
 - pysar::prep_gamma, 147
 - pysar::prep_giant_ifg_list, 149
 - pysar::prep_isce, 150
 - pysar::prep_roipac, 152
 - pysar::pysarApp, 154
 - pysar::reference_epoch, 156
 - pysar::remove_plane, 158
 - pysar::save_gmt, 159
 - pysar::save_hdfeos5, 160
 - pysar::save_kml, 162
 - pysar::save_roipac, 164
 - pysar::seed_data, 165
 - pysar::select_network, 169
 - pysar::spatial_average, 170
 - pysar::spatial_filter, 171
 - pysar::stacking, 172
 - pysar::subset, 175
 - pysar::temporal_filter, 181
 - pysar::timeseries2velocity, 182
 - pysar::timeseries_rms, 184
 - pysar::transect, 185
 - pysar::tropcor_phase_elevation, 207
 - pysar::tropcor_pyaps, 208
 - pysar::tsviewer, 212
 - pysar::unwrap_error, 224
 - pysar::view, 228
 - troposphere_uncertainty, 233
- coherence2fisher_info_index
 - pysar::ifgram_inversion, 92
- coherence2phase_variance_ds
 - pysar::ifgram_inversion, 92
- coherence_matrix
 - pysar::_network, 37
- color
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 195
- colormap
 - pysar::tsviewer, 215
- cols
 - delayTimeseries::timeseries, 258
- compression
 - pysar::load_dem, 110
- con
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 243
- connect
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- convert_data
 - pysar::hdfeos5_2json_mbtiles, 89
- coord_geo2radar
 - pysar::subset, 175
- coord_glob2radar.py, 267
- coord_radar2geo
 - pysar::subset, 175
- coord_radar2glob.py, 268
- Coordinate.md, 268
- copy_file
 - pysar::load_data, 107
- corners
 - pysar::match, 116
- correct_dem.py, 268
- correct_lod_file
 - pysar::lod, 112
- correlation_with_dem.py, 268
- create_area_table_if_not_exists
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- createParser
 - pysar::prep4timeseries, 145
 - pysar::prep_isce, 150

critical_perp_baseline
 pysar::_network, 37

curl_login
 pysar::add_attribute_insarmaps::InsarResultset↔
 Controller, 246

cursor
 pysar::add_attribute_insarmaps::InsarResultset↔
 Controller, 243

D
 pysar::multi_transect, 127
 pysar::transect_legacy, 195

d_ts
 pysar::tsviewer, 215

d_v
 pysar::tsviewer, 215

DATA_INFO
 pysar::tropcor_pyaps, 209

DATE12_LIST
 pysar::plot_network, 143

dBh_dBv_timeseries
 pysar::_pysar_utilities, 51

DEM.md, 269

DESCRIPTION
 pysar::prep_gamma, 149
 pysar::prep_roipac, 153
 pysar::temporal_coherence, 180
 pysar::unwrap_error, 225

DROP_DATE_TXT
 pysar::timeseries2velocity, 183

Data
 delayTimeseries::timeseries, 258

data
 plot_tropcor_phase_elevation, 26
 pysar::load_dem, 110

data2
 plot_tropcor_phase_elevation, 26

data_lim
 pysar::tsviewer, 215

data_operation
 pysar::image_math, 99

dataList
 plot_tropcor_phase_elevation, 26

dataset
 pysar::multi_transect, 128
 pysar::transect_legacy, 195

date12_list2index
 pysar::_network, 38

date_index
 pysar::_datetime, 33

date_list
 pysar::_pysar_utilities, 51

date_list2grib_file
 pysar::tropcor_pyaps, 208

date_list2tbase
 pysar::_datetime, 33

date_list2vector
 pysar::_datetime, 33

date_num
 pysar::tsviewer, 215

dateList
 delayTimeseries::timeseries, 258
 pysar::tsviewer, 215

daterange
 dloadUtil, 22

dates
 pysar::tsviewer, 215

db
 pysar::add_attribute_insarmaps::InsarResultset↔
 Controller, 243

dbHost
 pysar::json_mbtiles2insarmaps, 104

dbPassword
 pysar::json_mbtiles2insarmaps, 104

dbUsername
 pysar::json_mbtiles2insarmaps, 104

delayTimeseries, 22
 nearest_valid, 22
 write_to_h5, 22

delayTimeseries.py, 268

delayTimeseries::timeseries
 __init__, 256
 close, 256
 cols, 258
 Data, 258
 dateList, 258
 dist, 258
 distance, 256
 estimate_seasonal, 256
 file, 258
 h5, 258
 idx, 258
 lat, 258
 lat_first, 259
 lat_step, 259
 load, 257
 lon, 259
 lon_first, 259
 lon_step, 259
 numDates, 259
 numPixels, 259
 open, 257
 relative_std, 259
 relative_std_velocity, 259
 sample, 257
 statistics, 257
 std_timeseries, 257
 std_velocity, 257
 uncertainty_vs_distance, 257

delimiter
 pysar::tsviewer, 215

dem
 plot_tropcor_phase_elevation, 26
 pysar::load_dem, 111
 pysar::tsviewer, 216

dem_atr
 plot_tropcor_phase_elevation, 26

- dem_error.py, 269
- dem_file
 - pysar::tsviewer, 216
- demFile
 - plot_tropcor_phase_elevation, 26
 - pysar::load_dem, 111
- demRsc
 - pysar::load_dem, 111
- design_matrix
 - pysar::_pysar_utilities, 51
- df0
 - pysar::transect_legacy, 195
- df0_km
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- df1
 - pysar::transect_legacy, 195
- dg
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- diff.py, 269
- diff_data
 - pysar::diff, 82
- diff_file
 - pysar::diff, 82
- dist
 - delayTimeseries::timeseries, 258
- dist_point_from_line
 - pysar::multi_transect, 122
 - pysar::transect_legacy, 190
- DistGPS
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- distance
 - delayTimeseries::timeseries, 256
- dload_grib
 - pysar::tropcor_pyaps, 208
- dloadUtil, 22
 - daterange, 22
 - download_atmosphereModel, 23
 - download_modis, 23
 - get_date, 23
 - pwv2zwd, 23
 - read_modis, 23
 - zwd2swd, 23
- dloadUtil.py, 270
- dms2d
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- Documentation-Generation.md, 270
- download
 - troposphere_uncertainty, 233
- download_atmosphereModel
 - dloadUtil, 23
- download_modis
 - dloadUtil, 23
- dp
 - pysar::multi_transect, 128
- dpi
 - animation, 20
 - plot_tropcor_phase_elevation, 27
 - pysar::tsviewer, 216
- draw_lalo_label
 - pysar::view::Basemap2, 236
- drawscale
 - pysar::view::Basemap2, 236
- dset
 - pysar::load_dem, 111
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 196
- dtype
 - pysar::tsviewer, 216
- DX
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 196
- dx
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 196
- DY
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- dy
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 196
- e_ts
 - pysar::tsviewer, 216
- EXAMPLE
 - pysar::add, 74
 - pysar::asc_desc, 77
 - pysar::dem_error, 82
 - pysar::generate_mask, 84
 - pysar::geocode, 88
 - pysar::ifgram_inversion, 97
 - pysar::ifgram_simulation, 98
 - pysar::image_math, 99
 - pysar::load_data, 109
 - pysar::lod, 112
 - pysar::mask, 115
 - pysar::match, 117
 - pysar::modify_network, 119
 - pysar::multilook, 141
 - pysar::plot_network, 143
 - pysar::prep4timeseries, 146
 - pysar::prep_gamma, 149
 - pysar::prep_giant_ifg_list, 150
 - pysar::prep_roipac, 153
 - pysar::pysarApp, 154
 - pysar::reference_epoch, 157
 - pysar::remove_plane, 158
 - pysar::save_gmt, 160
 - pysar::save_hdfeos5, 162
 - pysar::save_kml, 163
 - pysar::save_roipac, 165
 - pysar::seed_data, 168
 - pysar::select_network, 169
 - pysar::spatial_average, 170

- pysar::spatial_filter, 172
- pysar::stacking, 173
- pysar::subset, 178
- pysar::temporal_coherence, 180
- pysar::temporal_filter, 182
- pysar::timeseries2velocity, 183
- pysar::timeseries_rms, 185
- pysar::transect, 187
- pysar::tropcor_phase_elevation, 207
- pysar::tropcor_pyaps, 209
- pysar::tsviewer, 217
- pysar::unwrap_error, 225
- pysar::view, 231
- troposphere_uncertainty, 234
- earth_radius
 - pysar::transect_legacy, 196
- ecmwf
 - plot_tropcor_phase_elevation, 27
- ecmwFile
 - plot_tropcor_phase_elevation, 27
- ecolor
 - pysar::tsviewer, 216
- epoch
 - plot_tropcor_phase_elevation, 27
 - pysar::tsviewer, 216
- error_file
 - pysar::tsviewer, 216
- error_fileContent
 - pysar::tsviewer, 216
- error_ts
 - pysar::tsviewer, 217
- estimate_seasonal
 - delayTimeseries::timeseries, 256
 - troposphere_uncertainty, 233
- ex_date
 - pysar::tsviewer, 217
- ex_date_list
 - pysar::tsviewer, 217
- ex_dates
 - pysar::tsviewer, 217
- ex_error_ts
 - pysar::tsviewer, 217
- ex_idx_list
 - pysar::tsviewer, 217
- Example.md, 270
- ext
 - pysar::load_dem, 111
- extract_attribute
 - pysar::prep_roipac, 152
- extract_attribute_dem_geo
 - pysar::prep_gamma, 147
- extract_attribute_dem_radar
 - pysar::prep_gamma, 147
- extract_attribute_interferogram
 - pysar::prep_gamma, 147
- extract_attribute_lookup_table
 - pysar::prep_gamma, 148
- extractIsceMetadata
 - pysar::prep4timeseries, 145
 - pysar::prep_isce, 151
- FLOAT_ZERO
 - pysar::save_hdfeos5, 162
- facecolor
 - pysar::tsviewer, 217
- Fault_lat
 - pysar::multi_transect, 129
- fault_loc
 - pysar::transect_legacy, 196
- Fault_lon
 - pysar::multi_transect, 129
- FaultCoords
 - pysar::multi_transect, 129
- FaultLine
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- fig
 - animation, 20
 - plot_tropcor_phase_elevation, 27
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- fig2
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 197
- fig_base
 - pysar::tsviewer, 217
- fig_ts
 - pysar::tsviewer, 218
- fig_v
 - pysar::tsviewer, 218
- figName
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 197
- figsize
 - plot_tropcor_phase_elevation, 27
 - pysar::tsviewer, 218
- figsize_multi
 - pysar, 31
- figsize_single_max
 - pysar, 31
- figsize_single_min
 - pysar, 31
- file
 - delayTimeseries::timeseries, 258
 - pysar::_sensor::JERS, 249
- File-Descriptions.md, 270
- file_list
 - pysar::tsviewer, 218
- file_operation
 - pysar::image_math, 99
- fileExtension
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 197
- fileList
 - animation, 20
- fileName
 - pysar::multi_transect, 130

- pysar::transect_legacy, 197
- filter_data
 - pysar::spatial_filter, 171
- filter_file
 - pysar::spatial_filter, 171
- find_row_column
 - pysar::insar_vs_gps, 101
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- flip_ud
 - pysar::tsviewer, 218
- fmt
 - pysar::tsviewer, 218
- fontsize
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 197
- format_coord
 - pysar::tsviewer, 212, 218
- four_corners
 - pysar::_pysar_utilities, 52
- GDAL2NUMPY_DATATYPE
 - pysar::prep4timeseries, 146
 - pysar::prep_isce, 152
- GPS_in_bound
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 197
- GPS_in_bound_st
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 197
- GPS_lat
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 198
- GPS_lon
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 198
- GPS_station
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- GPSx
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- GPSxx
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- GPSy
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- GPSyy
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 199
- GPS
 - pysar::multi_transect, 130
 - pysar::transect_legacy, 197
- Gamma-File-Decription.md, 270
- gamma_view.py, 270
- generate_curls
 - pysar::_pysar_utilities, 52
- generate_mask.py, 270
- geocode.py, 271
- geocode_file
 - pysar::geocode, 85
- geocode_file_geo_lut
 - pysar::geocode, 85
- geocode_file_radar_lut
 - pysar::geocode, 85
- geocode_output_filename
 - pysar::geocode, 86
- geometry_dataset
 - pysar::_readfile, 68
- get
 - pysar::insarmaps_query::BasicHTTP, 237
- get_H5_filename
 - pysar::hdfeos5_2insarmaps, 88
- get_box_overlap_index
 - pysar::subset, 175
- get_coverage_box
 - pysar::subset, 176
- get_dataset_id
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- get_dataset_names
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 241
- get_date
 - dloadUtil, 23
 - pysar::hdfeos5_2json_mbtiles, 89
- get_date12_list
 - pysar::_network, 38
- get_decimal_date
 - pysar::hdfeos5_2json_mbtiles, 90
- get_delay
 - pysar::tropcor_pyaps, 208
- get_distance
 - pysar::_variance, 70
- get_epoch_full_list_from_input
 - pysar::view, 228
- get_exclude_date
 - pysar::timeseries2velocity, 182
- get_file_list
 - pysar::_pysar_utilities, 52
- get_file_stack
 - pysar::_pysar_utilities, 52
- get_geo_lat_lon
 - pysar::save_gmt, 159
- get_geometry_file
 - pysar::_pysar_utilities, 53
- get_hdfeos5_filename
 - pysar::save_hdfeos5, 160
- get_intersect
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- get_lalo_ref
 - pysar::prep_gamma, 148
- get_lat_lon
 - pysar::_variance, 70
 - pysar::multi_transect, 123

- pysar::transect_legacy, 191
- get_lookup_file
 - pysar::_pysar_utilities, 53
- get_lookup_row_col
 - pysar::_pysar_utilities, 53
- get_mission_name
 - pysar::prep_giant_ifg_list, 149
 - pysar::save_hdfeos5, 161
- get_modis_v3, 23
 - main, 24
 - out, 24
 - start_time_main, 24
 - time_elapsed, 24
 - usage, 24
- get_modis_v3.py, 271
- get_overlap_lalo
 - pysar::asc_desc, 76
- get_perp_baseline
 - pysar::prep_gamma, 148
- get_residual_rms
 - pysar::_pysar_utilities, 53
- get_residual_std
 - pysar::_pysar_utilities, 54
- get_scale_from_disp_unit
 - pysar::transect, 185
- get_start_end_point
 - pysar::multi_transect, 123
- get_transect
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- get_triangles
 - pysar::_pysar_utilities, 54
- get_unavco_name
 - pysar::json_mbtiles2insarmaps, 103
- get_velocity_filename
 - pysar::timeseries2velocity, 183
- glob2radar
 - pysar::_pysar_utilities, 54
- Google-Earth.md, 272
- gps_to_LOS
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- gpsFile
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- gpsLOS_ref
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- gpsLOS
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 198
- group
 - pysar::load_dem, 111
- gx
 - pysar::multi_transect, 131
 - pysar::transect_legacy, 199
- gy
 - pysar::multi_transect, 132
- pysar::transect_legacy, 199
- h5
 - delayTimeseries::timeseries, 258
 - pysar::load_dem, 111
 - pysar::tsviewer, 218
- h5file_theta
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- hbound
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- hdfeos5_2insarmaps.py, 272
- hdfeos5_2json_mbtiles.py, 272
- header_info
 - pysar::tsviewer, 219
- headersOutput
 - pysar::add_attribute_insarmaps::InsarDataset←
Controller, 247
- heading
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- hillshade
 - pysar::_pysar_utilities, 54
- Home.md, 272
- host
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 243
- i
 - animation, 20
 - plot_tropcor_phase_elevation, 27
- IDXref
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 200
- IDYref
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
- IDX
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- IDY
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 200
- IFGRAM_LIST_FILE
 - pysar::_network, 47
- INT_ZERO
 - pysar::save_hdfeos5, 162
- idx
 - delayTimeseries::timeseries, 258
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- idxRef
 - pysar::multi_transect, 132
 - pysar::transect_legacy, 199
- ifgram_closure.py, 272
- ifgram_date_list
 - pysar::_datetime, 33
- ifgram_inversion

- pysar::ifgram_inversion, 92
- ifgram_inversion.py, 273
- ifgram_inversion_patch
 - pysar::ifgram_inversion, 93
- ifgram_reconstruction.py, 273
- ifgram_simulation.py, 273
- igram_perp_baseline_list
 - pysar::_network, 38
- im
 - animation, 20
- image_math.py, 274
- img
 - animation, 20
 - pysar::tsviewer, 219
- imgs
 - animation, 21
- incidence_angle
 - pysar::_network, 38
 - pysar::_pysar_utilities, 55
- incidence_angle.py, 274
- index_table_on
 - pysar::add_attribute_insarmaps::InsarDatabase↔Controller, 241
- info.py, 274
- Info_aboutFault
 - pysar::transect_legacy, 200
- inps
 - pysar::tsviewer, 219
- input_ex_date
 - pysar::tsviewer, 219
- insar_vs_gps.py, 275
- insarData
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
- InsarDatabaseController, 238
- InsarDatasetController, 244
- insarmaps_query.py, 275
- insert_dataset_into_area_table
 - pysar::add_attribute_insarmaps::InsarDatabase↔Controller, 242
- interp_weights
 - pysar::geocode, 86
- interpolate
 - pysar::geocode, 86
- interpolation
 - pysar::tsviewer, 219
- is_file_exist
 - pysar::_pysar_utilities, 55
- is_plot_attribute
 - pysar::_readfile, 63
- JERS, 248
- json_mbtiles2insarmaps.py, 275
- k
 - pysar::tsviewer, 219
- l1
 - pysar::l1, 104
- l1.py, 276
- l1blas
 - pysar::l1, 104
- l1mosek
 - pysar::l1, 105
- l1mosek2
 - pysar::l1, 105
- LOGO
 - pysar::pysarApp, 155
- Lat
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
- lat
 - delayTimeseries::timeseries, 258
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
 - pysar::tsviewer, 219
- Lat0
 - pysar::multi_transect, 133
- Lat1
 - pysar::multi_transect, 133
- lat_all
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
- lat_first
 - delayTimeseries::timeseries, 259
- lat_step
 - delayTimeseries::timeseries, 259
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 200
 - pysar::tsviewer, 219
- lat_transect
 - pysar::multi_transect, 133
 - pysar::transect_legacy, 201
- lbound
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
- left_lr
 - pysar::tsviewer, 220
- Length
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
- length
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
 - pysar::tsviewer, 220
- line
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 192
- linewidth
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
- list_ifgram2date12
 - pysar::_datetime, 34
- load
 - delayTimeseries::timeseries, 257
- load_data.py, 276
- load_data_from_template

- pysar::load_data, 107
- load_dem.py, 277
- load_file
 - pysar::load_data, 107
- load_geometry_hdf5
 - pysar::load_data, 108
- load_multi_group_hdf5
 - pysar::load_data, 108
- load_single_dataset_hdf5
 - pysar::load_data, 108
- lod.py, 277
- log
 - pysar::select_network, 169
- Lon
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
- lon
 - delayTimeseries::timeseries, 259
 - pysar::multi_transect, 134
 - pysar::transect_legacy, 201
 - pysar::tsviewer, 220
- Lon0
 - pysar::multi_transect, 134
- Lon1
 - pysar::multi_transect, 134
- lon_all
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 201
- lon_first
 - delayTimeseries::timeseries, 259
- lon_step
 - delayTimeseries::timeseries, 259
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
 - pysar::tsviewer, 220
- lon_transect
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- look_angle.py, 277
- los2enu.py, 278
- lrlat
 - pysar::tsviewer, 220
- lrlon
 - pysar::tsviewer, 220
- m
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- m1
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- m_prof_edge
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- main
 - get_modis_v3, 24
 - pysar::add, 74
 - pysar::add_attribute, 75
 - pysar::add_attribute_insarmaps, 76
 - pysar::asc_desc, 76
 - pysar::baseline_error, 77
 - pysar::baseline_trop, 78
 - pysar::coord_glob2radar, 78
 - pysar::coord_radar2glob, 79
 - pysar::correct_dem, 79
 - pysar::correlation_with_dem, 80
 - pysar::dem_error, 81
 - pysar::diff, 83
 - pysar::gamma_view, 83
 - pysar::generate_mask, 84
 - pysar::geocode, 87
 - pysar::hdfeos5_2insarmaps, 89
 - pysar::hdfeos5_2json_mbtiles, 90
 - pysar::ifgram_closure, 91
 - pysar::ifgram_inversion, 93
 - pysar::ifgram_reconstruction, 97
 - pysar::ifgram_simulation, 98
 - pysar::image_math, 99
 - pysar::incidence_angle, 100
 - pysar::info, 100
 - pysar::insar_vs_gps, 101
 - pysar::insarmaps_query, 102
 - pysar::json_mbtiles2insarmaps, 103
 - pysar::load_data, 109
 - pysar::lod, 112
 - pysar::look_angle, 113
 - pysar::los2enu, 113
 - pysar::mask, 114
 - pysar::match, 116
 - pysar::modify_network, 117
 - pysar::multi_transect, 124
 - pysar::multilook, 141
 - pysar::perp_baseline, 142
 - pysar::plot_network, 143
 - pysar::prep4timeseries, 145
 - pysar::prep_gamma, 148
 - pysar::prep_giant_ifg_list, 149
 - pysar::prep_isce, 151
 - pysar::prep_roipac, 152
 - pysar::pysarApp, 154
 - pysar::quality_map, 155
 - pysar::range_distance, 156
 - pysar::reference_epoch, 156
 - pysar::remove_plane, 158
 - pysar::rewrap, 158
 - pysar::save_gmt, 159
 - pysar::save_hdfeos5, 161
 - pysar::save_kml, 162
 - pysar::save_mat, 163
 - pysar::save_roipac, 164
 - pysar::seed_data, 165
 - pysar::select_network, 169
 - pysar::spatial_average, 170
 - pysar::spatial_filter, 172
 - pysar::stacking, 173
 - pysar::subset, 176
 - pysar::sum_epochs, 179

- pysar::temporal_average, 179
- pysar::temporal_coherence, 180
- pysar::temporal_derivative, 181
- pysar::temporal_filter, 182
- pysar::timeseries2velocity, 183
- pysar::timeseries_rms, 184
- pysar::transect, 185
- pysar::transect_legacy, 192
- pysar::tropcor_phase_elevation, 207
- pysar::tropcor_pyaps, 209
- pysar::unwrap_error, 224
- pysar::view, 228
- troposphere_uncertainty, 233
- make_json_file
 - pysar::hdfeos5_2json_mbtils, 90
- make_triangle
 - pysar::_pysar_utilities, 55
- manual_offset_estimate
 - pysar::match, 116
- manual_select_pairs_to_remove
 - pysar::modify_network, 118
- manual_select_reference_yx
 - pysar::seed_data, 166
- manual_select_start_end_point
 - pysar::transect, 186
- markededgecolor
 - pysar::tsviewer, 220
- mask
 - plot_tropcor_phase_elevation, 27
 - pysar::tsviewer, 220
- mask.py, 278
- mask_file
 - pysar::mask, 114
 - pysar::tsviewer, 220
- mask_matrix
 - pysar::mask, 114
- maskFile
 - plot_tropcor_phase_elevation, 27
- matFile
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- match.py, 278
- match_two_files
 - pysar::match, 116
- max
 - pysar::_datetime::progress_bar, 252
- metadata_pysar2unavco
 - pysar::save_hdfeos5, 161
- mf
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- mfc
 - pysar::multi_transect, 135
 - pysar::transect_legacy, 202
- miami_path
 - pysar, 31
- min
 - pysar::_datetime::progress_bar, 253
- mode
 - pysar::_network, 39
 - pysar::_pysar_utilities, 55
 - pysar::load_data, 109
- modify_file_date12_list
 - pysar::modify_network, 118
- modify_network.py, 279
- mp
 - pysar::transect_legacy, 202
- mplColors
 - pysar::view, 232
- ms
 - plot_tropcor_phase_elevation, 28
 - pysar::multi_transect, 136
 - pysar::transect_legacy, 203
 - pysar::tsviewer, 221
- msk_atr
 - plot_tropcor_phase_elevation, 28
- multi_dataset_hdf5_file
 - pysar::_readfile, 68
- multi_group_hdf5_file
 - pysar::_readfile, 68
- multi_transect.py, 279
- multilook.py, 282
- multilook_attribute
 - pysar::multilook, 141
- multilook_dataset
 - pysar::pysarApp, 154
- multilook_file
 - pysar::multilook, 141
- multilook_matrix
 - pysar::multilook, 141
- NOTE
 - pysar::seed_data, 168
- nanmean
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 192
- nanstd
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 192
- ncols
 - plot_tropcor_phase_elevation, 28
- ndx
 - plot_tropcor_phase_elevation, 28
- nearest
 - pysar::insar_vs_gps, 101
 - pysar::match, 116
 - pysar::multi_transect, 124
 - pysar::seed_data, 166
 - pysar::transect_legacy, 192
- nearest_neighbor
 - pysar::modify_network, 118
- nearest_valid
 - delayTimeseries, 22
- needed_attributes
 - pysar::hdfeos5_2json_mbtils, 90
- network_inversion_sbass
 - pysar::ifgram_inversion, 94

- network_inversion_wls
 - pysar::ifgram_inversion, [94](#)
- NoInSAR
 - pysar::multi_transect, [136](#)
 - pysar::transect_legacy, [203](#)
- nonzero_mask
 - pysar::_pysar_utilities, [56](#)
- normalize_timeseries
 - pysar::_pysar_utilities, [56](#)
- normalize_timeseries_old
 - pysar::_pysar_utilities, [56](#)
- nrows
 - plot_tropcor_phase_elevation, [28](#)
 - pysar::multi_transect, [136](#)
 - pysar::transect_legacy, [203](#)
- Num_profiles
 - pysar::multi_transect, [136](#)
- numDates
 - delayTimeseries::timeseries, [259](#)
- numPixels
 - delayTimeseries::timeseries, [259](#)
- onclick
 - pysar::multi_transect, [125](#)
 - pysar::transect_legacy, [192](#)
- open
 - delayTimeseries::timeseries, [257](#)
- orientation
 - pysar::tsviewer, [221](#)
- out
 - get_modis_v3, [24](#)
- outName
 - animation, [21](#)
 - pysar::load_dem, [111](#)
 - pysar::tsviewer, [221](#)
- PLOT_TEMPLATE
 - pysar::view, [232](#)
- pair_merge
 - pysar::_network, [39](#)
- pair_sort
 - pysar::_network, [39](#)
- parallel_num
 - pysar, [31](#)
- password
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, [243](#)
- perp_baseline.py, [282](#)
- perp_baseline_ifgram2timeseries
 - pysar::_pysar_utilities, [56](#)
- perp_baseline_timeseries
 - pysar::_pysar_utilities, [56](#)
- phase_pdf_ds
 - pysar::ifgram_inversion, [94](#)
- phase_variance_ds
 - pysar::ifgram_inversion, [95](#)
- phase_variance_ps
 - pysar::ifgram_inversion, [95](#)
- plot_attribute_exists_for_dataset
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, [242](#)
- plot_bar_std
 - pysar::_plot, [48](#)
- plot_coherence_history
 - pysar::_network, [39](#)
- plot_coherence_matrix
 - pysar::_network, [39](#)
- plot_dem_lalo
 - pysar::view, [229](#)
- plot_dem_yx
 - pysar::view, [229](#)
- plot_matrix
 - pysar::view, [229](#)
- plot_network
 - pysar::_network, [40](#)
- plot_network.py, [283](#)
- plot_perp_baseline_hist
 - pysar::_network, [40](#)
- plot_timeseries_errorbar
 - pysar::tsviewer, [212](#)
- plot_timeseries_event
 - pysar::tsviewer, [212](#)
- plot_timeseries_scatter
 - pysar::tsviewer, [212](#)
- plot_tropcor_phase_elevation, [25](#)
 - atr, [25](#)
 - atr2, [25](#)
 - atr3, [25](#)
 - atr4, [26](#)
 - axes, [26](#)
 - bbox_inches, [26](#)
 - data, [26](#)
 - data2, [26](#)
 - dataList, [26](#)
 - dem, [26](#)
 - dem_atr, [26](#)
 - demFile, [26](#)
 - dpi, [27](#)
 - ecmwf, [27](#)
 - ecmwfFile, [27](#)
 - epoch, [27](#)
 - fig, [27](#)
 - figsize, [27](#)
 - i, [27](#)
 - mask, [27](#)
 - maskFile, [27](#)
 - ms, [28](#)
 - msk_atr, [28](#)
 - ncols, [28](#)
 - ndx, [28](#)
 - nrows, [28](#)
 - sharex, [28](#)
 - sharey, [28](#)
 - timeseriesFile, [28](#)
 - timeseriesFile2, [28](#)
 - tropHgt, [29](#)
 - tropHgtFile, [29](#)

- True, 29
- workDir, 29
- plot_tropcor_phase_elevation.py, 283
- point_on_line_with_distance_from_beginning
 - pysar::multi_transect, 125
- point_with_distance_from_line
 - pysar::multi_transect, 125
- prefix
 - pysar::_datetime::progress_bar, 253
- prep4timeseries.py, 284
- prep_gamma.py, 285
- prep_giant_ifg_list.py, 285
- prep_isce.py, 285
- prep_roipac.py, 286
- prepare_geometry
 - pysar::prep4timeseries, 145
 - pysar::prep_isce, 151
- prepare_stack
 - pysar::prep4timeseries, 145
 - pysar::prep_isce, 151
- prf
 - pysar::_sensor::JERS, 249
- print_attributes
 - pysar::info, 100
- print_hdf5_structure
 - pysar::info, 100
- print_timseries_date_info
 - pysar::info, 100
- print_warning
 - pysar::seed_data, 166
- progBar
 - pysar::_datetime::progress_bar, 253
- progress_bar, 250
- project_name2sensor
 - pysar::load_data, 109
 - pysar::select_network, 169
- pwv2zwd
 - dloadUtil, 23
- pysar, 29
 - figsize_multi, 31
 - figsize_single_max, 31
 - figsize_single_min, 31
 - miami_path, 31
 - parallel_num, 31
- pysar._datetime, 32
- pysar._gmt, 35
- pysar._network, 36
- pysar._plot, 47
- pysar._pysar_utilities, 48
- pysar._readfile, 62
- pysar._remove_surface, 69
- pysar._sensor, 70
- pysar._variance, 70
- pysar._writefile, 71
- pysar.add, 73
- pysar.add_attribute, 75
- pysar.add_attribute_insarmaps, 75
- pysar.asc_desc, 76
- pysar.baseline_error, 77
- pysar.baseline_trop, 78
- pysar.coord_glob2radar, 78
- pysar.coord_radar2glob, 79
- pysar.correct_dem, 79
- pysar.correlation_with_dem, 80
- pysar.dem_error, 80
- pysar.diff, 82
- pysar.gamma_view, 83
- pysar.generate_mask, 83
- pysar.geocode, 84
- pysar.hdfeos5_2insarmaps, 88
- pysar.hdfeos5_2json_mbtiles, 89
- pysar.ifgram_closure, 91
- pysar.ifgram_inversion, 91
- pysar.ifgram_reconstruction, 97
- pysar.ifgram_simulation, 97
- pysar.image_math, 98
- pysar.incidence_angle, 99
- pysar.info, 100
- pysar.insar_vs_gps, 101
- pysar.insarmaps_query, 102
- pysar.json_mbtiles2insarmaps, 103
- pysar.l1, 104
- pysar.load_data, 106
- pysar.load_dem, 110
- pysar.lod, 112
- pysar.look_angle, 113
- pysar.los2enu, 113
- pysar.mask, 114
- pysar.match, 115
- pysar.modify_network, 117
- pysar.multi_transect, 119
- pysar.multilook, 140
- pysar.perp_baseline, 142
- pysar.plot_network, 142
- pysar.prep4timeseries, 144
- pysar.prep_gamma, 147
- pysar.prep_giant_ifg_list, 149
- pysar.prep_isce, 150
- pysar.prep_roipac, 152
- pysar.pysarApp, 153
- pysar.quality_map, 155
- pysar.range_distance, 155
- pysar.reference_epoch, 156
- pysar.remove_plane, 157
- pysar.rewrap, 158
- pysar.save_gmt, 159
- pysar.save_hdfeos5, 160
- pysar.save_kml, 162
- pysar.save_mat, 163
- pysar.save_roipac, 164
- pysar.seed_data, 165
- pysar.select_network, 168
- pysar.spatial_average, 170
- pysar.spatial_filter, 171
- pysar.stacking, 172
- pysar.subset, 173

- pysar.sum_epochs, 178
- pysar.temporal_average, 179
- pysar.temporal_coherence, 179
- pysar.temporal_derivative, 181
- pysar.temporal_filter, 181
- pysar.timeseries2velocity, 182
- pysar.timeseries_rms, 184
- pysar.transect, 185
- pysar.transect_legacy, 187
- pysar.tropcor_phase_elevation, 207
- pysar.tropcor_pyaps, 207
- pysar.tsviewer, 210
- pysar.unwrap_error, 223
- pysar.view, 226
- pysar::_datetime
 - auto_adjust_axis_date, 32
 - closest_weather_product_time, 32
 - date_index, 33
 - date_list2tbase, 33
 - date_list2vector, 33
 - ifgram_date_list, 33
 - list_ifgram2date12, 34
 - read_date_list, 34
 - yymmdd, 34
 - yymmdd2yyyymmdd, 34
 - yyyymmdd, 34
 - yyyymmdd2years, 35
- pysar::_datetime::progress_bar
 - __init__, 251
 - amount, 252
 - close, 251
 - max, 252
 - min, 253
 - prefix, 253
 - progBar, 253
 - reset, 252
 - span, 253
 - start_time, 253
 - suffix, 253
 - update, 252
 - update_amount, 252
 - width, 253
- pysar::_gmt
 - write_gmt_simple, 35
- pysar::_network
 - auto_adjust_yaxis, 36
 - azimuth_bandwidth, 37
 - BASELINE_LIST_FILE, 47
 - calculate_doppler_overlap, 37
 - coherence_matrix, 37
 - critical_perp_baseline, 37
 - date12_list2index, 38
 - get_date12_list, 38
 - IFGRAM_LIST_FILE, 47
 - igram_perp_baseline_list, 38
 - incidence_angle, 38
 - mode, 39
 - pair_merge, 39
 - pair_sort, 39
 - plot_coherence_history, 39
 - plot_coherence_matrix, 39
 - plot_network, 40
 - plot_perp_baseline_hist, 40
 - range_bandwidth, 41
 - read_baseline_file, 41
 - read_igram_pairs, 41
 - read_pairs_list, 41
 - select_master_date, 42
 - select_master_interferogram, 42
 - select_pairs_all, 42
 - select_pairs_delaunay, 42
 - select_pairs_hierarchical, 43
 - select_pairs_mst, 43
 - select_pairs_sequential, 43
 - select_pairs_star, 44
 - signal2noise_ratio, 44
 - simulate_coherence, 44
 - threshold_coherence_based_mst, 45
 - threshold_doppler_overlap, 45
 - threshold_perp_baseline, 46
 - threshold_temporal_baseline, 46
 - wavelength, 46
 - write_pairs_list, 47
- pysar::_plot
 - plot_bar_std, 48
- pysar::_pysar_utilities
 - add_attribute, 49
 - azimuth_ground_resolution, 49
 - Bh_Bv_timeseries, 49
 - check_drop_ifgram, 50
 - check_file_size, 50
 - check_loaded_dataset, 50
 - check_parallel, 50
 - check_variable_name, 51
 - circle_index, 51
 - dBh_dBv_timeseries, 51
 - date_list, 51
 - design_matrix, 51
 - four_corners, 52
 - generate_curls, 52
 - get_file_list, 52
 - get_file_stack, 52
 - get_geometry_file, 53
 - get_lookup_file, 53
 - get_lookup_row_col, 53
 - get_residual_rms, 53
 - get_residual_std, 54
 - get_triangles, 54
 - glob2radar, 54
 - hillshade, 54
 - incidence_angle, 55
 - is_file_exist, 55
 - make_triangle, 55
 - mode, 55
 - nonzero_mask, 56
 - normalize_timeseries, 56

- normalize_timeseries_old, 56
- perp_baseline_ifgram2timeseries, 56
- perp_baseline_timeseries, 56
- radar2glob, 57
- range_distance, 57
- range_ground_resolution, 57
- spatial_average, 58
- stacking, 58
- temporal_average, 58
- timeseries_coherence, 59
- timeseries_inversion_FGLS, 59
- timeseries_inversion_L1, 59
- timeseries_rms, 59
- timeseries_std, 60
- touch, 60
- update_attribute_or_not, 60
- update_file, 60
- update_template_file, 61
- which, 61
- yymmdd, 61
- yymmdd2YYYYMMDD, 61
- yyyyymmdd, 62
- pysar::_readfile
 - attribute_envi2roipac, 62
 - attribute_gamma2roipac, 63
 - attribute_isce2roipac, 63
 - check_variable_name, 63
 - geometry_dataset, 68
 - is_plot_attribute, 63
 - multi_dataset_hdf5_file, 68
 - multi_group_hdf5_file, 68
 - read, 63
 - read_GPS_USGS, 66
 - read_attribute, 64
 - read_bool, 64
 - read_complex_float32, 64
 - read_complex_int16, 65
 - read_float32, 65
 - read_gamma_par, 66
 - read_isce_xml, 66
 - read_multiple, 67
 - read_real_float32, 67
 - read_real_float64, 67
 - read_real_int16, 67
 - read_roipac_rsc, 68
 - read_template, 68
 - single_dataset_hdf5_file, 69
- pysar::_remove_surface
 - remove_data_multiple_surface, 69
 - remove_data_surface, 69
 - remove_surface, 69
- pysar::_sensor::JERS
 - __init__, 249
 - bandwidth, 249
 - center_frequency, 249
 - file, 249
 - prf, 249
 - swath_width, 250
- pysar::_variance
 - bin_variance, 70
 - get_distance, 70
 - get_lat_lon, 70
 - sample_data, 70
 - structure_function, 71
- pysar::_writefile
 - write, 71
 - write_complex64, 72
 - write_complex_int16, 72
 - write_dem, 72
 - write_float32, 72
 - write_real_float32, 73
 - write_real_int16, 73
 - write_roipac_rsc, 73
- pysar::add
 - add_files, 74
 - add_matrix, 74
 - cmdLineParse, 74
 - EXAMPLE, 74
 - main, 74
- pysar::add_attribute
 - main, 75
 - usage, 75
- pysar::add_attribute_insarmaps
 - build_parser, 75
 - main, 76
- pysar::add_attribute_insarmaps::InsarDatabase↔
 - Controller
 - __init__, 240
 - add_attribute, 240
 - add_plot_attribute, 240
 - attribute_exists_for_dataset, 240
 - close, 241
 - cluster_table_using, 241
 - con, 243
 - connect, 241
 - create_area_table_if_not_exists, 241
 - cursor, 243
 - db, 243
 - get_dataset_id, 241
 - get_dataset_names, 241
 - host, 243
 - index_table_on, 241
 - insert_dataset_into_area_table, 242
 - password, 243
 - plot_attribute_exists_for_dataset, 242
 - remove_dataset_if_there, 242
 - remove_point_table_if_there, 242
 - run_raw_query, 242
 - table_exists, 243
 - username, 243
- pysar::add_attribute_insarmaps::InsarDatasetController
 - __init__, 246
 - bodyOutput, 247
 - curl_login, 246
 - headersOutput, 247
 - remove_mbtiles, 246

- serverPassword, [247](#)
- serverUsername, [247](#)
- setup_curl, [246](#)
- upload_mbtiles, [246](#)
- pysar::asc_desc
 - cmdLineParse, [76](#)
 - EXAMPLE, [77](#)
 - get_overlap_lalo, [76](#)
 - main, [76](#)
 - REFERENCE, [77](#)
- pysar::baseline_error
 - main, [77](#)
 - to_percent, [77](#)
 - usage, [77](#)
- pysar::baseline_trop
 - main, [78](#)
 - to_percent, [78](#)
 - usage, [78](#)
- pysar::coord_glob2radar
 - main, [78](#)
 - usage, [78](#)
- pysar::coord_radar2glob
 - main, [79](#)
 - usage, [79](#)
- pysar::correct_dem
 - main, [79](#)
 - usage, [79](#)
- pysar::correlation_with_dem
 - main, [80](#)
 - usage, [80](#)
- pysar::dem_error
 - check_exclude_date, [80](#)
 - cmdLineParse, [81](#)
 - EXAMPLE, [82](#)
 - main, [81](#)
 - REFERENCE, [82](#)
 - read_template2inps, [81](#)
 - TEMPLATE, [82](#)
 - topographic_residual_inversion, [81](#)
- pysar::diff
 - cmdLineParse, [82](#)
 - diff_data, [82](#)
 - diff_file, [82](#)
 - main, [83](#)
 - usage, [83](#)
- pysar::gamma_view
 - main, [83](#)
 - usage, [83](#)
- pysar::generate_mask
 - cmdLineParse, [84](#)
 - EXAMPLE, [84](#)
 - main, [84](#)
- pysar::geocode
 - cmdLineParse, [85](#)
 - EXAMPLE, [88](#)
 - geocode_file, [85](#)
 - geocode_file_geo_lut, [85](#)
 - geocode_file_radar_lut, [85](#)
 - geocode_output_filename, [86](#)
 - interp_weights, [86](#)
 - interpolate, [86](#)
 - main, [87](#)
 - read_template2inps, [87](#)
 - TEMPLATE, [88](#)
 - update_attribute_geo_lut, [87](#)
 - update_attribute_radar_lut, [87](#)
- pysar::hdfeos5_2insarmaps
 - build_parser, [88](#)
 - get_H5_filename, [88](#)
 - main, [89](#)
- pysar::hdfeos5_2json_mbtiles
 - build_parser, [89](#)
 - convert_data, [89](#)
 - get_date, [89](#)
 - get_decimal_date, [90](#)
 - main, [90](#)
 - make_json_file, [90](#)
 - needed_attributes, [90](#)
 - region_name_from_project_name, [90](#)
 - serialize_dictionary, [90](#)
- pysar::ifgram_closure
 - main, [91](#)
 - usage, [91](#)
- pysar::ifgram_inversion
 - ceil_to_1, [92](#)
 - cmdLineParse, [92](#)
 - coherence2fisher_info_index, [92](#)
 - coherence2phase_variance_ds, [92](#)
 - EXAMPLE, [97](#)
 - ifgram_inversion, [92](#)
 - ifgram_inversion_patch, [93](#)
 - main, [93](#)
 - network_inversion_sbass, [94](#)
 - network_inversion_wls, [94](#)
 - phase_pdf_ds, [94](#)
 - phase_variance_ds, [95](#)
 - phase_variance_ps, [95](#)
 - REFERENCE, [97](#)
 - read_template2inps, [95](#)
 - round_to_1, [96](#)
 - TEMPLATE, [97](#)
 - temporal_coherence, [96](#)
 - write_timeseries_hdf5_file, [96](#)
- pysar::ifgram_reconstruction
 - main, [97](#)
 - usage, [97](#)
- pysar::ifgram_simulation
 - cmdLineParse, [98](#)
 - EXAMPLE, [98](#)
 - main, [98](#)
- pysar::image_math
 - cmdLineParse, [98](#)
 - data_operation, [99](#)
 - EXAMPLE, [99](#)
 - file_operation, [99](#)
 - main, [99](#)

- pysar::incidence_angle
 - main, 100
 - usage, 100
- pysar::info
 - main, 100
 - print_attributes, 100
 - print_hdf5_structure, 100
 - print_timseries_date_info, 100
 - usage, 101
- pysar::insar_vs_gps
 - find_row_column, 101
 - main, 101
 - nearest, 101
 - readGPSfile, 101
 - usage, 102
- pysar::insarmaps_query
 - build_parser, 102
 - buildURL, 102
 - main, 102
- pysar::insarmaps_query::BasicHTTP
 - get, 237
- pysar::json_mbtiles2insarmaps
 - build_parser, 103
 - dbHost, 104
 - dbPassword, 104
 - dbUsername, 104
 - get_unavco_name, 103
 - main, 103
 - upload_insarmaps_metadata, 103
 - upload_json, 103
- pysar::l1
 - __MOSEK, 105
 - l1, 104
 - l1blas, 104
 - l1mosek, 105
 - l1mosek2, 105
 - task, 105
 - x, 106
- pysar::load_data
 - auto_path_miami, 106
 - check_existed_hdf5_file, 106
 - check_file_size, 107
 - cmdLineParse, 107
 - copy_file, 107
 - EXAMPLE, 109
 - load_data_from_template, 107
 - load_file, 107
 - load_geometry_hdf5, 108
 - load_multi_group_hdf5, 108
 - load_single_dataset_hdf5, 108
 - main, 109
 - mode, 109
 - project_name2sensor, 109
 - sensorList, 110
 - TEMPLATE, 110
- pysar::load_dem
 - amp, 110
 - compression, 110
 - data, 110
 - dem, 111
 - demFile, 111
 - demRsc, 111
 - dset, 111
 - ext, 111
 - group, 111
 - h5, 111
 - outName, 111
- pysar::lod
 - cmdLineParse, 112
 - correct_lod_file, 112
 - EXAMPLE, 112
 - main, 112
 - REFERENCE, 112
- pysar::look_angle
 - main, 113
 - usage, 113
- pysar::los2enu
 - main, 113
 - usage, 113
- pysar::mask
 - cmdLineParse, 114
 - EXAMPLE, 115
 - main, 114
 - mask_file, 114
 - mask_matrix, 114
 - update_mask, 115
- pysar::match
 - cmdLineParse, 115
 - corners, 116
 - EXAMPLE, 117
 - main, 116
 - manual_offset_estimate, 116
 - match_two_files, 116
 - nearest, 116
- pysar::modify_network
 - cmdLineParse, 117
 - EXAMPLE, 119
 - main, 117
 - manual_select_pairs_to_remove, 118
 - modify_file_date12_list, 118
 - nearest_neighbor, 118
 - read_template2inps, 118
 - reset_pairs, 119
 - TEMPLATE, 119
- pysar::multi_transect
 - alpha, 126
 - avglnSAR, 126
 - ax, 126
 - axes, 126
 - axes2, 126
 - axis, 126
 - c, 127
 - c_prof_edge, 127
 - cf, 127
 - check_result, 127
 - check_result2, 127

- check_st_in_box, [122](#)
- check_st_in_box2, [122](#)
- cid, [127](#)
- color, [127](#)
- D, [127](#)
- dataset, [128](#)
- df0_km, [128](#)
- dg, [128](#)
- dist_point_from_line, [122](#)
- DistGPS, [128](#)
- dms2d, [123](#)
- dp, [128](#)
- dset, [128](#)
- DX, [128](#)
- dx, [128](#)
- DY, [129](#)
- dy, [128](#)
- Fault_lat, [129](#)
- Fault_lon, [129](#)
- FaultCoords, [129](#)
- FaultLine, [129](#)
- fig, [129](#)
- fig2, [129](#)
- figName, [129](#)
- fileExtension, [130](#)
- fileName, [130](#)
- find_row_column, [123](#)
- fontsize, [130](#)
- GPS_in_bound, [130](#)
- GPS_in_bound_st, [130](#)
- GPS_lat, [130](#)
- GPS_lon, [130](#)
- GPS_station, [131](#)
- GPSx, [131](#)
- GPSxx, [131](#)
- GPSy, [131](#)
- GPSyy, [131](#)
- GPS, [130](#)
- get_intersect, [123](#)
- get_lat_lon, [123](#)
- get_start_end_point, [123](#)
- get_transect, [123](#)
- gps_to_LOS, [124](#)
- gpsFile, [131](#)
- gpsLOS_ref, [131](#)
- gpsLOS, [131](#)
- gx, [131](#)
- gy, [132](#)
- h5file_theta, [132](#)
- hbound, [132](#)
- heading, [132](#)
- IDXref, [132](#)
- IDYref, [133](#)
- IDX, [132](#)
- IDY, [132](#)
- idx, [132](#)
- idxRef, [132](#)
- insarData, [133](#)
- Lat, [133](#)
- lat, [133](#)
- Lat0, [133](#)
- Lat1, [133](#)
- lat_all, [133](#)
- lat_step, [133](#)
- lat_transect, [133](#)
- lbound, [134](#)
- Length, [134](#)
- length, [134](#)
- line, [124](#)
- linewidth, [134](#)
- Lon, [134](#)
- lon, [134](#)
- Lon0, [134](#)
- Lon1, [134](#)
- lon_all, [135](#)
- lon_step, [135](#)
- lon_transect, [135](#)
- m, [135](#)
- m1, [135](#)
- m_prof_edge, [135](#)
- main, [124](#)
- matFile, [135](#)
- mf, [135](#)
- mfc, [135](#)
- ms, [136](#)
- nanmean, [124](#)
- nanstd, [124](#)
- nearest, [124](#)
- NoInSAR, [136](#)
- nrows, [136](#)
- Num_profiles, [136](#)
- onclick, [125](#)
- point_on_line_with_distance_from_beginning, [125](#)
- point_with_distance_from_line, [125](#)
- read_fault_coords, [125](#)
- readGPSfile, [125](#)
- redGPSfile, [125](#)
- redGPSfile_cmm4, [126](#)
- Se, [136](#)
- Sn, [136](#)
- Stations, [136](#)
- stationsList, [136](#)
- stdInSAR, [137](#)
- theta, [137](#)
- transect, [137](#)
- transect_lat, [137](#)
- transect_lon, [137](#)
- unitVec, [137](#)
- usage, [126](#)
- Ve, [137](#)
- Vn, [137](#)
- Width, [137](#)
- x, [138](#)
- X0, [138](#)
- x0, [138](#)
- X1, [138](#)

- x1, [138](#)
- XX0, [139](#)
- XX1, [139](#)
- xc, [138](#)
- Xf0, [138](#)
- Xf1, [138](#)
- xlim, [138](#)
- y, [139](#)
- Y0, [139](#)
- y0, [139](#)
- Y1, [139](#)
- y1, [139](#)
- YY0, [140](#)
- YY1, [140](#)
- yc, [139](#)
- Yf0, [139](#)
- Yf1, [140](#)
- ylim, [140](#)
- zi, [140](#)
- pysar::multilook
 - cmdLineParse, [141](#)
 - EXAMPLE, [141](#)
 - main, [141](#)
 - multilook_attribute, [141](#)
 - multilook_file, [141](#)
 - multilook_matrix, [141](#)
- pysar::perp_baseline
 - main, [142](#)
 - usage, [142](#)
- pysar::plot_network
 - BL_LIST, [143](#)
 - cmdLineParse, [142](#)
 - DATE12_LIST, [143](#)
 - EXAMPLE, [143](#)
 - main, [143](#)
 - read_template2inps, [143](#)
 - TEMPLATE, [143](#)
- pysar::prep4timeseries
 - attribute_isce2roipac, [144](#)
 - baselineTimeseries, [144](#)
 - check_variable_name, [144](#)
 - cmdLineParse, [144](#)
 - createParser, [145](#)
 - EXAMPLE, [146](#)
 - extractIsceMetadata, [145](#)
 - GDAL2NUMPY_DATATYPE, [146](#)
 - main, [145](#)
 - prepare_geometry, [145](#)
 - prepare_stack, [145](#)
 - read_baseline, [145](#)
 - read_rsc, [146](#)
 - read_template, [146](#)
 - write_rsc, [146](#)
- pysar::prep_gamma
 - cmdLineParse, [147](#)
 - DESCRIPTION, [149](#)
 - EXAMPLE, [149](#)
 - extract_attribute_dem_geo, [147](#)
 - extract_attribute_dem_radar, [147](#)
 - extract_attribute_interferogram, [147](#)
 - extract_attribute_lookup_table, [148](#)
 - get_lalo_ref, [148](#)
 - get_perp_baseline, [148](#)
 - main, [148](#)
- pysar::prep_giant_ifg_list
 - cmdLineParse, [149](#)
 - EXAMPLE, [150](#)
 - get_mission_name, [149](#)
 - main, [149](#)
- pysar::prep_isce
 - baselineTimeseries, [150](#)
 - cmdLineParse, [150](#)
 - createParser, [150](#)
 - extractIsceMetadata, [151](#)
 - GDAL2NUMPY_DATATYPE, [152](#)
 - main, [151](#)
 - prepare_geometry, [151](#)
 - prepare_stack, [151](#)
 - read_baseline, [151](#)
 - write_rsc, [151](#)
- pysar::prep_roipac
 - cmdLineParse, [152](#)
 - DESCRIPTION, [153](#)
 - EXAMPLE, [153](#)
 - extract_attribute, [152](#)
 - main, [152](#)
- pysar::pysarApp
 - check_geocode_file, [153](#)
 - check_subset_file, [153](#)
 - cmdLineParse, [154](#)
 - EXAMPLE, [154](#)
 - LOGO, [155](#)
 - main, [154](#)
 - multilook_dataset, [154](#)
 - subset_dataset, [154](#)
 - TEMPLATE, [155](#)
 - UM_FILE_STRUCT, [155](#)
- pysar::quality_map
 - main, [155](#)
 - usage, [155](#)
- pysar::range_distance
 - main, [156](#)
 - usage, [156](#)
- pysar::reference_epoch
 - cmdLineParse, [156](#)
 - EXAMPLE, [157](#)
 - main, [156](#)
 - read_template2inps, [156](#)
 - ref_date_attribute, [157](#)
 - ref_date_file, [157](#)
 - TEMPLATE, [157](#)
- pysar::remove_plane
 - cmdLineParse, [158](#)
 - EXAMPLE, [158](#)
 - main, [158](#)
- pysar::rewrap

- main, 158
- rewrap, 158
- usage, 159
- pysar::save_gmt
 - cmdLineParse, 159
 - EXAMPLE, 160
 - get_geo_lat_lon, 159
 - main, 159
 - write_grd_file, 159
- pysar::save_hdfEOS5
 - BOOL_ZERO, 161
 - CPX_ZERO, 161
 - cmdLineParse, 160
 - EXAMPLE, 162
 - FLOAT_ZERO, 162
 - get_hdfEOS5_filename, 160
 - get_mission_name, 161
 - INT_ZERO, 162
 - main, 161
 - metadata_pysar2unavco, 161
 - read_template2inps, 161
 - TEMPALTE, 162
- pysar::save_kml
 - cmdLineParse, 162
 - EXAMPLE, 163
 - main, 162
 - write_kmz_file, 163
- pysar::save_mat
 - main, 163
 - usage, 164
 - yyyymmdd2years, 164
- pysar::save_roipac
 - cmdLineParse, 164
 - EXAMPLE, 165
 - main, 164
 - usage, 164
- pysar::seed_data
 - cmdLineParse, 165
 - EXAMPLE, 168
 - main, 165
 - manual_select_reference_yx, 166
 - NOTE, 168
 - nearest, 166
 - print_warning, 166
 - random_select_reference_yx, 166
 - read_seed_reference2inps, 166
 - read_seed_template2inps, 166
 - remove_reference_pixel, 167
 - seed_attributes, 167
 - seed_file_inps, 167
 - seed_file_reference_value, 167
 - select_max_coherence_yx, 167
 - TEMPLATE, 168
- pysar::select_network
 - cmdLineParse, 169
 - EXAMPLE, 169
 - log, 169
 - main, 169
 - project_name2sensor, 169
 - REFERENCE, 169
 - read_template2inps, 169
 - sar_sensor_list, 170
 - TEMPLATE, 170
- pysar::spatial_average
 - cmdLineParse, 170
 - EXAMPLE, 170
 - main, 170
- pysar::spatial_filter
 - cmdLineParse, 171
 - EXAMPLE, 172
 - filter_data, 171
 - filter_file, 171
 - main, 172
- pysar::stacking
 - cmdLineParse, 172
 - EXAMPLE, 173
 - main, 173
- pysar::subset
 - bbox_geo2radar, 174
 - bbox_radar2geo, 174
 - box_geo2pixel, 174
 - box_pixel2geo, 174
 - check_box_within_data_coverage, 175
 - cmdLineParse, 175
 - coord_geo2radar, 175
 - coord_radar2geo, 175
 - EXAMPLE, 178
 - get_box_overlap_index, 175
 - get_coverage_box, 176
 - main, 176
 - read_subset_template2box, 176
 - subset_attribute, 176
 - subset_box2inps, 177
 - subset_file, 177
 - subset_file_list, 177
 - subset_input_dict2box, 178
- pysar::sum_epochs
 - main, 179
 - usage, 179
- pysar::temporal_average
 - main, 179
 - usage, 179
- pysar::temporal_coherence
 - DESCRIPTION, 180
 - EXAMPLE, 180
 - main, 180
 - REFERENCE, 180
 - temporal_coherence, 180
 - USAGE, 181
 - usage, 180
- pysar::temporal_derivative
 - main, 181
 - usage, 181
- pysar::temporal_filter
 - cmdLineParse, 181
 - EXAMPLE, 182

- main, [182](#)
- pysar::timeseries2velocity
 - cmdLineParse, [182](#)
 - DROP_DATE_TXT, [183](#)
 - EXAMPLE, [183](#)
 - get_exclude_date, [182](#)
 - get_velocity_filename, [183](#)
 - main, [183](#)
 - read_template2inps, [183](#)
 - TEMPLATE, [184](#)
- pysar::timeseries_rms
 - cmdLineParse, [184](#)
 - EXAMPLE, [185](#)
 - main, [184](#)
 - read_template2inps, [184](#)
 - TEMPLATE, [185](#)
- pysar::transect
 - cmdLineParse, [185](#)
 - EXAMPLE, [187](#)
 - get_scale_from_disp_unit, [185](#)
 - main, [185](#)
 - manual_select_start_end_point, [186](#)
 - read_lonlat_file, [186](#)
 - transect_lalo, [186](#)
 - transect_list, [186](#)
 - transect_yx, [186](#)
- pysar::transect_legacy
 - alpha, [193](#)
 - avglnSAR, [193](#)
 - ax, [193](#)
 - axes, [194](#)
 - axes2, [194](#)
 - axis, [194](#)
 - c, [194](#)
 - c_prof_edge, [194](#)
 - cf, [194](#)
 - check_result, [194](#)
 - check_result2, [194](#)
 - check_st_in_box, [190](#)
 - check_st_in_box2, [190](#)
 - cid, [194](#)
 - color, [195](#)
 - D, [195](#)
 - dataset, [195](#)
 - df0, [195](#)
 - df0_km, [195](#)
 - df1, [195](#)
 - dg, [195](#)
 - dist_point_from_line, [190](#)
 - DistGPS, [195](#)
 - dms2d, [191](#)
 - dset, [196](#)
 - DX, [196](#)
 - dx, [196](#)
 - DY, [196](#)
 - dy, [196](#)
 - earth_radius, [196](#)
 - fault_loc, [196](#)
 - FaultLine, [196](#)
 - fig, [196](#)
 - fig2, [197](#)
 - figName, [197](#)
 - fileExtension, [197](#)
 - fileName, [197](#)
 - find_row_column, [191](#)
 - fontsize, [197](#)
 - GPS_in_bound, [197](#)
 - GPS_in_bound_st, [197](#)
 - GPS_lat, [198](#)
 - GPS_lon, [198](#)
 - GPS_station, [198](#)
 - GPSx, [198](#)
 - GPSxx, [198](#)
 - GPSy, [198](#)
 - GPSyy, [199](#)
 - GPS, [197](#)
 - get_intersect, [191](#)
 - get_lat_lon, [191](#)
 - get_transect, [191](#)
 - gps_to_LOS, [191](#)
 - gpsFile, [198](#)
 - gpsLOS_ref, [198](#)
 - gpsLOS, [198](#)
 - gx, [199](#)
 - gy, [199](#)
 - h5file_theta, [199](#)
 - hbound, [199](#)
 - heading, [199](#)
 - IDXref, [200](#)
 - IDYref, [200](#)
 - IDX, [199](#)
 - IDY, [200](#)
 - idx, [199](#)
 - idxRef, [199](#)
 - Info_aboutFault, [200](#)
 - insarData, [200](#)
 - Lat, [200](#)
 - lat, [200](#)
 - lat_all, [200](#)
 - lat_step, [200](#)
 - lat_transect, [201](#)
 - lbound, [201](#)
 - Length, [201](#)
 - length, [201](#)
 - line, [192](#)
 - linewidth, [201](#)
 - Lon, [201](#)
 - lon, [201](#)
 - lon_all, [201](#)
 - lon_step, [202](#)
 - lon_transect, [202](#)
 - m, [202](#)
 - m1, [202](#)
 - m_prof_edge, [202](#)
 - main, [192](#)
 - matFile, [202](#)

- mf, [202](#)
- mfc, [202](#)
- mp, [202](#)
- ms, [203](#)
- nanmean, [192](#)
- nanstd, [192](#)
- nearest, [192](#)
- NoInSAR, [203](#)
- nrows, [203](#)
- onclick, [192](#)
- readGPSfile, [193](#)
- redGPSfile, [193](#)
- redGPSfile_cmm4, [193](#)
- Se, [203](#)
- Sn, [203](#)
- Stations, [203](#)
- stationsList, [203](#)
- stdInSAR, [203](#)
- theta, [204](#)
- transect, [204](#)
- transect_lat, [204](#)
- transect_lon, [204](#)
- unitVec, [204](#)
- Usage, [193](#)
- Ve, [204](#)
- Vn, [204](#)
- Width, [204](#)
- x, [204](#)
- X0, [205](#)
- x0, [205](#)
- X1, [205](#)
- x1, [205](#)
- XX0, [205](#)
- XX1, [205](#)
- xc, [205](#)
- y, [205](#)
- Y0, [206](#)
- y0, [205](#)
- Y1, [206](#)
- y1, [206](#)
- YY0, [206](#)
- YY1, [206](#)
- yc, [206](#)
- ylim, [206](#)
- zi, [206](#)
- pysar::tropcor_phase_elevation
 - cmdLineParse, [207](#)
 - EXAMPLE, [207](#)
 - main, [207](#)
 - REFERENCE, [207](#)
- pysar::tropcor_pyaps
 - cmdLineParse, [208](#)
 - DATA_INFO, [209](#)
 - date_list2grib_file, [208](#)
 - dload_grib, [208](#)
 - EXAMPLE, [209](#)
 - get_delay, [208](#)
 - main, [209](#)
 - REFERENCE, [209](#)
 - TEMPLATE, [209](#)
- pysar::tsviewer
 - atr, [213](#)
 - ax_time, [214](#)
 - ax_ts, [214](#)
 - ax_v, [214](#)
 - axisbg, [214](#)
 - bbox_inches, [214](#)
 - cbar, [214](#)
 - cid, [214](#)
 - clim, [214](#)
 - cmap, [215](#)
 - cmdLineParse, [212](#)
 - colormap, [215](#)
 - d_ts, [215](#)
 - d_v, [215](#)
 - data_lim, [215](#)
 - date_num, [215](#)
 - dateList, [215](#)
 - dates, [215](#)
 - delimiter, [215](#)
 - dem, [216](#)
 - dem_file, [216](#)
 - dpi, [216](#)
 - dtype, [216](#)
 - e_ts, [216](#)
 - EXAMPLE, [217](#)
 - ecolor, [216](#)
 - epoch, [216](#)
 - error_file, [216](#)
 - error_fileContent, [216](#)
 - error_ts, [217](#)
 - ex_date, [217](#)
 - ex_date_list, [217](#)
 - ex_dates, [217](#)
 - ex_error_ts, [217](#)
 - ex_idx_list, [217](#)
 - facecolor, [217](#)
 - fig_base, [217](#)
 - fig_ts, [218](#)
 - fig_v, [218](#)
 - figsize, [218](#)
 - file_list, [218](#)
 - flip_ud, [218](#)
 - fmt, [218](#)
 - format_coord, [212](#), [218](#)
 - h5, [218](#)
 - header_info, [219](#)
 - img, [219](#)
 - inps, [219](#)
 - input_ex_date, [219](#)
 - interpolation, [219](#)
 - k, [219](#)
 - lat, [219](#)
 - lat_step, [219](#)
 - left_lr, [220](#)
 - length, [220](#)

- lon, [220](#)
- lon_step, [220](#)
- lrlat, [220](#)
- lrlon, [220](#)
- markeredgecolor, [220](#)
- mask, [220](#)
- mask_file, [220](#)
- ms, [221](#)
- orientation, [221](#)
- outName, [221](#)
- plot_timeseries_errorbar, [212](#)
- plot_timeseries_event, [212](#)
- plot_timeseries_scatter, [212](#)
- read_timeseries_lalo, [212](#)
- read_timeseries_yx, [213](#)
- ref_d_v, [221](#)
- ref_yx, [221](#)
- time_slider_update, [213](#)
- timeseries_file, [221](#)
- tims, [221](#)
- transparent, [221](#)
- True, [221](#)
- tslider, [222](#)
- ullat, [222](#)
- ullon, [222](#)
- unit_fac, [222](#)
- update_timeseries, [213](#)
- valinit, [222](#)
- width, [222](#)
- x, [222](#)
- y, [222](#)
- ylim_mat, [222](#)
- yticks, [223](#)
- yx, [223](#)
- zero_idx, [223](#)
- pysar::unwrap_error
 - bridging_data, [223](#)
 - cmdLineParse, [224](#)
 - DESCRIPTION, [225](#)
 - EXAMPLE, [225](#)
 - main, [224](#)
 - REFERENCE, [225](#)
 - read_template2inps, [224](#)
 - TEMPLATE, [225](#)
 - unwrap_error_correction_bridging, [224](#)
 - unwrap_error_correction_phase_closure, [224](#)
- pysar::view
 - add_inner_title, [227](#)
 - auto_figure_title, [227](#)
 - auto_flip_direction, [227](#)
 - auto_row_col_num, [227](#)
 - check_colormap_input, [228](#)
 - check_multilook_input, [228](#)
 - cmdLineParse, [228](#)
 - EXAMPLE, [231](#)
 - get_epoch_full_list_from_input, [228](#)
 - main, [228](#)
 - mplColors, [232](#)
 - PLOT_TEMPLATE, [232](#)
 - plot_dem_lalo, [229](#)
 - plot_dem_yx, [229](#)
 - plot_matrix, [229](#)
 - round_to_1, [230](#)
 - scale_data2disp_unit, [230](#)
 - scale_data4disp_unit_and_rewrap, [230](#)
 - update_matrix_with_plot_inps, [231](#)
 - update_plot_inps_with_display_setting_file, [231](#)
 - update_plot_inps_with_meta_dict, [231](#)
- pysar::view::Basemap2
 - auto_lalo_sequence, [235](#)
 - draw_lalo_label, [236](#)
 - drawscale, [236](#)
- pysarApp.py, [286](#)
- quality_map.py, [287](#)
- README.md, [287](#)
- REFERENCE
 - pysar::asc_desc, [77](#)
 - pysar::dem_error, [82](#)
 - pysar::ifgram_inversion, [97](#)
 - pysar::lod, [112](#)
 - pysar::select_network, [169](#)
 - pysar::temporal_coherence, [180](#)
 - pysar::tropcor_phase_elevation, [207](#)
 - pysar::tropcor_pyaps, [209](#)
 - pysar::unwrap_error, [225](#)
- radar2glob
 - pysar::_pysar_utilities, [57](#)
- random_select_reference_yx
 - pysar::seed_data, [166](#)
- range_bandwidth
 - pysar::_network, [41](#)
- range_distance
 - pysar::_pysar_utilities, [57](#)
- range_distance.py, [287](#)
- range_ground_resolution
 - pysar::_pysar_utilities, [57](#)
- read
 - pysar::_readfile, [63](#)
- read_GPS_USGS
 - pysar::_readfile, [66](#)
- read_attribute
 - pysar::_readfile, [64](#)
- read_baseline
 - pysar::prep4timeseries, [145](#)
 - pysar::prep_isce, [151](#)
- read_baseline_file
 - pysar::_network, [41](#)
- read_bool
 - pysar::_readfile, [64](#)
- read_complex_float32
 - pysar::_readfile, [64](#)
- read_complex_int16
 - pysar::_readfile, [65](#)
- read_date_list
 - pysar::_datetime, [34](#)

read_fault_coords
 pysar::multi_transect, 125
 read_float32
 pysar::_readfile, 65
 read_gamma_par
 pysar::_readfile, 66
 read_igram_pairs
 pysar::_network, 41
 read_isce_xml
 pysar::_readfile, 66
 read_lonlat_file
 pysar::transect, 186
 read_modis
 dloadUtil, 23
 read_multiple
 pysar::_readfile, 67
 read_pairs_list
 pysar::_network, 41
 read_real_float32
 pysar::_readfile, 67
 read_real_float64
 pysar::_readfile, 67
 read_real_int16
 pysar::_readfile, 67
 read_roipac_rsc
 pysar::_readfile, 68
 read_rsc
 pysar::prep4timeseries, 146
 read_seed_reference2inps
 pysar::seed_data, 166
 read_seed_template2inps
 pysar::seed_data, 166
 read_subset_template2box
 pysar::subset, 176
 read_template
 pysar::_readfile, 68
 pysar::prep4timeseries, 146
 read_template2inps
 pysar::dem_error, 81
 pysar::geocode, 87
 pysar::ifgram_inversion, 95
 pysar::modify_network, 118
 pysar::plot_network, 143
 pysar::reference_epoch, 156
 pysar::save_hdfeos5, 161
 pysar::select_network, 169
 pysar::timeseries2velocity, 183
 pysar::timeseries_rms, 184
 pysar::unwrap_error, 224
 read_timeseries_lalo
 pysar::tsviewer, 212
 read_timeseries_yx
 pysar::tsviewer, 213
 readGPSfile
 pysar::insar_vs_gps, 101
 pysar::multi_transect, 125
 pysar::transect_legacy, 193
 redGPSfile
 pysar::multi_transect, 125
 pysar::transect_legacy, 193
 redGPSfile_cmm4
 pysar::multi_transect, 126
 pysar::transect_legacy, 193
 ref_d_v
 pysar::tsviewer, 221
 ref_date_attribute
 pysar::reference_epoch, 157
 ref_date_file
 pysar::reference_epoch, 157
 ref_yx
 pysar::tsviewer, 221
 reference_epoch.py, 287
 region_name_from_project_name
 pysar::hdfeos5_2json_mbtiles, 90
 relative_std
 delayTimeseries::timeseries, 259
 relative_std_velocity
 delayTimeseries::timeseries, 259
 remove_data_multiple_surface
 pysar::_remove_surface, 69
 remove_data_surface
 pysar::_remove_surface, 69
 remove_dataset_if_there
 pysar::add_attribute_insarmaps::InsarDatabase←
 Controller, 242
 remove_mbtiles
 pysar::add_attribute_insarmaps::InsarDataset←
 Controller, 246
 remove_plane.py, 288
 remove_point_table_if_there
 pysar::add_attribute_insarmaps::InsarDatabase←
 Controller, 242
 remove_reference_pixel
 pysar::seed_data, 167
 remove_surface
 pysar::_remove_surface, 69
 reset
 pysar::_datetime::progress_bar, 252
 reset_pairs
 pysar::modify_network, 119
 rewrap
 pysar::rewrap, 158
 rewrap.py, 288
 round_to_1
 pysar::ifgram_inversion, 96
 pysar::view, 230
 run_raw_query
 pysar::add_attribute_insarmaps::InsarDatabase←
 Controller, 242
 SAR-Sensor-Parameter.md, 288
 sample
 delayTimeseries::timeseries, 257
 sample_data
 pysar::_variance, 70
 sar_sensor_list
 pysar::select_network, 170

- save_gmt.py, 288
- save_hdfeos5.py, 289
- save_kml.py, 289
- save_mat.py, 289
- save_roipac.py, 290
- savefig_kwargs
 - animation, 21
- savefigDict
 - animation, 21
- scale_data2disp_unit
 - pysar::view, 230
- scale_data4disp_unit_and_rewrap
 - pysar::view, 230
- Se
 - pysar::multi_transect, 136
 - pysar::transect_legacy, 203
- seed_attributes
 - pysar::seed_data, 167
- seed_data.py, 290
- seed_file_inps
 - pysar::seed_data, 167
- seed_file_reference_value
 - pysar::seed_data, 167
- select_master_date
 - pysar::_network, 42
- select_master_interferogram
 - pysar::_network, 42
- select_max_coherence_yx
 - pysar::seed_data, 167
- select_network.py, 291
- select_pairs_all
 - pysar::_network, 42
- select_pairs_delaunay
 - pysar::_network, 42
- select_pairs_hierarchical
 - pysar::_network, 43
- select_pairs_mst
 - pysar::_network, 43
- select_pairs_sequential
 - pysar::_network, 43
- select_pairs_star
 - pysar::_network, 44
- sensorList
 - pysar::load_data, 110
- serialize_dictionary
 - pysar::hdfeos5_2json_mbtiles, 90
- serverPassword
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 247
- serverUsername
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 247
- setup_curl
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 246
- sharex
 - plot_tropcor_phase_elevation, 28
- sharey
 - plot_tropcor_phase_elevation, 28
- signal2noise_ratio
 - pysar::_network, 44
- simulate_coherence
 - pysar::_network, 44
- single_dataset_hdf5_file
 - pysar::_readfile, 69
- Sn
 - pysar::multi_transect, 136
 - pysar::transect_legacy, 203
- span
 - pysar::_datetime::progress_bar, 253
- spatial_average
 - pysar::_pysar_utilities, 58
- spatial_average.py, 291
- spatial_filter.py, 292
- stacking
 - pysar::_pysar_utilities, 58
- stacking.py, 292
- start_time
 - pysar::_datetime::progress_bar, 253
- start_time_main
 - get_modis_v3, 24
- Stations
 - pysar::multi_transect, 136
 - pysar::transect_legacy, 203
- stationsList
 - pysar::multi_transect, 136
 - pysar::transect_legacy, 203
- statistics
 - delayTimeseries::timeseries, 257
 - troposphere_uncertainty, 233
- std_timeseries
 - delayTimeseries::timeseries, 257
- std_velocity
 - delayTimeseries::timeseries, 257
- stdInSAR
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 203
- structure_function
 - pysar::_variance, 71
- subset.py, 292
- subset_attribute
 - pysar::subset, 176
- subset_box2inps
 - pysar::subset, 177
- subset_dataset
 - pysar::pysarApp, 154
- subset_file
 - pysar::subset, 177
- subset_file_list
 - pysar::subset, 177
- subset_input_dict2box
 - pysar::subset, 178
- suffix
 - pysar::_datetime::progress_bar, 253
- sum_epochs.py, 293
- swath_width

- pysar::_sensor::JERS, 250
- TEMPALTE
 - pysar::save_hdfes5, 162
- TEMPLATE
 - pysar::dem_error, 82
 - pysar::geocode, 88
 - pysar::ifgram_inversion, 97
 - pysar::load_data, 110
 - pysar::modify_network, 119
 - pysar::plot_network, 143
 - pysar::pysarApp, 155
 - pysar::reference_epoch, 157
 - pysar::seed_data, 168
 - pysar::select_network, 170
 - pysar::timeseries2velocity, 184
 - pysar::timeseries_rms, 185
 - pysar::tropcor_pyaps, 209
 - pysar::unwrap_error, 225
- table_exists
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 243
- task
 - pysar::l1, 105
- temporal_average
 - pysar::_pysar_utilities, 58
- temporal_average.py, 293
- temporal_coherence
 - pysar::ifgram_inversion, 96
 - pysar::temporal_coherence, 180
- temporal_coherence.py, 294
- temporal_derivative.py, 294
- temporal_filter.py, 294
- theta
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- threshold_coherence_based_mst
 - pysar::_network, 45
- threshold_doppler_overlap
 - pysar::_network, 45
- threshold_perp_baseline
 - pysar::_network, 46
- threshold_temporal_baseline
 - pysar::_network, 46
- time_elapsed
 - get_modis_v3, 24
- time_slider_update
 - pysar::tsviewer, 213
- timeseries, 254
- timeseries2velocity.py, 295
- timeseries_coherence
 - pysar::_pysar_utilities, 59
- timeseries_file
 - pysar::tsviewer, 221
- timeseries_inversion_FGLS
 - pysar::_pysar_utilities, 59
- timeseries_inversion_L1
 - pysar::_pysar_utilities, 59
- timeseries_rms
 - pysar::_pysar_utilities, 59
- timeseries_rms.py, 295
- timeseries_std
 - pysar::_pysar_utilities, 60
- timeseriesFile
 - plot_tropcor_phase_elevation, 28
- timeseriesFile2
 - plot_tropcor_phase_elevation, 28
- tims
 - pysar::tsviewer, 221
- titleList
 - animation, 21
- to_percent
 - pysar::baseline_error, 77
 - pysar::baseline_trop, 78
- topographic_residual_inversion
 - pysar::dem_error, 81
- touch
 - pysar::_pysar_utilities, 60
- transect
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- transect.py, 295
- transect_lalo
 - pysar::transect, 186
- transect_lat
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- transect_legacy.py, 296
- transect_list
 - pysar::transect, 186
- transect_lon
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- transect_yx
 - pysar::transect, 186
- transparent
 - pysar::tsviewer, 221
- tropHgt
 - plot_tropcor_phase_elevation, 29
- tropHgtFile
 - plot_tropcor_phase_elevation, 29
- tropcor_phase_elevation.py, 299
- tropcor_pyaps.py, 299
- troposphere_uncertainty, 233
 - cmdLineParse, 233
 - download, 233
 - EXAMPLE, 234
 - estimate_seasonal, 233
 - main, 233
 - statistics, 233
 - velocity_uncertainty, 233
 - velocity_uncertainty_vs_distance, 234
- troposphere_uncertainty.py, 300
- True
 - plot_tropcor_phase_elevation, 29
 - pysar::tsviewer, 221
- tslider

- pysar::tsviewer, 222
- tsviewer.py, 300
- tll
 - animation, 21
- UM_FILE_STRUCT
 - pysar::pysarApp, 155
- UNAVCO-InSAR-Archive.md, 302
- USAGE
 - pysar::temporal_coherence, 181
- ullat
 - pysar::tsviewer, 222
- ullon
 - pysar::tsviewer, 222
- uncertainty_vs_distance
 - delayTimeseries::timeseries, 257
- unit_fac
 - pysar::tsviewer, 222
- unitVec
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- unwrap_error.py, 302
- unwrap_error_correction_bridging
 - pysar::unwrap_error, 224
- unwrap_error_correction_phase_closure
 - pysar::unwrap_error, 224
- update
 - pysar::_datetime::progress_bar, 252
- update_amount
 - pysar::_datetime::progress_bar, 252
- update_attribute_geo_lut
 - pysar::geocode, 87
- update_attribute_or_not
 - pysar::_pysar_utilities, 60
- update_attribute_radar_lut
 - pysar::geocode, 87
- update_file
 - pysar::_pysar_utilities, 60
- update_mask
 - pysar::mask, 115
- update_matrix_with_plot_inps
 - pysar::view, 231
- update_plot_inps_with_display_setting_file
 - pysar::view, 231
- update_plot_inps_with_meta_dict
 - pysar::view, 231
- update_template_file
 - pysar::_pysar_utilities, 61
- update_timeseries
 - pysar::tsviewer, 213
- updatefig
 - animation, 19
- upload_insarmaps_metadata
 - pysar::json_mbtiles2insarmaps, 103
- upload_json
 - pysar::json_mbtiles2insarmaps, 103
- upload_mbtiles
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 246

- Usage
 - pysar::transect_legacy, 193
- usage
 - get_modis_v3, 24
 - pysar::add_attribute, 75
 - pysar::baseline_error, 77
 - pysar::baseline_trop, 78
 - pysar::coord_glob2radar, 78
 - pysar::coord_radar2glob, 79
 - pysar::correct_dem, 79
 - pysar::correlation_with_dem, 80
 - pysar::diff, 83
 - pysar::gamma_view, 83
 - pysar::ifgram_closure, 91
 - pysar::ifgram_reconstruction, 97
 - pysar::incidence_angle, 100
 - pysar::info, 101
 - pysar::insar_vs_gps, 102
 - pysar::look_angle, 113
 - pysar::los2enu, 113
 - pysar::multi_transect, 126
 - pysar::perp_baseline, 142
 - pysar::quality_map, 155
 - pysar::range_distance, 156
 - pysar::unwrap, 159
 - pysar::save_mat, 164
 - pysar::save_roipac, 164
 - pysar::sum_epochs, 179
 - pysar::temporal_average, 179
 - pysar::temporal_coherence, 180
 - pysar::temporal_derivative, 181
- username
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 243
- valinit
 - pysar::tsviewer, 222
- Ve
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- velocity_uncertainty
 - troposphere_uncertainty, 233
- velocity_uncertainty_vs_distance
 - troposphere_uncertainty, 234
- view.py, 303
- Vn
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- wavelength
 - pysar::_network, 46
- Web-Viewer.md, 303
- which
 - pysar::_pysar_utilities, 61
- Width
 - pysar::multi_transect, 137
 - pysar::transect_legacy, 204
- width
 - pysar::_datetime::progress_bar, 253

- pysar::tsviewer, 222
- work_dir
 - animation, 21
- workDir
 - plot_tropcor_phase_elevation, 29
- write
 - pysar::_writefile, 71
- write_complex64
 - pysar::_writefile, 72
- write_complex_int16
 - pysar::_writefile, 72
- write_dem
 - pysar::_writefile, 72
- write_float32
 - pysar::_writefile, 72
- write_gmt_simple
 - pysar::_gmt, 35
- write_grd_file
 - pysar::save_gmt, 159
- write_kmz_file
 - pysar::save_kml, 163
- write_pairs_list
 - pysar::_network, 47
- write_real_float32
 - pysar::_writefile, 73
- write_real_int16
 - pysar::_writefile, 73
- write_roipac_rsc
 - pysar::_writefile, 73
- write_rsc
 - pysar::prep4timeseries, 146
 - pysar::prep_isce, 151
- write_timeseries_hdf5_file
 - pysar::ifgram_inversion, 96
- write_to_h5
 - delayTimeseries, 22
- writer
 - animation, 21
- x
 - pysar::l1, 106
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 204
 - pysar::tsviewer, 222
- X0
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 205
- x0
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 205
- X1
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 205
- x1
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 205
- XX0
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 205
- XX1
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 205
- xc
 - pysar::multi_transect, 138
 - pysar::transect_legacy, 205
- Xf0
 - pysar::multi_transect, 138
- Xf1
 - pysar::multi_transect, 138
- xlim
 - pysar::multi_transect, 138
- y
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 205
 - pysar::tsviewer, 222
- Y0
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 206
- y0
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 205
- Y1
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 206
- y1
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 206
- YY0
 - pysar::multi_transect, 140
 - pysar::transect_legacy, 206
- YY1
 - pysar::multi_transect, 140
 - pysar::transect_legacy, 206
- yc
 - pysar::multi_transect, 139
 - pysar::transect_legacy, 206
- Yf0
 - pysar::multi_transect, 139
- Yf1
 - pysar::multi_transect, 140
- ylim
 - pysar::multi_transect, 140
 - pysar::transect_legacy, 206
- ylim_mat
 - pysar::tsviewer, 222
- yticks
 - pysar::tsviewer, 223
- yx
 - pysar::tsviewer, 223
- yymmdd
 - pysar::_datetime, 34
 - pysar::_pysar_utilities, 61
- yymmdd2YYYYMMDD
 - pysar::_pysar_utilities, 61
- yymmdd2yyyymmdd
 - pysar::_datetime, 34
- yyyymmdd

- [pysar::_datetime](#), [34](#)
 - [pysar::_pysar_utilities](#), [62](#)
- [yyyymmdd2years](#)
 - [pysar::_datetime](#), [35](#)
 - [pysar::save_mat](#), [164](#)
- [zero_idx](#)
 - [pysar::tsviewer](#), [223](#)
- [zi](#)
 - [pysar::multi_transect](#), [140](#)
 - [pysar::transect_legacy](#), [206](#)
- [zwd2swd](#)
 - [dloadUtil](#), [23](#)