



A Python package for InSAR time series analysis

PySAR Documentation

Version 1.2.0

Zhang Yunjun, Heresh Fattahi

2017 Aug

0 Introduction

PySAR is made available for non-commercial applications only and can be downloaded from: <https://yunjunz.github.io/PySAR/>

When using this software please reference Yunjun et al. [2017]:
Yunjun Z., H. Fattahi, F. Amelung, (2017), *A geodetic approach for InSAR time series analysis: phase correction and error propagation*. (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.

PySAR is an open-source python-based software package that implement InSAR (Interferometric Synthetic Aperture Radar) time series analysis. It includes a routine time series processing approach and some independent toolbox for the time series analysis. PySAR is based on the initial work done by Scott Baker while during his PhD in University of Miami (now in UNAVCO). Alfredo Terrero develops the code for University of Miami time-series web viewer (<http://insarmaps.miami.edu>).

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	README	2
2	_Sidebar	4
3	Attributes	4
4	Bibliography	6
5	Coordinate	7
6	Corrected DEM	7
7	Example	7
8	File-Descriptions	8
9	Gamma-File-Decription	9
10	Google-Earth	10
11	Home	10
12	SAR-Sensor-Parameter	10
13	UNAVCO-InSAR-Archive	11
14	Web-Viewer	11
15	Namespace Index	12
16	Hierarchical Index	15
17	Class Index	15
18	File Index	16
19	Namespace Documentation	19
20	Class Documentation	228

21 File Documentation	254
Index	299

1 README

PySAR is a Python package for InSAR (Interferometric Synthetic Aperture Radar) time series analysis. It reads stack of unwrapped interferograms in ROI_PAC, Gamma and ISCE format, and produces three dimensional (2D in space and 1D in time) ground displacement.

1. Installation

1.1 Pre-requisite

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 /Users/jeromezhang/Documents/development/python/anaconda2
set path = ( ${PYTHON2DIR}/bin $path )
```

Then run the following in your terminal:

```
cd ~/Documents/development/python
wget https://repo.continuum.io/archive/Anaconda2-4.2.0-MacOSX-x86_64.sh
chmod +x Anaconda2-4.2.0-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 ~/Documents/development/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 ~/Documents/development/python
git clone https://github.com/yunjunz/PySAR.git
```

To use the package, you need to: 1) add the path to PySAR directory to your \$PYTHONPATH and 2) add \${PY←SAR_HOME}/pysar and \${PY←SAR_HOME}/shellscripts to your \$path. Depending on your shell, you may use commands below to setup pysar, by adding the following to your source file.

For bash user, add to your .bashrc file:

```
export PYSAR_HOME=~/.Documents/development/python/PySAR"    #for released version, "~/.Documents/development/pyt
export PYTHONPATH=${PYSAR_HOME}:${PYTHONPATH}
export PATH="${PYSAR_HOME}/pysar:${PYSAR_HOME}/shellscripts:$PATH"
```

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

```
setenv PYSAR_HOME ~/.Documents/development/python/PySAR    #for released version, "~/.Documents/development/pytho
setenv PYTHONPATH ${PYSAR_HOME}
set path = ( $PYSAR_HOME/pysar $PYSAR_HOME/shellscripts $path)
```

2. Running PySAR

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 interefrograms 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.

Example: Kuju Volcano example with ALOS data

Download the test data: [Download Link](#) and unzip it.

Create a custom template file:

```
cd ~/Documents/insarlab/KujuAlosAT422F650/PYSAR
vi KujuAlosAT422F650.template
```

Include the following pysar options in your template:

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

Save your template file and run PySAR as:

```
pysarApp.py KujuAlosAT422F650.template
```

Inside [pysarApp.py](#), it reads the unwrapped interefrograms, refernces 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 interefrograms, 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!).

Build your own processing recipe

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

3. Documentation

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

4. Google Group

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.

Required attributes from `ROI_PAC`:

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`

Optional attributes from `ROI_PAC`

- `ANTENNA_SIDE` = -1 for right looking radar, used in `save_unavco`
- `AZIMUTH_PIXEL_SIZE` = Azimuth pixel size at orbital altitude (multiply by $Re/(Re+h)$ for ground size (m), where Re is the local earth radius), used in `baseline_error/tropo` 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_BOTTOM_HDR` = Vertical baseline separation at bottom of the interferogram, used in `_pysar_utilities`

Attributes generated by PySAR automatically:

- 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
- 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

Reference

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.
2. 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.

Reference

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 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
```

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

```
cd KujuAlosAT422F650/PYSAR pysarApp.py KujuAlosAT422F650.template view.py velocity.h5 -d dem↵
Radar.h5 -u cm -m -2.5 -M 0.5 -c jet_r -lalo-label
```

By default, it's using ROI_PAC product in radar coordinate in ROIPAC/RADAR folder; if you want to try Py↵ SAR 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](#)

Reference

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.

8 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
- single_dataset (**1group-1dset-1atr**) = one group with one dataset and one attribute dict per group i.e. velocity, dem, rmse, temporal_coherence, mask

multi_group

- 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

multi_dataset

- timeseries.h5 = multi_dataset type, time series displacement, generated in network inversion step

single_dataset

- 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

ROI_PAC files

- 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

9 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 coregistered 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)

10 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](#)

11 Home

Github Page: <https://yunjunz.github.io/PySAR/> Google Group: <https://groups.google.com/forum/#!forum/py-sar>

Documentation: [PDF](#)

12 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

13 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
```

add_attribute.txt

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   = ''
```

Reference

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

14 Web-Viewer

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

<http://insarmaps.miami.edu>

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

15 Namespace Index

15.1 Packages

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

delayTimeseries	19
dloadUtil	20
get_modis_v3	21
plot_tropcor_phase_elevation	22
pysar	27
pysar._datetime	29
pysar._gmt	32
pysar._network	33
pysar._plot	43
pysar._pysar_utilities	44
pysar._readfile	57
pysar._remove_surface	63
pysar._sensor	64
pysar._variance	64
pysar._writefile	66
pysar.add	68
pysar.add_attribute	69
pysar.add_attribute_insarmaps	70
pysar.asc_desc	70
pysar.baseline_error	72
pysar.baseline_trop	72
pysar.coord_glob2radar	73
pysar.coord_radar2glob	73

pysar.correct_dem	74
pysar.correlation_with_dem	74
pysar.dem_error	75
pysar.diff	76
pysar.download_ecmwf	77
pysar.epoch_coherence	79
pysar.gamma_view	80
pysar.generate_mask	81
pysar.geocode	81
pysar.geocode_orig	83
pysar.ifgram_closure	85
pysar.ifgram_inversion	86
pysar.ifgram_reconstruction	86
pysar.ifgram_simulation	87
pysar.image_math	88
pysar.incidence_angle	89
pysar.info	89
pysar.insar_vs_gps	90
pysar.insarmaps_query	91
pysar.json_mbtiles2insarmaps	92
pysar.l1	93
pysar.load_data	95
pysar.load_data_bak	99
pysar.load_dem	102
pysar.lod	104
pysar.look_angle	104
pysar.los2enu	105
pysar.mask	105
pysar.match	107
pysar.modify_network	108
pysar.multi_transect	111
pysar.multilook	132

pysar.perp_baseline	134
pysar.plot_atmDrop	134
pysar.plot_network	138
pysar.prep_gamma	140
pysar.prep_isce	142
pysar.prep_roipac	144
pysar.pysarApp	145
pysar.quality_map	148
pysar.range_distance	148
pysar.reference_epoch	149
pysar.remove_plane	150
pysar.rewrap	151
pysar.save_gmt	152
pysar.save_kml	153
pysar.save_mat	154
pysar.save_mat_orig	154
pysar.save_roipac	155
pysar.save_unavco	156
pysar.seed_data	157
pysar.select_network	161
pysar.spatial_average	162
pysar.spatial_filter	163
pysar.subset	165
pysar.sum_epochs	170
pysar.temporal_average	170
pysar.temporal_coherence	171
pysar.temporal_derivative	172
pysar.temporal_filter	173
pysar.timeseries2velocity	173
pysar.timeseries_rms	175
pysar.transect	176
pysar.transect_legacy	179

pysar.tropcor_phase_elevation	198
pysar.tropcor_pyaps	199
pysar.tropcor_pyaps_orig	201
pysar.tsviewer	203
pysar.unavco2insarmaps	215
pysar.unavco2json_mbtiles	216
pysar.unwrap_error	218
pysar.view	220
troposphere_uncertainty	226

16 Hierarchical Index

16.1 Class Hierarchy

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

BasicHTTP	231
object	
timeseries	248
JERS	242
InsarDatabaseController	232
InsarDatasetController	238
progress_bar	244
Basemap	
Basemap2	228

17 Class Index

17.1 Class List

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

Basemap2	
Class #####	228
BasicHTTP	231
InsarDatabaseController	232
InsarDatasetController	238

JERS

Program is part of PySAR v1.0 # Copyright(c) 2016, Yunjun Zhang # Author: Yunjun Zhang # 242

progress_bar

Simple progress bar##### 244

timeseries

248

18 File Index**18.1 File List**

Here is a list of all files with brief descriptions:

__init__.py	254
_datetime.py	254
_gmt.py	255
_network.py	255
_plot.py	256
_pysar_utilities.py	256
_readfile.py	257
_remove_surface.py	258
_sensor.py	258
_variance.py	258
_writefile.py	259
add.py	259
add_attribute.py	259
add_attribute_insarmaps.py	260
asc_desc.py	260
baseline_error.py	260
baseline_trop.py	261
coord_glob2radar.py	261
coord_radar2glob.py	261
correct_dem.py	261
correlation_with_dem.py	262
delayTimeseries.py	262
dem_error.py	262

diff.py	263
dloadUtil.py	263
download_ecmwf.py	263
epoch_coherence.py	264
gamma_view.py	264
generate_mask.py	265
geocode.py	265
geocode_orig.py	265
get_modis_v3.py	266
ifgram_closure.py	266
ifgram_inversion.py	266
ifgram_reconstruction.py	267
ifgram_simulation.py	267
image_math.py	267
incidence_angle.py	268
info.py	268
insar_vs_gps.py	268
insarmaps_query.py	268
json_mbtiles2insarmaps.py	269
l1.py	269
load_data.py	270
load_data_bak.py	270
load_dem.py	271
lod.py	271
look_angle.py	271
los2enu.py	272
mask.py	272
match.py	272
modify_network.py	273
multi_transect.py	273
multilook.py	276
perp_baseline.py	276

plot_atmDrop.py	277
plot_network.py	277
plot_tropcor_phase_elevation.py	278
prep_gamma.py	279
prep_isce.py	279
prep_roipac.py	279
pysarApp.py	280
quality_map.py	280
range_distance.py	281
reference_epoch.py	281
remove_plane.py	281
rewrap.py	282
save_gmt.py	282
save_kml.py	282
save_mat.py	283
save_mat_orig.py	283
save_roipac.py	283
save_unavco.py	283
seed_data.py	284
select_network.py	284
spatial_average.py	285
spatial_filter.py	285
subset.py	285
sum_epochs.py	286
temporal_average.py	286
temporal_coherence.py	287
temporal_derivative.py	287
temporal_filter.py	287
timeseries2velocity.py	288
timeseries_rms.py	288
transect.py	288
transect_legacy.py	289

tropcor_phase_elevation.py	292
tropcor_pyaps.py	292
tropcor_pyaps_orig.py	293
troposphere_uncertainty.py	293
tsviewer.py	294
unavco2insarmaps.py	295
unavco2json_mbtiles.py	296
unwrap_error.py	296
view.py	297

19 Namespace Documentation

19.1 delayTimeseries Namespace Reference

Classes

- class [timeseries](#)

Functions

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

19.1.1 Function Documentation

19.1.1.1 nearest_valid()

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

19.1.1.2 write_to_h5()

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

19.2 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)

19.2.1 Function Documentation

19.2.1.1 daterange()

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

19.2.1.2 download_atmosphereModel()

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

19.2.1.3 download_modis()

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

19.2.1.4 get_date()

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

19.2.1.5 pwv2zwd()

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


19.2.1.6 read_modis()

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

19.2.1.7 zwd2swd()

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

19.3 get_modis_v3 Namespace Reference

Functions

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

Variables

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

19.3.1 Function Documentation

19.3.1.1 main()

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

19.3.1.2 usage()

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

19.3.2 Variable Documentation

19.3.2.1 out

```
out
```

19.3.2.2 start_time_main

start_time_main

19.3.2.3 time_elapsed

time_elapsed

19.4 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](#)

19.4.1 Variable Documentation

19.4.1.1 `atr``atr`**19.4.1.2** `atr2``atr2`**19.4.1.3** `atr3``atr3`**19.4.1.4** `atr4``atr4`**19.4.1.5** `axes``axes`**19.4.1.6** `bbox_inches``bbox_inches`**19.4.1.7** `data``data`**19.4.1.8** `data2``data2`**19.4.1.9** `dataList``dataList`

19.4.1.10 dem

dem

19.4.1.11 dem_atr

dem_atr

19.4.1.12 demFile

demFile

19.4.1.13 dpi

dpi

19.4.1.14 ecmwf

ecmwf

19.4.1.15 ecmwfFile

ecmwfFile

19.4.1.16 epoch

epoch

19.4.1.17 fig

fig

19.4.1.18 figsize

figsize

19.4.1.19 i

i

19.4.1.20 mask

mask

19.4.1.21 maskFile

maskFile

19.4.1.22 ms

ms

19.4.1.23 msk_atr

msk_atr

19.4.1.24 ncols

ncols

19.4.1.25 ndx

ndx

19.4.1.26 nrows

nrows

19.4.1.27 sharex

sharex

19.4.1.28 sharey

sharey

19.4.1.29 timeseriesFile

timeseriesFile

19.4.1.30 timeseriesFile2

timeseriesFile2

19.4.1.31 tropHgt

tropHgt

19.4.1.32 tropHgtFile

tropHgtFile

19.4.1.33 True

True

19.4.1.34 workDir

workDir

19.5 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](#)
- [download_ecmwf](#)
- [epoch_coherence](#)
- [gamma_view](#)
- [generate_mask](#)
- [geocode](#)
- [geocode_orig](#)
- [ifgram_closure](#)
- [ifgram_inversion](#)
- [ifgram_reconstruction](#)
- [ifgram_simulation](#)
- [image_math](#)
- [incidence_angle](#)
- [info](#)
- [insar_vs_gps](#)
- [insarmaps_query](#)
- [json_mbtiles2insarmaps](#)
- [l1](#)
- [load_data](#)
- [load_data_bak](#)
- [load_dem](#)
- [lod](#)
- [look_angle](#)
- [los2enu](#)
- [mask](#)
- [match](#)
- [modify_network](#)
- [multi_transect](#)
- [multilook](#)

- [perp_baseline](#)
- [plot_atmDrop](#)
- [plot_network](#)
- [prep_gamma](#)
- [prep_isce](#)
- [prep_roipac](#)
- [pysarApp](#)
- [quality_map](#)
- [range_distance](#)
- [reference_epoch](#)
- [remove_plane](#)
- [rewrap](#)
- [save_gmt](#)
- [save_kml](#)
- [save_mat](#)
- [save_mat_orig](#)
- [save_roipac](#)
- [save_unavco](#)
- [seed_data](#)
- [select_network](#)
- [spatial_average](#)
- [spatial_filter](#)
- [subset](#)
- [sum_epochs](#)
- [temporal_average](#)
- [temporal_coherence](#)
- [temporal_derivative](#)
- [temporal_filter](#)
- [timeseries2velocity](#)
- [timeseries_rms](#)
- [transect](#)
- [transect_legacy](#)
- [tropcor_phase_elevation](#)
- [tropcor_pyaps](#)
- [tropcor_pyaps_orig](#)
- [tsviewer](#)
- [unavco2insarmaps](#)
- [unavco2json_mbtiles](#)
- [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](#) = [20.0, 12.0]

19.5.1 Variable Documentation

19.5.1.1 figsize_multi

```
list figsize_multi = [20.0, 12.0]
```

19.5.1.2 figsize_single_max

```
float figsize_single_max = 12.0
```

19.5.1.3 figsize_single_min

```
float figsize_single_min = 6.0
```

19.5.1.4 miami_path

```
bool miami_path = True
```

19.5.1.5 parallel_num

```
int parallel_num = 8
```

19.6 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)
- def [list_ifgram2date12](#) (ifgram_list)

19.6.1 Function Documentation

19.6.1.1 auto_adjust_xaxis_date()

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

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
```

19.6.1.2 date_index()

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

19.6.1.3 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
```

19.6.1.4 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
```

19.6.1.5 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

19.6.1.6 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)
```

19.6.1.7 read_date_list()

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

Read Date List from txt file

19.6.1.8 yymmdd()

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

19.6.1.9 yymmdd2yyyymmdd()

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

19.6.1.10 `yyyymmdd()`

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

19.6.1.11 `yyyymmdd2years()`

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

19.7 `pysar._gmt` Namespace Reference

Functions

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

19.7.1 Function Documentation

19.7.1.1 `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
```

19.8 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)
- def [igram_perp_baseline_list](#) (File)
- def [azimuth_bandwidth](#) (sensor)
- def [range_bandwidth](#) (sensor)
- def [wavelength](#) (sensor)
- def [incidence_angle](#) (sensor)
- def [signal2noise_ratio](#) (sensor)
- def [critical_perp_baseline](#) (sensor)
- def [calculate_doppler_overlap](#) (dop_a, dop_b, bandwidth_az)
- 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)
- 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=[])
- def [plot_perp_baseline_hist](#) (ax, date8_list, pbase_list, plot_dict={}, date8_list_drop=[])
- def [plot_coherence_matrix](#) (ax, date12_list, coherence_list, 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](#)

19.8.1 Function Documentation

19.8.1.1 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
```

19.8.1.2 azimuth_bandwidth()

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

Find the hardwired azimuth bandwidth in hertz for the given satellite

19.8.1.3 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.
```

19.8.1.4 coherence_matrix()

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

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
```

19.8.1.5 critical_perp_baseline()

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

Critical Perpendicular Baseline for each satellite

19.8.1.6 date12_list2index()

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

Convert list of date12 string into list of index

19.8.1.7 get_date12_list()

```
def pysar._network.get_date12_list (
    File )
```

Read Date12 info from input file: Pairs.list or 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('Pairs.list')
```

19.8.1.8 igram_perp_baseline_list()

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

Get perpendicular baseline list from input multi_group hdf5 file

19.8.1.9 incidence_angle()

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

19.8.1.10 mode()

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

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

19.8.1.11 pair_merge()

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

19.8.1.12 pair_sort()

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

19.8.1.13 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

19.8.1.14 plot_coherence_matrix()

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

Plot Coherence Matrix of input network

19.8.1.15 plot_network()

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

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
    coh_date12_list: list of date, corresponding to coherence_list
    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
```

Output

```
ax : matplotlib axes object
```

19.8.1.16 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
date8_list_drop : list of string, date dropped in YYYYMMDD format
    e.g. ['20080711', '20081011']
```

Output:

```
ax : matplotlib axes object
```

19.8.1.17 range_bandwidth()

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

19.8.1.18 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	0.000000000000	2155.2	/scratch/KyushuT422F650AlosA/SLC/070106/
070709	2631.9	0.07	0.0000000	0.000000000000	0.000000000000	2155.2	/scratch/KyushuT422F650AlosA/SLC/070709/
070824	2787.3	0.07	0.0000000	0.000000000000	0.000000000000	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]
```

19.8.1.19 read_igram_pairs()

```
def pysar._network.read_igram_pairs (
    igramFile )
```

Read pairs index from hdf5 file

19.8.1.20 read_pairs_list()

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

Read Pairs List file like below:

```
070311-070426
070311-070611
...
```

19.8.1.21 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.

19.8.1.22 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; otherwise, choose the closest pair among all pairs as master interferogram.

Example:

```
master_date12 = pnet.select_master_ifgram(date12_list, date_list, pbase_list)
```

19.8.1.23 select_pairs_all()

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

Select All Possible Pairs/Interferograms

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

Output: date12_list - list date12 in YYMMDD-YYMMDD 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.

19.8.1.24 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),

19.8.1.25 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.

19.8.1.26 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

19.8.1.27 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

19.8.1.28 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),

19.8.1.29 signal2noise_ratio()

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

Find 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>

19.8.1.30 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
```

19.8.1.31 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
```

19.8.1.32 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)
```

19.8.1.33 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)
```

19.8.1.34 wavelength()

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

19.8.1.35 write_pairs_list()

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

Write pairs list file.

19.8.2 Variable Documentation

19.8.2.1 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      0.000000  0.000000  2155.2   /KyushuT422F650AlosA/SLC/070106/
4 070709   2631.9     0.07      0.000000  0.000000  2155.2   /KyushuT422F650AlosA/SLC/070709/
5 070824   2787.3     0.07      0.000000  0.000000  2155.2   /KyushuT422F650AlosA/SLC/070824/
6 ...
7 '''
```

19.8.2.2 IFGRAM_LIST_FILE

string IFGRAM_LIST_FILE

Initial value:

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

19.9 pysar._plot Namespace Reference

Functions

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

19.9.1 Function Documentation

19.9.1.1 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

19.10 pysar._pysar_utilities Namespace Reference

Functions

- def [check_loaded_dataset](#) (work_dir='.', inps=None, print_message=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)
- def [perp_baseline_timeseries](#) (atr, dimension=1)
- def [range_distance](#) (atr, dimension=2)
- def [incidence_angle](#) (atr, dimension=2, print_message=True)
- def [which](#) (program)
- def [get_file_stack](#) (File, maskFile=None)
- def [check_drop_ifgram](#) (h5, atr, ifgram_list, print_message=True)
- def [nonzero_mask](#) (File, outFile='mask.h5')
- def [get_spatial_average](#) (File, maskFile=None, box=None, saveList=True)
- def [spatial_average](#) (File, mask=None, box=None, saveList=False)
- def [temporal_average](#) (File, outFile=None)
- def [get_file_list](#) (fileList, abspath=False)
- def [check_file_size](#) (fname_list, mode_width=None, mode_length=None)
- def [mode](#) (thelist)
- def [range_resolution](#) (atr, print_message=True)
- def [azimuth_resolution](#) (atr)
- def [glob2radar](#) (lat, lon, transFile='geomap *.trans', atr_rdr=dict(), print_message=True)
- def [radar2glob](#) (az, rg, transFile='geomap *.trans', atr_rdr=dict(), print_message=True)
- def [check_variable_name](#) (path)
- def [hillshade](#) (data, scale)
- def [date_list](#) (h5file)
- def [design_matrix](#) (ifgramFile=None, date12_list=[])
- def [timeseries_inversion](#) (ifgramFile, timeseriesFile)
- 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 [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)

19.10.1 Function Documentation

19.10.1.1 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

19.10.1.2 azimuth_resolution()

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

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

19.10.1.3 Bh_Bv_timeseries()

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

19.10.1.4 check_drop_ifgram()

```
def pysar._pysar_utilities.check_drop_ifgram (
    h5,
    atr,
    ifgram_list,
    print_message = True )
```

Update ifgram_list based on 'drop_ifgram' attribute

Inputs:

- h5 - HDF5 file object
- atr - dict, file attribute
- ifgram_list - list of string, all group name existed in file

Outputs:

- ifgram_list_out - list of string, group name with drop_ifgram = 'yes'
- ifgram_list_drop - list of string, group name with drop_ifgram = 'no'

19.10.1.5 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.

19.10.1.6 check_loaded_dataset()

```
def pysar._pysar_utilities.check_loaded_dataset (
    work_dir = './',
    inps = None,
    print_message = 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
 trans_file : string, file name/path of transformation mapping 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)
```

19.10.1.7 check_parallel()

```
def pysar._pysar_utilities.check_parallel (
    file_num = 1 )
```

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

19.10.1.8 check_variable_name()

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

19.10.1.9 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')

19.10.1.10 date_list()

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

19.10.1.11 dBh_dBv_timeseries()

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

19.10.1.12 design_matrix()

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

Make the design matrix for the inversion based on date12_list.

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 (igram_num, date_num-1)

representing date combination for each interferogram

B - 2D np.array in size (igram_num, date_num-1)

representing temporal baseline timeseries between master and slave date for each interferogram

19.10.1.13 four_corners()

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

Return 4 corners lat/lon

19.10.1.14 generate_curls()

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

19.10.1.15 get_file_list()

```
def pysar._pysar_utilities.get_file_list (
    fileList,
    abspath = False )
```

Get all existed files matching the input list of file pattern

Inputs:

fileList - string or list of string, input file pattern
abspath - bool, return absolute path or not

Output:

fileListOut - list of string, existed file path/name

Example:

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

19.10.1.16 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

19.10.1.17 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('timeseries_ECMWF_demErrInvResid.h5', 'maskTempCoh.h5')
```

19.10.1.18 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')
```

19.10.1.19 get_spatial_average()

```
def pysar._pysar_utilities.get_spatial_average (
    File,
    maskFile = None,
    box = None,
    saveList = True )
```

Get spatial average info from input File.

Inputs:

File - string, path of HDF5 file or txt 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

outputs:

mean_list - list of float, spatial average value of file
 date_list - list of string for date info

Example:

```
import pysar._pysar_utilities as ut
mean_list, date_list = ut.get_spatial_average('coherence.h5', 'maskTempCoh.h5')
```

19.10.1.20 get_triangles()

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

19.10.1.21 glob2radar()

```
def pysar._pysar_utilities.glob2radar (
    lat,
    lon,
    transFile = 'geomap*.trans',
    atr_rdr = dict(),
    print_message = True )
```

Convert geo coordinates into radar coordinates.

Inputs:

```
lat/lon      - np.array, float, latitude/longitude
transFile    - 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
```

19.10.1.22 hillshade()

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

from scott baker, ptisk library

19.10.1.23 incidence_angle()

```
def pysar._pysar_utilities.incidence_angle (
    atr,
    dimension = 2,
    print_message = 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

19.10.1.24 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.

19.10.1.25 make_triangle()

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

19.10.1.26 mode()

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

Find Mode (most common) item in the list

19.10.1.27 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

19.10.1.28 normalize_timeseries()

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

Normalize timeseries of 2D matrix in time domain

19.10.1.29 normalize_timeseries_old()

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

19.10.1.30 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

19.10.1.31 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]

19.10.1.32 radar2glob()

```
def pysar._pysar_utilities.radar2glob (
    az,
    rg,
    transFile = 'geomap*.trans',
    atr_rdr = dict(),
    print_message = True )
```

Convert radar coordinates into geo coordinates

Inputs:

rg/az - np.array, int, range/azimuth pixel number
 transFile - 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

19.10.1.33 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

19.10.1.34 range_resolution()

```
def pysar._pysar_utilities.range_resolution (
    atr,
    print_message = True )
```

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

19.10.1.35 spatial_average()

```
def pysar._pysar_utilities.spatial_average (
    File,
    mask = None,
    box = None,
    saveList = False )
```

Calculate Spatial Average.

Only non-nan pixel is considered.

Input:

```
File : string, path of input file
mask : 2D np.array, mask file
box : 4-tuple defining the left, upper, right, and lower pixel coordinate
saveList: bool, save (list of) mean value into text file
```

Output:

```
meanList : list for float, average value in space for each epoch of input file
```

Example:

```
meanList = spatial_average('coherence.h5')
meanList = spatial_average('coherence.h5', mask, saveList=True)
refList = spatial_average('unwrapIfgram.h5', box=(100,200,101,201))
```

19.10.1.36 stacking()

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

Stack multi-temporal dataset into one
equivalent to temporal sum

19.10.1.37 temporal_average()

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

Calculate temporal average.

19.10.1.38 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')
```

19.10.1.39 timeseries_inversion()

```
def pysar._pysar_utilities.timeseries_inversion (
    ifgramFile,
    timeseriesFile )
```

Implementation of the SBAS algorithm.

modified from sbas.py written by scott baker, 2012

Usage:

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

19.10.1.40 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
#####
```

19.10.1.41 timeseries_inversion_L1()

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

19.10.1.42 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.

19.10.1.43 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.

19.10.1.44 update_attribute_or_not()

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

Compare new attributes with existing ones

19.10.1.45 update_file()

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

Check whether to update outFile 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)

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):
```

19.10.1.46 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

19.10.1.47 which()

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

Test if executable exists

19.10.1.48 yymdd()

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

19.10.1.49 yymdd2YYYYMMDD()

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

19.10.1.50 yyyyymmdd()

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

19.11 pysar._readfile Namespace Reference

Functions

- def [read](#) (File, box=(), epoch=None)
- 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)
- def [attribute_isce2roipac](#) (xml_dict)
- def [read_float32](#) (File, box=None)
- def [read_complex_float32](#) (fname, byteorder=None, real_imag=False)
- def [read_real_float32](#) (fname, byteorder=None)
- def [read_complex_int16](#) (File, box=None, real_imag=False)
- def [read_dem](#) (File)
- def [read_real_int16](#) (File)
- def [read_flag](#) (File)
- 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']
- list [single_dataset_hdf5_file](#) = ['dem','mask','rmse','temporal_coherence', 'velocity']

19.11.1 Function Documentation

19.11.1.1 `attribute_gamma2roipac()`

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

Convert Gamma par attribute into ROI_PAC format

19.11.1.2 `attribute_isce2roipac()`

```
def pysar._readfile.attribute_isce2roipac (
    xml_dict )
```

Convert ISCE xml attribute into ROI_PAC format

19.11.1.3 `check_variable_name()`

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

19.11.1.4 `is_plot_attribute()`

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

19.11.1.5 read()

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

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', (100,1100, 500, 2500))
data, atr = read('timeseries.h5', (), '20101120')
data, atr = read('timeseries.h5', (100,1100, 500, 2500), '20101120')
az, atr = read('geomap*.trans', (), 'azimuth')
rg,az,atr = read('geomap*.trans')
```

19.11.1.6 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

19.11.1.7 read_complex_float32()

```
def pysar._readfile.read_complex_float32 (
    fname,
    byteorder = None,
    real_imag = 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
byteorder : str, optional, order of reading byte in the file
real_imag : 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)
```

19.11.1.8 read_complex_int16()

```
def pysar._readfile.read_complex_int16 (
    File,
    box = None,
    real_imag = 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))
```

19.11.1.9 read_dem()

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

Read real int 16 data matrix, i.e. ROI_PAC .dem file.

Input: roi_pac format dem file

Usage: dem, atr = read_real_int16('gsi10m_30m.dem')

19.11.1.10 read_flag()

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

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

19.11.1.11 read_float32()

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

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, ...,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))
```

19.11.1.12 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

19.11.1.13 read_GPS_USGS()

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

19.11.1.14 read_isce_xml()

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

Read ISCE .xml file input a python dictionary structure.

19.11.1.15 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))
```

19.11.1.16 read_real_float32()

```
def pysar._readfile.read_real_float32 (
    fname,
    byteorder = None )
```

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

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

byteorder : 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')
```

19.11.1.17 read_real_int16()

```
def pysar._readfile.read_real_int16 (
    File )
```

Same as read_dem() above

19.11.1.18 read_roipac_rsc()

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

Read ROI_PAC .rsc file into a python dictionary structure.

19.11.1.19 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, ':')
```

19.11.2 Variable Documentation

19.11.2.1 multi_dataset_hdf5_file

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

19.11.2.2 multi_group_hdf5_file

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

19.11.2.3 single_dataset_hdf5_file

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

19.12 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)

19.12.1 Function Documentation

19.12.1.1 `remove_data_multiple_surface()`

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

19.12.1.2 `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

19.12.1.3 `remove_surface()`

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

19.13 `pysar._sensor` Namespace Reference

Classes

- class [JERS](#)

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

19.14 `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)

19.14.1 Function Documentation

19.14.1.1 bin_variance()

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

19.14.1.2 get_distance()

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

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

19.14.1.3 get_lat_lon()

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

Get lat/lon of all pixels

19.14.1.4 sample_data()

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

19.14.1.5 `structure_function()`

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

19.15 `pysar._writefile` Namespace Reference

Functions

- def `write` (args)
- def `write_ropac_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)

19.15.1 Function Documentation

19.15.1.1 `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')
```

19.15.1.2 write_complex64()

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

Writes roi_pac .int data

19.15.1.3 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, ...

19.15.1.4 write_dem()

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

19.15.1.5 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)
```

19.15.1.6 write_real_float32()

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

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

19.15.1.7 write_real_int16()

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

19.15.1.8 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
```

19.16 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](#)

19.16.1 Function Documentation

19.16.1.1 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')
```


19.16.1.2 add_matrix()

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

Sum of 2 input matrix

19.16.1.3 cmdLineParse()

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

19.16.1.4 main()

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

19.16.2 Variable Documentation

19.16.2.1 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 '''
```

19.17 pysar.add_attribute Namespace Reference

Functions

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

19.17.1 Function Documentation

19.17.1.1 `main()`

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

19.17.1.2 `usage()`

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

19.18 `pysar.add_attribute_insarmaps` Namespace Reference

Classes

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

Functions

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

19.18.1 Function Documentation

19.18.1.1 `build_parser()`

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

19.18.1.2 `main()`

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

19.19 `pysar.asc_desc` Namespace Reference

Functions

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

Variables

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

19.19.1 Function Documentation

19.19.1.1 cmdLineParse()

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

19.19.1.2 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

19.19.1.3 main()

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

19.19.2 Variable Documentation

19.19.2.1 EXAMPLE

EXAMPLE

19.19.2.2 REFERENCE

REFERENCE

19.20 pysar.baseline_error Namespace Reference

Functions

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

19.20.1 Function Documentation

19.20.1.1 main()

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

19.20.1.2 to_percent()

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

19.20.1.3 usage()

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

19.21 pysar.baseline_trop Namespace Reference

Functions

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

19.21.1 Function Documentation

19.21.1.1 main()

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

19.21.1.2 to_percent()

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

19.21.1.3 usage()

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

19.22 pysar.coord_glob2radar Namespace Reference

Functions

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

19.22.1 Function Documentation

19.22.1.1 main()

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

19.22.1.2 usage()

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

19.23 pysar.coord_radar2glob Namespace Reference

Functions

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

19.23.1 Function Documentation

19.23.1.1 main()

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

19.23.1.2 usage()

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

19.24 pysar.correct_dem Namespace Reference

Functions

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

19.24.1 Function Documentation

19.24.1.1 main()

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

19.24.1.2 usage()

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

19.25 pysar.correlation_with_dem Namespace Reference

Functions

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

19.25.1 Function Documentation

19.25.1.1 main()

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

19.25.1.2 usage()

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

19.26 pysar.dem_error Namespace Reference

Functions

- def [read_template2inps](#) (template_file, inps=None)
- def [get_exclude_date](#) (inps, date_list_all)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

19.26.1 Function Documentation

19.26.1.1 cmdLineParse()

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

19.26.1.2 get_exclude_date()

```
def pysar.dem_error.get_exclude_date (
    inps,
    date_list_all )
```

19.26.1.3 main()

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

19.26.1.4 read_template2inps()

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

Read input template file into inps.ex_date

19.26.2 Variable Documentation

19.26.2.1 EXAMPLE

EXAMPLE

19.26.2.2 REFERENCE

REFERENCE

19.26.2.3 TEMPLATE

TEMPLATE

19.27 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)

19.27.1 Function Documentation

19.27.1.1 cmdLineParse()

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


19.27.1.2 diff_data()

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

data1 - data2

19.27.1.3 diff_file()

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

Subtraction/difference of two input files

19.27.1.4 main()

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

19.27.1.5 usage()

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

19.28 pysar.download_ecmwf Namespace Reference

Variables

- [start_date](#)
- [end_date](#)
- [hour](#)
- [step](#)
- [days](#)
- [dateListFile](#)
- [f](#)
- [date](#)
- [date_str](#)
- [tropCmd](#)
- [runFile](#)
- [maxJobNum](#)
- [jobCmd](#)

19.28.1 Variable Documentation**19.28.1.1 date**

date

19.28.1.2 date_str

date_str

19.28.1.3 dateListFile

dateListFile

19.28.1.4 days

days

19.28.1.5 end_date

end_date

19.28.1.6 f

f

19.28.1.7 hour

hour

19.28.1.8 jobCmd

jobCmd

19.28.1.9 maxJobNum

maxJobNum

19.28.1.10 runFile

runFile

19.28.1.11 start_date

start_date

19.28.1.12 step

step

19.28.1.13 tropCmd

tropCmd

19.29 pysar.epoch_coherence Namespace Reference

Functions

- def [epoch_coherence_file](#) (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

19.29.1 Function Documentation

19.29.1.1 cmdLineParse()

```
def pysar.epoch_coherence.cmdLineParse ( )
```

19.29.1.2 epoch_coherence_file()

```
def pysar.epoch_coherence.epoch_coherence_file (
    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 = epoch_coherence_file('timeseries_ECMWF_demErrInvResid_quadratic.h5')
```

19.29.1.3 main()

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

19.29.2 Variable Documentation

19.29.2.1 EXAMPLE

EXAMPLE

19.30 pysar.gamma_view Namespace Reference

Functions

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

19.30.1 Function Documentation

19.30.1.1 main()

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

19.30.1.2 usage()

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

19.31 pysar.generate_mask Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.31.1 Function Documentation

19.31.1.1 cmdLineParse()

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

19.31.1.2 main()

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

19.31.2 Variable Documentation

19.31.2.1 EXAMPLE

EXAMPLE

19.32 pysar.geocode Namespace Reference

Functions

- def [update_attribute4isce](#) (atr_rdr, inps, geo_data)
- def [geocode_attribute_with_geo_lut](#) (atr_rdr, atr_lut, print_msg=True)
- def [geocode_file_with_geo_lut](#) (fname, lut_file=None, method='nearest', fill_value=np.nan, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

19.32.1 Function Documentation

19.32.1.1 cmdLineParse()

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

19.32.1.2 geocode_attribute_with_geo_lut()

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

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.
```

19.32.1.3 geocode_file_with_geo_lut()

```
def pysar.geocode.geocode_file_with_geo_lut (
    fname,
    lut_file = None,
    method = 'nearest',
    fill_value = np.nan,
    fname_out = None )
```

Geocode file using ROI_PAC/Gamma lookup table file.

Related module: `scipy.interpolate.RegularGridInterpolator`

Inputs:

```
fname      : string, file to be geocoded
lut_file   : string, optional, lookup table file generated by ROIPAC or Gamma
              i.e. geomap_4rlks.trans          from ROI_PAC
              sim_150911-150922.UTM_TO_RDC     from Gamma
method     : string, optional, interpolation/resampling method, supporting nearest, linear
fill_value : value used for points outside of the interpolation domain.
              If None, values outside the domain are extrapolated.
fname_out  : string, optional, output geocoded filename
```

Output:

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

19.32.1.4 main()

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

19.32.1.5 update_attribute4isce()

```
def pysar.geocode.update_attribute4isce (
    atr_rdr,
    inps,
    geo_data )
```

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/lon - 1D np.array of lat/lon value
geo_data - 2D matrix, with shape info used.
```

Output:

```
atr - dict, attributes of output file in geo coord.
```

19.32.2 Variable Documentation

19.32.2.1 EXAMPLE

EXAMPLE

19.33 pysar.geocode_orig Namespace Reference

Functions

- def [update_attribute4isce](#) (atr_rdr, inps, geo_data)
- def [geocode_attribute_with_geo_lookup_table](#) (atr_rdr, atr_lut, print_message=True)
- def [geocode_file_with_geo_lookup_table](#) (fname, lookup_file=None, interp_method='nearest', fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

19.33.1 Function Documentation

19.33.1.1 cmdLineParse()

```
def pysar.geocode_orig.cmdLineParse ( )
```

19.33.1.2 geocode_attribute_with_geo_lookup_table()

```
def pysar.geocode_orig.geocode_attribute_with_geo_lookup_table (
    atr_rdr,
    atr_lut,
    print_message = True )
```

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_message : bool, print out message or not
```

Output:

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

19.33.1.3 geocode_file_with_geo_lookup_table()

```
def pysar.geocode_orig.geocode_file_with_geo_lookup_table (
    fname,
    lookup_file = None,
    interp_method = 'nearest',
    fname_out = None )
```

Geocode file using ROI_PAC/Gamma lookup table file.

Inputs:

```
fname          : string, file to be geocoded
lookup_file    : string, optional, lookup table file generated by ROIPAC 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, cubic
fname_out     : string, optional, output geocoded filename
```

Output:

```
fname_out
```

A faster way is as below:

<https://stackoverflow.com/questions/20915502/speedup-scipy-griddata-for-multiple-interpolations-between-two-in>

19.33.1.4 main()

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


19.33.1.5 update_attribute4isce()

```
def pysar.geocode_orig.update_attribute4isce (
    atr_rdr,
    inps,
    geo_data )
```

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/lon - 1D np.array of lat/lon value
geo_data - 2D matrix, with shape info used.
```

Output:

```
atr - dict, attributes of output file in geo coord.
```

19.33.2 Variable Documentation

19.33.2.1 EXAMPLE

EXAMPLE

19.34 pysar.ifgram_closure Namespace Reference

Functions

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

19.34.1 Function Documentation

19.34.1.1 main()

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

19.34.1.2 usage()

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

19.35 pysar.ifgram_inversion Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.35.1 Function Documentation

19.35.1.1 cmdLineParse()

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

19.35.1.2 main()

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

19.35.2 Variable Documentation

19.35.2.1 EXAMPLE

EXAMPLE

19.36 pysar.ifgram_reconstruction Namespace Reference

Functions

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

19.36.1 Function Documentation

19.36.1.1 main()

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

19.36.1.2 usage()

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

19.37 pysar.ifgram_simulation Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.37.1 Function Documentation

19.37.1.1 cmdLineParse()

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

19.37.1.2 main()

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

19.37.2 Variable Documentation

19.37.2.1 EXAMPLE

EXAMPLE

19.38 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](#)

19.38.1 Function Documentation

19.38.1.1 cmdLineParse()

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

19.38.1.2 data_operation()

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

Mathmatic operation of 2D matrix

19.38.1.3 file_operation()

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

Mathmathic operation of file

19.38.1.4 main()

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

19.38.2 Variable Documentation

19.38.2.1 EXAMPLE

EXAMPLE

19.39 pysar.incidence_angle Namespace Reference

Functions

- def `usage` ()
- def `main` (argv)

19.39.1 Function Documentation

19.39.1.1 `main()`

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

19.39.1.2 `usage()`

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

19.40 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)

19.40.1 Function Documentation

19.40.1.1 main()

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

19.40.1.2 print_attributes()

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

19.40.1.3 print_hdf5_structure()

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

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

19.40.1.4 print_timseries_date_info()

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

19.40.1.5 usage()

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

19.41 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)

19.41.1 Function Documentation

19.41.1.1 find_row_column()

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

19.41.1.2 main()

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

19.41.1.3 nearest()

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

19.41.1.4 readGPSfile()

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

19.41.1.5 usage()

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

19.42 pysar.insarmaps_query Namespace Reference

Classes

- class [BasicHTTP](#)

Functions

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

19.42.1 Function Documentation

19.42.1.1 build_parser()

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

19.42.1.2 buildURL()

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

19.42.1.3 main()

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

19.43 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](#)

19.43.1 Function Documentation

19.43.1.1 build_parser()

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


19.43.1.2 get_unavco_name()

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

19.43.1.3 main()

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

19.43.1.4 upload_insarmaps_metadata()

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

19.43.1.5 upload_json()

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

19.43.2 Variable Documentation

19.43.2.1 dbHost

dbHost

19.43.2.2 dbPassword

dbPassword

19.43.2.3 dbUsername

dbUsername

19.44 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`

19.44.1 Function Documentation

19.44.1.1 `l1()`

```
def pysar.l1.l1 (
    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.
```

19.44.1.2 `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.
```

19.44.1.3 `l1mosek()`

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

```
minimize   e'*v

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

19.44.1.4 l1mosek2()

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

minimize    e'*s + e'*t
subject to  P*u - q = s - t
           s, t >= 0
```

19.44.2 Variable Documentation

19.44.2.1 __MOSEK

```
__MOSEK [private]
```

19.44.2.2 task

```
task
```

19.44.2.3 x

```
x
```

19.45 pysar.load_data Namespace Reference

Functions

- def [auto_path_miami](#) (inps, template={})
 - Sub Functions #####.*
- def [mode](#) (thelist)
- def [check_file_size](#) (fileList, mode_width=None, mode_length=None)
- def [check_existed_hdf5_file](#) (roipacFileList, hdf5File)
- def [load_multi_group_hdf5](#) (fileType, fileList, hdf5File='unwraplgram.h5', extra_meta_dict=dict())
- def [load_single_dataset_hdf5](#) (file_type, infile, outfile, extra_meta_dict=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

- [EXAMPLE](#)

Usage #####.

- [TEMPLATE](#)

19.45.1 Function Documentation

19.45.1.1 auto_path_miami()

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

Sub Functions #####.

Auto File Path Setting for Geodesy Lab - University of Miami

19.45.1.2 check_existed_hdf5_file()

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

Check file list with existed hdf5 file

19.45.1.3 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.

19.45.1.4 cmdLineParse()

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

19.45.1.5 copy_file()

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

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

19.45.1.6 load_data_from_template()

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

Load dataset for PySAR time series using input template

19.45.1.7 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)
    timeseries_dir : directory of time series analysis, e.g. KujuAlosAT422F650/PYSAR
    insar_processor: 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))
```

19.45.1.8 load_multi_group_hdf5()

```
def pysar.load_data.load_multi_group_hdf5 (
    fileType,
    fileList,
    hdf5File = 'unwrapIfgram.h5',
    extra_meta_dict = 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
hdf5File : string, file name/path of the multi-group hdf5 PySAR file
extra_meta_dict : dict, extra attribute dictionary
```

Outputs:

```
hdf5File : output hdf5 file name
fileList : list of string, files newly added
```

19.45.1.9 load_single_dataset_hdf5()

```
def pysar.load_data.load_single_dataset_hdf5 (
    file_type,
    infile,
    outfile,
    extra_meta_dict = 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
extra_meta_dict : dict, extra attributes to output file
```

Output:

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

19.45.1.10 main()

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

Main Function #####.

19.45.1.11 mode()

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

Find Mode (most common) item in the list

19.45.2 Variable Documentation

19.45.2.1 EXAMPLE

EXAMPLE

Usage #####.

19.45.2.2 TEMPLATE

TEMPLATE

19.46 pysar.load_data_bak Namespace Reference

Functions

- def [auto_path_miami](#) (inps, template={})
Sub Functions #####.
- def [mode](#) (thelist)
- def [check_file_size](#) (fileList, mode_width=None, mode_length=None)
- def [check_existed_hdf5_file](#) (roipacFileList, hdf5File)
- def [roipac2multi_group_hdf5](#) (fileType, fileList, hdf5File='unwrapfgram.h5', extra_meta_dict=dict())
- def [roipac_nonzero_mask](#) (unwFileList, maskFile='mask.h5')
- def [roipac2single_dataset_hdf5](#) (file_type, infile, outfile, extra_meta_dict=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

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

19.46.1 Function Documentation

19.46.1.1 auto_path_miami()

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

Sub Functions #####.

Auto File Path Setting for Geodesy Lab - University of Miami

19.46.1.2 check_existed_hdf5_file()

```
def pysar.load_data_bak.check_existed_hdf5_file (
    roipacFileList,
    hdf5File )
```

Check file list with existed hdf5 file

19.46.1.3 check_file_size()

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

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

19.46.1.4 cmdLineParse()

```
def pysar.load_data_bak.cmdLineParse ( )
```

19.46.1.5 copy_file()

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

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

19.46.1.6 load_data_from_template()

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

Load dataset for PySAR time series using input template

19.46.1.7 load_file()

```
def pysar.load_data_bak.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)
    timeseries_dir : directory of time series analysis, e.g. KujuAlosAT422F650/PYSAR
    insar_processor: 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))
```


19.46.1.8 main()

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

Main Function #####.

19.46.1.9 mode()

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

Find Mode (most common) item in the list

19.46.1.10 roipac2multi_group_hdf5()

```
def pysar.load_data_bak.roipac2multi_group_hdf5 (
    fileType,
    fileList,
    hdf5File = 'unwrapIfgram.h5',
    extra_meta_dict = 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
hdf5File : string, file name/path of the multi-group hdf5 PySAR file
extra_meta_dict : dict, extra attribute dictionary
```

Outputs:

```
hdf5File : output hdf5 file name
fileList : list of string, files newly added
```

19.46.1.11 roipac2single_dataset_hdf5()

```
def pysar.load_data_bak.roipac2single_dataset_hdf5 (
    file_type,
    infile,
    outfile,
    extra_meta_dict = 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
extra_meta_dict : dict, extra attributes to output file
```

Output:

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

19.46.1.12 roipac_nonzero_mask()

```
def pysar.load_data_bak.roipac_nonzero_mask (
    unwFileList,
    maskFile = 'mask.h5' )
```

Generate mask for non-zero amplitude pixel of ROI_PAC .unw file list.

19.46.2 Variable Documentation

19.46.2.1 EXAMPLE

EXAMPLE

Usage #####.

19.46.2.2 TEMPLATE

TEMPLATE

19.47 pysar.load_dem Namespace Reference

Variables

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

19.47.1 Variable Documentation

19.47.1.1 amp

amp

19.47.1.2 compression

compression

19.47.1.3 data

data

19.47.1.4 dem

dem

19.47.1.5 demFile

demFile

19.47.1.6 demRsc

demRsc

19.47.1.7 dset

dset

19.47.1.8 ext

ext

19.47.1.9 group

group

19.47.1.10 h5

h5

19.47.1.11 outName

outName

19.48 pysar.lod Namespace Reference

Functions

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

19.48.1 Function Documentation

19.48.1.1 correct_lod_file()

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

19.48.1.2 main()

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

19.48.1.3 usage()

```
def pysar.lod.usage ( )
```

19.49 pysar.look_angle Namespace Reference

Functions

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

19.49.1 Function Documentation

19.49.1.1 main()

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

19.49.1.2 usage()

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

19.50 pysar.los2enu Namespace Reference

Functions

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

19.50.1 Function Documentation

19.50.1.1 main()

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

19.50.1.2 usage()

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

19.51 pysar.mask Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.51.1 Function Documentation

19.51.1.1 cmdLineParse()

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

19.51.1.2 main()

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

19.51.1.3 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
```

19.51.1.4 mask_matrix()

```
def pysar.mask.mask_matrix (
    data_mat,
    mask_mat )
```

mask a 2D matrix data with mask

19.51.1.5 update_mask()

```
def pysar.mask.update_mask (
    mask,
    inps_dict = None )
```

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

19.51.2 Variable Documentation

19.51.2.1 EXAMPLE

EXAMPLE

19.52 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](#)

19.52.1 Function Documentation

19.52.1.1 cmdLineParse()

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

19.52.1.2 corners()

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

Get corners coordinate.

19.52.1.3 main()

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

19.52.1.4 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.

19.52.1.5 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 overlapping.

19.52.1.6 nearest()

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

find nearest neighbour

19.52.2 Variable Documentation

19.52.2.1 EXAMPLE

EXAMPLE

19.53 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](#)

19.53.1 Function Documentation

19.53.1.1 cmdLineParse()

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

19.53.1.2 main()

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

Main Function #####.

19.53.1.3 manual_select_pairs_to_remove()

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

Manually select interferograms to remove

19.53.1.4 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
result to a new file
outFile - string, output file name

Output:

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

19.53.1.5 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
```

19.53.1.6 read_template2inps()

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

Read input template options into Namespace inps

19.53.1.7 reset_pairs()

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

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

19.53.2 Variable Documentation**19.53.2.1 EXAMPLE**

EXAMPLE

Usage #####.

19.53.2.2 TEMPLATE

TEMPLATE

19.54 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](#)

19.54.1 Function Documentation

19.54.1.1 `check_st_in_box()`

```
def pysar.multi_transect.check_st_in_box (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```

19.54.1.2 `check_st_in_box2()`

```
def pysar.multi_transect.check_st_in_box2 (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```

19.54.1.3 `dist_point_from_line()`

```
def pysar.multi_transect.dist_point_from_line (
    m,
    c,
    x,
    y,
    dx,
    dy )
```

19.54.1.4 `dms2d()`

```
def pysar.multi_transect.dms2d (
    Coord )
```

19.54.1.5 find_row_column()

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

19.54.1.6 get_intersect()

```
def pysar.multi_transect.get_intersect (
    m,
    c,
    x,
    y )
```

19.54.1.7 get_lat_lon()

```
def pysar.multi_transect.get_lat_lon (
    h5file )
```

19.54.1.8 get_start_end_point()

```
def pysar.multi_transect.get_start_end_point (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L,
    dx,
    dy )
```

19.54.1.9 get_transect()

```
def pysar.multi_transect.get_transect (
    z,
    x0,
    y0,
    x1,
    y1 )
```

19.54.1.10 `gps_to_LOS()`

```
def pysar.multi_transect.gps_to_LOS (
    Ve,
    Vn,
    theta,
    heading )
```

19.54.1.11 `line()`

```
def pysar.multi_transect.line (
    x0,
    y0,
    x1,
    y1 )
```

19.54.1.12 `main()`

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

19.54.1.13 `nanmean()`

```
def pysar.multi_transect.nanmean (
    data,
    args )
```

19.54.1.14 `nanstd()`

```
def pysar.multi_transect.nanstd (
    data,
    args )
```

19.54.1.15 `nearest()`

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


19.54.1.16 onclick()

```
def pysar.multi_transect.onclick (
    event )
```

19.54.1.17 point_on_line_with_distance_from_beginning()

```
def pysar.multi_transect.point_on_line_with_distance_from_beginning (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L )
```

19.54.1.18 point_with_distance_from_line()

```
def pysar.multi_transect.point_with_distance_from_line (
    Xf0,
    Yf0,
    Xf1,
    Yf1,
    L )
```

19.54.1.19 read_fault_coords()

```
def pysar.multi_transect.read_fault_coords (
    Fault_coord_file,
    Dp )
```

19.54.1.20 readGPSfile()

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

19.54.1.21 redGPSfile()

```
def pysar.multi_transect.redGPSfile (
    gpsFile )
```

19.54.1.22 redGPSfile_cmm4()

```
def pysar.multi_transect.redGPSfile_cmm4 (
    gpsFile )
```

19.54.1.23 usage()

```
def pysar.multi_transect.usage ( )
```

19.54.2 Variable Documentation**19.54.2.1 alpha**

alpha

19.54.2.2 avgInSAR

avgInSAR

19.54.2.3 ax

ax

19.54.2.4 axes

axes

19.54.2.5 axes2

axes2

19.54.2.6 axis

axis

19.54.2.7 c

c

19.54.2.8 c_prof_edge

c_prof_edge

19.54.2.9 cf

cf

19.54.2.10 check_result

check_result

19.54.2.11 check_result2

check_result2

19.54.2.12 cid

cid

19.54.2.13 color

color

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

To plot the Fault location on the profile try:

19.54.2.14 D

D

19.54.2.15 dataset

dataset

19.54.2.16 df0_km

df0_km

19.54.2.17 dg

dg

19.54.2.18 DistGPS

DistGPS

19.54.2.19 dp

dp

19.54.2.20 dset

dset

19.54.2.21 dx

dx

19.54.2.22 DX

DX

19.54.2.23 dy

dy

19.54.2.24 DY

DY

19.54.2.25 Fault_lat

Fault_lat

19.54.2.26 Fault_lon

Fault_lon

19.54.2.27 FaultCoords

FaultCoords

19.54.2.28 FaultLine

FaultLine

19.54.2.29 fig

fig

19.54.2.30 fig2

fig2

19.54.2.31 figName

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)

19.54.2.32 fileExtension

fileExtension

19.54.2.33 fileName

fileName

19.54.2.34 fontsize

fontsize

19.54.2.35 GPS

GPS

19.54.2.36 GPS_in_bound

GPS_in_bound

19.54.2.37 GPS_in_bound_st

GPS_in_bound_st

19.54.2.38 GPS_lat

GPS_lat

19.54.2.39 GPS_lon

GPS_lon

19.54.2.40 GPS_station

GPS_station

19.54.2.41 gpsFile

gpsFile

19.54.2.42 gpsLOS

gpsLOS

19.54.2.43 gpsLOS_ref

gpsLOS_ref

19.54.2.44 GPSx

GPSx

19.54.2.45 GPSxx

GPSxx

19.54.2.46 GPSy

GPSy

19.54.2.47 GPSyy

GPSyy

19.54.2.48 gx

gx

19.54.2.49 gy

gy

19.54.2.50 h5file_theta

h5file_theta

19.54.2.51 hbound

hbound

19.54.2.52 heading

heading

19.54.2.53 idx

idx

19.54.2.54 IDX

IDX

19.54.2.55 idxRef

idxRef

19.54.2.56 IDXref

IDXref

19.54.2.57 IDY

IDY

19.54.2.58 IDYref

IDYref

19.54.2.59 insarData

insarData

19.54.2.60 lat

lat

19.54.2.61 Lat

Lat

19.54.2.62 Lat0

Lat0

19.54.2.63 Lat1

Lat1

19.54.2.64 lat_all

lat_all

19.54.2.65 lat_step

lat_step

19.54.2.66 lat_transect

lat_transect

19.54.2.67 lbound

lbound

lower and higher bounds for displaying the profile

19.54.2.68 Length

Length

19.54.2.69 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
```

19.54.2.70 linewidth

linewidth

19.54.2.71 lon

lon

19.54.2.72 Lon

Lon

19.54.2.73 Lon0

Lon0

19.54.2.74 Lon1

Lon1

19.54.2.75 lon_all

lon_all

19.54.2.76 lon_step

lon_step

19.54.2.77 lon_transect

lon_transect

19.54.2.78 m

m

19.54.2.79 m1

m1

19.54.2.80 m_prof_edge

m_prof_edge

19.54.2.81 matFile

matFile

19.54.2.82 mf

mf

19.54.2.83 mfc

mfc

19.54.2.84 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')

19.54.2.85 NoInSAR

NoInSAR

19.54.2.86 nrows

nrows

19.54.2.87 Num_profiles

Num_profiles

19.54.2.88 Se

Se

19.54.2.89 Sn

Sn

19.54.2.90 Stations

Stations

19.54.2.91 stationsList

stationsList

19.54.2.92 stdInSAR

stdInSAR

19.54.2.93 theta

theta

19.54.2.94 transect

transect

19.54.2.95 transect_lat

transect_lat

19.54.2.96 transect_lon

transect_lon

19.54.2.97 unitVec

unitVec

19.54.2.98 Ve

Ve

19.54.2.99 Vn

Vn

19.54.2.100 Width

Width

19.54.2.101 x

x

19.54.2.102 x0

x0

19.54.2.103 X0

X0

19.54.2.104 x1

x1

19.54.2.105 X1

X1

19.54.2.106 xc

xc

19.54.2.107 Xf0

Xf0

19.54.2.108 Xf1

Xf1

19.54.2.109 xlim

xlim

19.54.2.110 XX0

XX0

19.54.2.111 XX1

XX1

19.54.2.112 y

y

19.54.2.113 y0

y0

19.54.2.114 Y0

Y0

19.54.2.115 y1

y1

19.54.2.116 Y1

Y1

19.54.2.117 yc

yc

19.54.2.118 Yf0

Yf0

19.54.2.119 Yf1

Yf1

19.54.2.120 ylim

ylim

19.54.2.121 YY0

YY0

19.54.2.122 YY1

YY1

19.54.2.123 zi

zi

19.55 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_message=True)
- def [multilook_file](#) (infile, lks_y, lks_x, outfile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

19.55.1 Function Documentation

19.55.1.1 cmdLineParse()

```
def pysar.multilook.cmdLineParse ( )
```

19.55.1.2 main()

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

19.55.1.3 multilook_attribute()

```
def pysar.multilook.multilook_attribute (
    atr_dict,
    lks_y,
    lks_x,
    print_message = True )
```

19.55.1.4 multilook_file()

```
def pysar.multilook.multilook_file (
    infile,
    lks_y,
    lks_x,
    outfile = None )
```

19.55.1.5 multilook_matrix()

```
def pysar.multilook.multilook_matrix (
    matrix,
    lks_y,
    lks_x )
```

Sub Functions #####.

19.55.2 Variable Documentation

19.55.2.1 EXAMPLE

EXAMPLE

19.56 pysar.perp_baseline Namespace Reference

Functions

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

19.56.1 Function Documentation

19.56.1.1 main()

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

19.56.1.2 usage()

```
def pysar.perp_baseline.usage ( )
```

19.57 pysar.plot_atmDrop Namespace Reference

Variables

- [projectList](#)
- [projectDir](#)
- [numProject](#)
- [fig](#)
- [figsize](#)
- [ax1](#)
- [ax2](#)
- [offset](#)
- [fl](#)
- *Read txt file.*
- [lines](#)
- [lineNum](#)
- [dateList6](#)
- [meanList](#)
- [pixList](#)
- [line_s](#)
- [dateList](#)
- [dates](#)
- [datevector](#)
- [idxMean](#)
- [key](#)
- [idxPix](#)
- [sc1](#)
- *Plot.*
- [c](#)
- [s](#)
- [alpha](#)
- [vmin](#)
- [vmax](#)
- [sc2](#)
- [fontsize](#)
- [cbar](#)
- [bbox_inches](#)
- [transparent](#)

19.57.1 Variable Documentation

19.57.1.1 `alpha`

`alpha`

19.57.1.2 `ax1`

`ax1`

19.57.1.3 `ax2`

`ax2`

19.57.1.4 `bbox_inches`

`bbox_inches`

19.57.1.5 `c`

`c`

19.57.1.6 `cbar`

`cbar`

19.57.1.7 `dateList`

`dateList`

19.57.1.8 `dateList6`

`dateList6`

19.57.1.9 dates

dates

19.57.1.10 datevector

datevector

19.57.1.11 fig

fig

19.57.1.12 figsize

figsize

19.57.1.13 fl

fl

Read txt file.

19.57.1.14 fontsize

fontsize

19.57.1.15 idxMean

idxMean

19.57.1.16 idxPix

idxPix

19.57.1.17 key

key

19.57.1.18 line_s

line_s

19.57.1.19 lineNum

lineNum

19.57.1.20 lines

lines

19.57.1.21 meanList

meanList

19.57.1.22 numProject

numProject

19.57.1.23 offset

offset

19.57.1.24 pixList

pixList

19.57.1.25 projectDir

projectDir

19.57.1.26 projectList

projectList

19.57.1.27 s

s

19.57.1.28 sc1

sc1

Plot.

19.57.1.29 sc2

sc2

19.57.1.30 transparent

transparent

19.57.1.31 vmax

vmax

19.57.1.32 vmin

vmin

19.58 pysar.plot_network Namespace Reference**Functions**

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

Main Function #####.

Variables

- [BL_LIST](#)
- [DATE12_LIST](#)
- [EXAMPLE](#)

19.58.1 Function Documentation

19.58.1.1 cmdLineParse()

```
def pysar.plot_network.cmdLineParse ( )
```

19.58.1.2 main()

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

Main Function #####.

19.58.2 Variable Documentation

19.58.2.1 BL_LIST

BL_LIST

19.58.2.2 DATE12_LIST

DATE12_LIST

19.58.2.3 EXAMPLE

EXAMPLE

19.59 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](#)

19.59.1 Function Documentation

19.59.1.1 cmdLineParse()

```
def pysar.prep_gamma.cmdLineParse ( )
```

19.59.1.2 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

19.59.1.3 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
 For example, it read input file, sim_150911-150922.hgt_sim,
 find its associated par file, sim_150911-150922.diff_par, read it, and
 convert to ROI_PAC style and write it to an rsc file, sim_150911-150922.hgt_sim.rsc

19.59.1.4 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. /PopoSLT143TxD/diff_filt_HDR_130118-130129_4rlks.amp

Output:

atr : dict, Attributes dictionary

19.59.1.5 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

19.59.1.6 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

19.59.1.7 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]

19.59.1.8 main()

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

19.59.2 Variable Documentation

19.59.2.1 DESCRIPTION

DESCRIPTION

19.59.2.2 EXAMPLE

EXAMPLE

19.60 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](#)

19.60.1 Function Documentation

19.60.1.1 baselineTimeseries()

```
def pysar.prep_isce.baselineTimeseries (
    baselineDir )
```

19.60.1.2 cmdLineParse()

```
def pysar.prep_isce.cmdLineParse (
    iargs = None )
```

19.60.1.3 createParser()

```
def pysar.prep_isce.createParser ( )
```

Command line parser.

19.60.1.4 extractIsceMetadata()

```
def pysar.prep_isce.extractIsceMetadata (
    xmlFile )
```

19.60.1.5 main()

```
def pysar.prep_isce.main (
    iargs = None )
```

19.60.1.6 prepare_geometry()

```
def pysar.prep_isce.prepare_geometry (
    geometryDir )
```

19.60.1.7 prepare_stack()

```
def pysar.prep_isce.prepare_stack (
    inputDir,
    filePattern,
    metadata,
    baselineDict )
```

19.60.1.8 read_baseline()

```
def pysar.prep_isce.read_baseline (
    baselineFile )
```

19.60.1.9 write_rsc()

```
def pysar.prep_isce.write_rsc (
    isceFile,
    dates,
    metadata,
    baselineDict )
```

19.60.2 Variable Documentation**19.60.2.1 GDAL2NUMPY_DATATYPE**

GDAL2NUMPY_DATATYPE

19.61 pysar.prep_roipac Namespace Reference**Functions**

- def [extract_attribute](#) (fname)
 Sub Functions #####
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

19.61.1 Function Documentation**19.61.1.1 cmdLineParse()**

```
def pysar.prep_roipac.cmdLineParse ( )
```

19.61.1.2 `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
```

19.61.1.3 `main()`

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

19.61.2 Variable Documentation

19.61.2.1 DESCRIPTION

DESCRIPTION

19.61.2.2 EXAMPLE

EXAMPLE

19.62 pysar.pysarApp Namespace Reference

Functions

- def [check_subset_file](#) (File, inps_dict, outFile=None, overwrite=False)
- def [check_geocode_file](#) (geomapFile, File, outFile=None)
- def [subset_dataset](#) (inps, geo_box4geo, pix_box4rdr)
- def [create_subset_dataset](#) (inps, pix_box=None, geo_box=None)
- def [multilook_dataset](#) (inps, lks_y=None, lks_x=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

19.62.1 Function Documentation

19.62.1.1 `check_geocode_file()`

```
def pysar.pysarApp.check_geocode_file (
    geomapFile,
    File,
    outFile = None )
```

Geocode input file or use existed geocoded file.

19.62.1.2 `check_subset_file()`

```
def pysar.pysarApp.check_subset_file (
    File,
    inps_dict,
    outFile = None,
    overwrite = False )
```

Subset input file or use existed subseted file.

19.62.1.3 `cmdLineParse()`

```
def pysar.pysarApp.cmdLineParse ( )
```

19.62.1.4 `create_subset_dataset()`

```
def pysar.pysarApp.create_subset_dataset (
    inps,
    pix_box = None,
    geo_box = None )
```

Create/prepare subset of datasets in different folder for time series analysis.

For dataset (unwrapped interferograms) in radar coord, only support subset in row/col or y/x

For dataset (unwrapped interferograms) in geo coord, lalo has higher priority than yx, if both are specified.

19.62.1.5 main()

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

19.62.1.6 multilook_dataset()

```
def pysar.pysarApp.multilook_dataset (
    inps,
    lks_y = None,
    lks_x = None )
```

Create a multilooked dataset

19.62.1.7 subset_dataset()

```
def pysar.pysarApp.subset_dataset (
    inps,
    geo_box4geo,
    pix_box4rdr )
```

Subset all file within dataset

with geo_box4geo for all geocoded file and pix_box4rdr for all files in radar coord.

Inputs:

inps - Namespace with all files that needs to be subseted and work_dir
geo_box4geo - tuple of 4 float, subset range in lat/lon for files in geo coord
pix_box4rdr - tuple of 4 int, subset range in y/x for files in radar coord

Output:

inps - Namespace, update file name/path info

19.62.2 Variable Documentation

19.62.2.1 EXAMPLE

EXAMPLE

19.62.2.2 LOGO

LOGO

19.62.2.3 TEMPLATE

TEMPLATE

19.62.2.4 UM_FILE_STRUCT

UM_FILE_STRUCT

19.63 pysar.quality_map Namespace Reference

Functions

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

19.63.1 Function Documentation

19.63.1.1 main()

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

19.63.1.2 usage()

```
def pysar.quality_map.usage ( )
```

19.64 pysar.range_distance Namespace Reference

Functions

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

19.64.1 Function Documentation

19.64.1.1 main()

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


19.64.1.2 usage()

```
def pysar.range_distance.usage ( )
```

19.65 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](#)

19.65.1 Function Documentation

19.65.1.1 cmdLineParse()

```
def pysar.reference_epoch.cmdLineParse ( )
```

19.65.1.2 main()

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

19.65.1.3 read_template2inps()

```
def pysar.reference_epoch.read_template2inps (
    templateFile,
    inps = None )
```

Update inps with options from templateFile

19.65.1.4 `ref_date_attribute()`

```
def pysar.reference_epoch.ref_date_attribute (
    atr_in,
    ref_date,
    date_list )
```

Update attribute dictionary for reference date

19.65.1.5 `ref_date_file()`

```
def pysar.reference_epoch.ref_date_file (
    inFile,
    ref_date,
    outFile = None )
```

Change input file reference date to a different one.

19.65.2 Variable Documentation

19.65.2.1 EXAMPLE

EXAMPLE

19.65.2.2 TEMPLATE

TEMPLATE

19.66 `pysar.remove_plane` Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.66.1 Function Documentation

19.66.1.1 cmdLineParse()

```
def pysar.remove_plane.cmdLineParse ( )
```

19.66.1.2 main()

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

19.66.2 Variable Documentation

19.66.2.1 EXAMPLE

EXAMPLE

19.67 pysar.rewrap Namespace Reference

Functions

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

19.67.1 Function Documentation

19.67.1.1 main()

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

19.67.1.2 rewrap()

```
def pysar.rewrap.rewrap (
    unw )
```

19.67.1.3 usage()

```
def pysar.rewrap.usage ( )
```

19.68 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](#)

19.68.1 Function Documentation

19.68.1.1 cmdLineParse()

```
def pysar.save_gmt.cmdLineParse ( )
```

19.68.1.2 get_geo_lat_lon()

```
def pysar.save_gmt.get_geo_lat_lon (
    atr )
```

19.68.1.3 main()

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

19.68.1.4 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

19.68.2 Variable Documentation

19.68.2.1 EXAMPLE

EXAMPLE

19.69 pysar.save_kml Namespace Reference

Functions

- def [write_kmz_file](#) (data, atr, out_name_base, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

19.69.1 Function Documentation

19.69.1.1 cmdLineParse()

```
def pysar.save_kml.cmdLineParse ( )
```

19.69.1.2 main()

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

19.69.1.3 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)
```

19.69.2 Variable Documentation

19.69.2.1 EXAMPLE

EXAMPLE

19.70 pysar.save_mat Namespace Reference

Functions

- def [usage](#) ()
- def [yyyymmdd2years](#) (date)
- def [main](#) (argv)

19.70.1 Function Documentation

19.70.1.1 main()

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

19.70.1.2 usage()

```
def pysar.save_mat.usage ( )
```

19.70.1.3 yyyymmdd2years()

```
def pysar.save_mat.yyyymmdd2years (  
    date )
```

19.71 pysar.save_mat_orig Namespace Reference

Functions

- def [usage](#) ()
- def [yyyymmdd2years](#) (date)
- def [main](#) (argv)

19.71.1 Function Documentation

19.71.1.1 main()

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

19.71.1.2 usage()

```
def pysar.save_mat_orig.usage ( )
```

19.71.1.3 yyyymmdd2years()

```
def pysar.save_mat_orig.yyyymmdd2years (
    date )
```

19.72 pysar.save_roipac Namespace Reference

Functions

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

19.72.1 Function Documentation

19.72.1.1 main()

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

19.72.1.2 usage()

```
def pysar.save_roipac.usage ( )
```

19.73 pysar.save_unavco Namespace Reference

Functions

- def [get_mission_name](#) (meta_dict)
- def [metadata_pysar2unavco](#) (pysar_meta_dict, dateList)
- def [get_unavco_filename](#) (timeseriesFile)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [INT_ZERO](#)
- [FLOAT_ZERO](#)
- [CPX_ZERO](#)
- [EXAMPLE](#)

19.73.1 Function Documentation

19.73.1.1 cmdLineParse()

```
def pysar.save_unavco.cmdLineParse ( )
```

19.73.1.2 get_mission_name()

```
def pysar.save_unavco.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.

19.73.1.3 get_unavco_filename()

```
def pysar.save_unavco.get_unavco_filename (
    timeseriesFile )
```

Get output file name of UNAVCO InSAR Archive

19.73.1.4 main()

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

19.73.1.5 metadata_pysar2unavco()

```
def pysar.save_unavco.metadata_pysar2unavco (
    pysar_meta_dict,
    dateList )
```

19.73.2 Variable Documentation

19.73.2.1 CPX_ZERO

CPX_ZERO

19.73.2.2 EXAMPLE

EXAMPLE

19.73.2.3 FLOAT_ZERO

FLOAT_ZERO

19.73.2.4 INT_ZERO

INT_ZERO

19.74 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_message=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](#)

19.74.1 Function Documentation

19.74.1.1 cmdLineParse()

```
def pysar.seed_data.cmdLineParse ( )
```

19.74.1.2 main()

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

Main Function #####.

19.74.1.3 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
```

19.74.1.4 nearest()

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

Sub Functions #####.

19.74.1.5 print_warning()

```
def pysar.seed_data.print_warning (
    next_method )
```

19.74.1.6 random_select_reference_yx()

```
def pysar.seed_data.random_select_reference_yx (
    data_mat,
    print_message = True )
```

19.74.1.7 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

19.74.1.8 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

19.74.1.9 remove_reference_pixel()

```
def pysar.seed_data.remove_reference_pixel (
    File )
```

Remove reference pixel info from input file

19.74.1.10 seed_attributes()

```
def pysar.seed_data.seed_attributes (
    atr_in,
    x,
    y )
```

19.74.1.11 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

19.74.1.12 seed_file_reference_value()

```
def pysar.seed_data.seed_file_reference_value (
    File,
    outName,
    refList,
    ref_y = '',
    ref_x = '' )
```

19.74.1.13 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

19.74.2 Variable Documentation**19.74.2.1 EXAMPLE**

EXAMPLE

19.74.2.2 NOTE

NOTE

19.74.2.3 TEMPLATE

TEMPLATE

Usage #####.

19.75 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](#)
- [METHOD](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

19.75.1 Function Documentation

19.75.1.1 cmdLineParse()

```
def pysar.select_network.cmdLineParse ( )
```

19.75.1.2 log()

```
def pysar.select_network.log (  
    msg )
```

Log function writen by Falk

19.75.1.3 main()

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

19.75.1.4 project_name2sensor()

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

19.75.1.5 read_template2inps()

```
def pysar.select_network.read_template2inps (
    templateFile,
    inps = None )
```

Read network options from template file into Namespace variable inps

19.75.2 Variable Documentation

19.75.2.1 EXAMPLE

EXAMPLE

19.75.2.2 METHOD

METHOD

19.75.2.3 REFERENCE

REFERENCE

19.75.2.4 sar_sensor_list

sar_sensor_list

19.75.2.5 TEMPLATE

TEMPLATE

19.76 pysar.spatial_average Namespace Reference

Functions

- def [cmdLineParse](#) ()
 - def [main](#) (argv)
- Main Function #####.

Variables

- [EXAMPLE](#)

Usage #####.

19.76.1 Function Documentation

19.76.1.1 cmdLineParse()

```
def pysar.spatial_average.cmdLineParse ( )
```

19.76.1.2 main()

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

Main Function #####.

19.76.2 Variable Documentation

19.76.2.1 EXAMPLE

EXAMPLE

Usage #####.

19.77 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](#)

19.77.1 Function Documentation

19.77.1.1 cmdLineParse()

```
def pysar.spatial_filter.cmdLineParse ( )
```

19.77.1.2 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.
```

19.77.1.3 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_gaussain, it's sigma in float
```

Output:

```
fname_out : string, optional, output file name/path
```

19.77.1.4 main()

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

19.77.2 Variable Documentation

19.77.2.1 EXAMPLE

EXAMPLE

19.78 pysar.subset Namespace Reference

Functions

- def [coord_geo2radar](#) (geoCoord, atr, coordType)
 - Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')*
- def [coord_radar2geo](#) (radarCoord, 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_message=True)
- def [get_coverage_box](#) (atr)
- def [read_subset_template2box](#) (templateFile)
- def [bbox_geo2radar](#) (geo_box, atr_rdr=dict(), transFile='geomap *.trans')
- def [bbox_radar2geo](#) (pix_box, atr_rdr=dict(), transFile='geomap *.trans')
- 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](#)

19.78.1 Function Documentation

19.78.1.1 bbox_geo2radar()

```
def pysar.subset.bbox_geo2radar (
    geo_box,
    atr_rdr = dict(),
    transFile = 'geomap*.trans' )
```

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
transFile  - 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.
```

19.78.1.2 bbox_radar2geo()

```
def pysar.subset.bbox_radar2geo (
    pix_box,
    atr_rdr = dict(),
    transFile = 'geomap*.trans' )
```

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
transFile  - 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
```

19.78.1.3 box_geo2pixel()

```
def pysar.subset.box_geo2pixel (
    geo_box,
    meta_dict )
```

Convert geo_box to pixel_box

19.78.1.4 box_pixel2geo()

```
def pysar.subset.box_pixel2geo (
    pixel_box,
    meta_dict )
```

Convert pixel_box to geo_box

19.78.1.5 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
```

19.78.1.6 cmdLineParse()

```
def pysar.subset.cmdLineParse ( )
```

19.78.1.7 coord_geo2radar()

```
def pysar.subset.coord_geo2radar (
    geoCoord,
    atr,
    coordType )
```

Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')

19.78.1.8 coord_radar2geo()

```
def pysar.subset.coord_radar2geo (
    radarCoord,
    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')

19.78.1.9 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)

19.78.1.10 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

19.78.1.11 main()

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

19.78.1.12 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.

19.78.1.13 subset_attribute()

```
def pysar.subset.subset_attribute (
    atr_dict,
    subset_box,
    print_message = 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
```

19.78.1.14 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

19.78.1.15 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, subset 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.
```

19.78.1.16 subset_file_list()

```
def pysar.subset.subset_file_list (
    fileList,
    inps )
```

Subset file list

19.78.1.17 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)
```

19.78.2 Variable Documentation

19.78.2.1 EXAMPLE

EXAMPLE

19.79 pysar.sum_epochs Namespace Reference

Functions

- def `usage()`
- def `main(argv)`

19.79.1 Function Documentation

19.79.1.1 `main()`

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

19.79.1.2 `usage()`

```
def pysar.sum_epochs.usage ( )
```

19.80 pysar.temporal_average Namespace Reference

Functions

- def `usage()`
Usage #####
- def `main(argv)`
Main Function #####

19.80.1 Function Documentation

19.80.1.1 main()

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

Main Function #####.

19.80.1.2 usage()

```
def pysar.temporal_average.usage ( )
```

Usage #####.

19.81 pysar.temporal_coherence Namespace Reference

Functions

- def [temporal_coherence](#) (timeseriesFile, ifgramFile)
- def [usage](#) ()
- def [main](#) (argv)

Variables

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

19.81.1 Function Documentation

19.81.1.1 main()

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

19.81.1.2 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
```

19.81.1.3 `usage()`

```
def pysar.temporal_coherence.usage ( )
```

19.81.2 Variable Documentation

19.81.2.1 DESCRIPTION

DESCRIPTION

19.81.2.2 EXAMPLE

EXAMPLE

19.81.2.3 REFERENCE

REFERENCE

19.81.2.4 USAGE

USAGE

19.82 `pysar.temporal_derivative` Namespace Reference

Functions

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

19.82.1 Function Documentation

19.82.1.1 `main()`

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


19.82.1.2 usage()

```
def pysar.temporal_derivative.usage ( )
```

19.83 pysar.temporal_filter Namespace Reference

Functions

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

Variables

- [EXAMPLE](#)

19.83.1 Function Documentation

19.83.1.1 cmdLineParse()

```
def pysar.temporal_filter.cmdLineParse ( )
```

19.83.1.2 main()

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

19.83.2 Variable Documentation

19.83.2.1 EXAMPLE

EXAMPLE

19.84 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](#)

19.84.1 Function Documentation

19.84.1.1 cmdLineParse()

```
def pysar.timeseries2velocity.cmdLineParse ( )
```

19.84.1.2 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

19.84.1.3 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')

19.84.1.4 main()

```
def pysar.timeseries2velocity.main (
    argv )
```

19.84.1.5 read_template2inps()

```
def pysar.timeseries2velocity.read_template2inps (
    template_file,
    inps = None )
```

Read input template file into inps.ex_date

19.84.2 Variable Documentation

19.84.2.1 DROP_DATE_TXT

DROP_DATE_TXT

19.84.2.2 EXAMPLE

EXAMPLE

19.84.2.3 TEMPLATE

TEMPLATE

19.85 pysar.timeseries_rms Namespace Reference

Functions

- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

19.85.1 Function Documentation

19.85.1.1 cmdLineParse()

```
def pysar.timeseries_rms.cmdLineParse ( )
```

19.85.1.2 main()

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

19.85.1.3 read_template2inps()

```
def pysar.timeseries_rms.read_template2inps (
    templateFile,
    inps = None )
```

Update inps with pysar.residualRms.* option from templateFile

19.85.2 Variable Documentation**19.85.2.1 EXAMPLE**

EXAMPLE

19.85.2.2 TEMPLATE

TEMPLATE

19.86 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](#)

19.86.1 Function Documentation

19.86.1.1 cmdLineParse()

```
def pysar.transect.cmdLineParse ( )
```

19.86.1.2 get_scale_from_disp_unit()

```
def pysar.transect.get_scale_from_disp_unit (
    disp_unit,
    data_unit )
```

19.86.1.3 main()

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

Main #####.

19.86.1.4 manual_select_start_end_point()

```
def pysar.transect.manual_select_start_end_point (
    File )
```

Manual Select Start/End Point in display figure.

19.86.1.5 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

19.86.1.6 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]

19.86.1.7 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
```

19.86.1.8 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])
```

19.86.2 Variable Documentation

19.86.2.1 EXAMPLE

EXAMPLE

19.87 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) minorLocator = 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](#)

19.87.1 Function Documentation

19.87.1.1 check_st_in_box()

```
def pysar.transect_legacy.check_st_in_box (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```

19.87.1.2 check_st_in_box2()

```
def pysar.transect_legacy.check_st_in_box2 (
    x,
    y,
    x0,
    y0,
    x1,
    y1,
    X0,
    Y0,
    X1,
    Y1 )
```

19.87.1.3 dist_point_from_line()

```
def pysar.transect_legacy.dist_point_from_line (
    m,
    c,
    x,
    y,
    dx,
    dy )
```

19.87.1.4 dms2d()

```
def pysar.transect_legacy.dms2d (
    Coord )
```

19.87.1.5 find_row_column()

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

19.87.1.6 get_intersect()

```
def pysar.transect_legacy.get_intersect (
    m,
    c,
    x,
    y )
```

19.87.1.7 get_lat_lon()

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

19.87.1.8 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.

19.87.1.9 gps_to_LOS()

```
def pysar.transect_legacy.gps_to_LOS (
    Ve,
    Vn,
    theta,
    heading )
```

19.87.1.10 line()

```
def pysar.transect_legacy.line (
    x0,
    y0,
    x1,
    y1 )
```

19.87.1.11 main()

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

19.87.1.12 nanmean()

```
def pysar.transect_legacy.nanmean (
    data,
    args )
```

19.87.1.13 nanstd()

```
def pysar.transect_legacy.nanstd (
    data,
    args )
```

19.87.1.14 nearest()

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

19.87.1.15 onclick()

```
def pysar.transect_legacy.onclick (
    event )
```

19.87.1.16 readGPSfile()

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

19.87.1.17 redGPSfile()

```
def pysar.transect_legacy.redGPSfile (
    gpsFile )
```

19.87.1.18 redGPSfile_cmm4()

```
def pysar.transect_legacy.redGPSfile_cmm4 (
    gpsFile )
```

19.87.1.19 Usage()

```
def pysar.transect_legacy.Usage ( )
```

19.87.2 Variable Documentation**19.87.2.1 alpha**

alpha

19.87.2.2 avglnSAR

avglnSAR

19.87.2.3 ax

ax

19.87.2.4 axes

axes

19.87.2.5 axes2

axes2

19.87.2.6 axis

axis

19.87.2.7 c

c

19.87.2.8 c_prof_edge

c_prof_edge

19.87.2.9 cf

cf

19.87.2.10 check_result

check_result

19.87.2.11 check_result2

check_result2

19.87.2.12 cid

cid

19.87.2.13 color

color

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

To plot the Fault location on the profile.

19.87.2.14 D

D

19.87.2.15 dataset

dataset

19.87.2.16 df0

df0

19.87.2.17 df0_km

df0_km

19.87.2.18 df1

df1

19.87.2.19 dg

dg

19.87.2.20 DistGPS

DistGPS

19.87.2.21 dset

dset

19.87.2.22 dx

dx

19.87.2.23 DX

DX

19.87.2.24 dy

dy

19.87.2.25 DY

DY

19.87.2.26 earth_radius

earth_radius

19.87.2.27 fault_loc

fault_loc

19.87.2.28 FaultLine

FaultLine

19.87.2.29 fig

fig

19.87.2.30 fig2

fig2

19.87.2.31 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)

19.87.2.32 fileExtension

fileExtension

19.87.2.33 fileName

fileName

19.87.2.34 fontsize

fontsize

19.87.2.35 GPS

GPS

19.87.2.36 GPS_in_bound

GPS_in_bound

19.87.2.37 GPS_in_bound_st

GPS_in_bound_st

19.87.2.38 GPS_lat

GPS_lat

19.87.2.39 GPS_lon

GPS_lon

19.87.2.40 GPS_station

GPS_station

19.87.2.41 gpsFile

gpsFile

19.87.2.42 gpsLOS

gpsLOS

19.87.2.43 gpsLOS_ref

gpsLOS_ref

19.87.2.44 GPSx

GPSx

19.87.2.45 GPSxx

GPSxx

19.87.2.46 GPSy

GPSy

19.87.2.47 GPSyy

GPSyy

19.87.2.48 gx

gx

19.87.2.49 gy

gy

19.87.2.50 h5file_theta

h5file_theta

19.87.2.51 hbound

hbound

19.87.2.52 heading

heading

19.87.2.53 idx

idx

19.87.2.54 IDX

IDX

19.87.2.55 idxRef

idxRef

19.87.2.56 IDXref

IDXref

19.87.2.57 IDY

IDY

19.87.2.58 IDYref

IDYref

19.87.2.59 Info_aboutFault

Info_aboutFault

19.87.2.60 insarData

insarData

19.87.2.61 Lat

Lat

19.87.2.62 lat

lat

19.87.2.63 lat_all

lat_all

19.87.2.64 lat_step

lat_step

19.87.2.65 lat_transect

lat_transect

19.87.2.66 lbound

`lbound`

lower and higher bounds for displaying the profile

19.87.2.67 length

`length`

19.87.2.68 Length

`Length`

19.87.2.69 linewidth

`linewidth`

19.87.2.70 Lon

`Lon`

19.87.2.71 lon

`lon`

19.87.2.72 lon_all

`lon_all`

19.87.2.73 lon_step

`lon_step`

19.87.2.74 lon_transect

lon_transect

19.87.2.75 m

m

19.87.2.76 m1

m1

19.87.2.77 m_prof_edge

m_prof_edge

19.87.2.78 matFile

matFile

19.87.2.79 mf

mf

19.87.2.80 mfc

mfc

19.87.2.81 mp

mp

19.87.2.82 ms

ms

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

19.87.2.83 NoInSAR

NoInSAR

19.87.2.84 nrows

nrows

19.87.2.85 Se

Se

19.87.2.86 Sn

Sn

19.87.2.87 Stations

Stations

19.87.2.88 stationsList

stationsList

19.87.2.89 stdlnSAR

stdlnSAR

19.87.2.90 **theta**

theta

19.87.2.91 **transect**

transect

19.87.2.92 **transect_lat**

transect_lat

19.87.2.93 **transect_lon**

transect_lon

19.87.2.94 **unitVec**

unitVec

19.87.2.95 **Ve**

Ve

19.87.2.96 **Vn**

Vn

19.87.2.97 **Width**

Width

19.87.2.98 **x**

x

19.87.2.99 x0

x0

19.87.2.100 X0

X0

19.87.2.101 x1

x1

19.87.2.102 X1

X1

19.87.2.103 xc

xc

19.87.2.104 XX0

XX0

19.87.2.105 XX1

XX1

19.87.2.106 y

y

19.87.2.107 y0

y0

19.87.2.108 Y0

Y0

19.87.2.109 y1

y1

19.87.2.110 Y1

Y1

19.87.2.111 yc

yc

19.87.2.112 ylim

ylim

19.87.2.113 YY0

YY0

19.87.2.114 YY1

YY1

19.87.2.115 zi

zi

19.88 pysar.tropcor_phase_elevation Namespace Reference**Functions**

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

Variables

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

19.88.1 Function Documentation

19.88.1.1 cmdLineParse()

```
def pysar.tropcor_phase_elevation.cmdLineParse ( )
```

19.88.1.2 main()

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

19.88.2 Variable Documentation

19.88.2.1 EXAMPLE

EXAMPLE

19.88.2.2 REFERENCE

REFERENCE

19.89 pysar.tropcor_pyaps Namespace Reference

Functions

- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')
- def [get_delay](#) (grib_file, atr, inps_dict)
- def [dload_grib](#) (date_list, hour, grib_source='ECMWF', weather_dir='./')
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

19.89.1 Function Documentation

19.89.1.1 `closest_weather_product_time()`

```
def pysar.tropcor_pyaps.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

19.89.1.2 `cmdLineParse()`

```
def pysar.tropcor_pyaps.cmdLineParse ( )
```

19.89.1.3 `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

19.89.1.4 `get_delay()`

```
def pysar.tropcor_pyaps.get_delay (
    grib_file,
    atr,
    inps_dict )
```

19.89.1.5 main()

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

19.89.2 Variable Documentation

19.89.2.1 EXAMPLE

EXAMPLE

19.89.2.2 REFERENCE

REFERENCE

19.89.2.3 TEMPLATE

TEMPLATE

19.90 pysar.tropcor_pyaps_orig Namespace Reference

Functions

- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')
- def [get_delay](#) (grib_file, atr, inps_dict)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

19.90.1 Function Documentation

19.90.1.1 closest_weather_product_time()

```
def pysar.tropcor_pyaps_orig.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

19.90.1.2 cmdLineParse()

```
def pysar.tropcor_pyaps_orig.cmdLineParse ( )
```

19.90.1.3 get_delay()

```
def pysar.tropcor_pyaps_orig.get_delay (
    grib_file,
    atr,
    inps_dict )
```

19.90.1.4 main()

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

19.90.2 Variable Documentation

19.90.2.1 EXAMPLE

EXAMPLE

19.90.2.2 REFERENCE

REFERENCE

19.90.2.3 TEMPLATE

TEMPLATE

19.91 pysar.tsviewer Namespace Reference

Functions

- def [read_timeseries_yx](#) (timeseries_file, y, x)
- 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](#)
- [length](#)
- [width](#)
- [ullon](#)
- [ullat](#)
- [lon_step](#)
- [lat_step](#)
- [lrlon](#)
- [lrlat](#)
- [y](#)
- [x](#)
- [yx](#)
- [ref_yx](#)
- [unit_fac](#)
- [flip_ud](#)
- [left_lr](#)
- [mask](#)
- [d_v](#)

- [ref_d_v](#)
- [data_lim](#)
- [ylim](#)
- [fig_v](#)

Fig 1 - Cumulative Displacement Map.

- [ax_v](#)
- [img](#)
- [cmap](#)
- [colormap](#)
- [clim](#)
- [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.

19.91.1 Function Documentation

19.91.1.1 cmdLineParse()

```
def pysar.tsviewer.cmdLineParse ( )
```

19.91.1.2 format_coord()

```
def pysar.tsviewer.format_coord (
    x,
    y )
```

19.91.1.3 plot_timeseries_errorbar()

```
def pysar.tsviewer.plot_timeseries_errorbar (
    ax,
    dis_ts,
    inps )
```

19.91.1.4 plot_timeseries_event()

```
def pysar.tsviewer.plot_timeseries_event (
    event )
```

Event function to get y/x from button press

19.91.1.5 plot_timeseries_scatter()

```
def pysar.tsviewer.plot_timeseries_scatter (
    ax,
    dis_ts,
    inps )
```

19.91.1.6 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

19.91.1.7 read_timeseries_yx()

```
def pysar.tsviewer.read_timeseries_yx (
    timeseries_file,
    y,
    x )
```

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

19.91.1.8 time_slider_update()

```
def pysar.tsviewer.time_slider_update (
    val )
```

Update Displacement Map using Slider

19.91.1.9 update_timeseries()

```
def pysar.tsviewer.update_timeseries (
    y,
    x )
```

Plot point time series displacement at pixel [y, x]

19.91.2 Variable Documentation

19.91.2.1 atr

atr

19.91.2.2 ax_time

ax_time

19.91.2.3 ax_ts

ax_ts

19.91.2.4 ax_v

ax_v

19.91.2.5 axisbg

axisbg

19.91.2.6 bbox_inches

bbox_inches

19.91.2.7 cbar

cbar

19.91.2.8 cid

```
cid = fig_v.canvas.mpl_connect('button_press_event', plot_timeseries_event)
```

Final linking of the canvas to the plots.

19.91.2.9 clim

clim

19.91.2.10 cmap

cmap

19.91.2.11 colormap

colormap

19.91.2.12 d_ts

d_ts

19.91.2.13 d_v

d_v

19.91.2.14 data_lim

data_lim

19.91.2.15 date_num

date_num

19.91.2.16 dateList

dateList

19.91.2.17 dates

dates

19.91.2.18 delimiter

string delimiter = [header_info](#))

19.91.2.19 dpi

dpi

19.91.2.20 dtype

dtype

19.91.2.21 e_ts

e_ts

19.91.2.22 ecolor

ecolor

19.91.2.23 error_file

error_file

19.91.2.24 error_fileContent

error_fileContent

19.91.2.25 error_ts

error_ts

19.91.2.26 ex_date

ex_date

19.91.2.27 ex_date_list

ex_date_list

19.91.2.28 ex_dates

ex_dates

19.91.2.29 ex_error_ts

ex_error_ts

19.91.2.30 ex_idx_list

ex_idx_list

19.91.2.31 EXAMPLE

EXAMPLE

19.91.2.32 facecolor

facecolor

19.91.2.33 fig_base

fig_base

Output.

19.91.2.34 fig_ts

fig_ts

Fig 2 - Time Series Displacement - Point.

19.91.2.35 fig_v

fig_v

Fig 1 - Cumulative Displacement Map.

19.91.2.36 figsize

figsize

19.91.2.37 flip_ud

`flip_ud`

19.91.2.38 fmt

`fmt`

19.91.2.39 format_coord

`format_coord`

19.91.2.40 h5

`h5`

19.91.2.41 header_info

`header_info`

19.91.2.42 img

`img`

19.91.2.43 inps

`inps`

Actual code.

19.91.2.44 input_ex_date

`input_ex_date`

19.91.2.45 k

k

19.91.2.46 lat

lat

19.91.2.47 lat_step

lat_step

19.91.2.48 left_lr

left_lr

19.91.2.49 length

length

19.91.2.50 lon

lon

19.91.2.51 lon_step

lon_step

19.91.2.52 lrlat

lrlat

19.91.2.53 lrlon

lrlon

19.91.2.54 markeredgecolor

markeredgecolor

19.91.2.55 mask

mask

19.91.2.56 ms

ms

19.91.2.57 orientation

orientation

19.91.2.58 outName

```
string outName = inps.fig_base+'_ts.pdf'
```

19.91.2.59 ref_d_v

ref_d_v

19.91.2.60 ref_yx

ref_yx

19.91.2.61 tims

tims

19.91.2.62 transparent

transparent

19.91.2.63 True

True

19.91.2.64 tslider

tslider

19.91.2.65 ullat

ullat

19.91.2.66 ullon

ullon

19.91.2.67 unit_fac

unit_fac

19.91.2.68 valinit

valinit

19.91.2.69 width

width

19.91.2.70 x

x

19.91.2.71 y

y

19.91.2.72 ylim

ylim

19.91.2.73 yticks

yticks

19.91.2.74 yx

yx

19.92 pysar.unavco2insarmaps Namespace Reference

Functions

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

19.92.1 Function Documentation

19.92.1.1 build_parser()

```
def pysar.unavco2insarmaps.build_parser ( )
```

19.92.1.2 get_H5_filename()

```
def pysar.unavco2insarmaps.get_H5_filename (
    path )
```

19.92.1.3 main()

```
def pysar.unavco2insarmaps.main ( )
```

19.93 pysar.unavco2json_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, dataset_keys, json_path, folder_name)
- def [make_json_file](#) (chunk_num, points, dataset_keys, json_path, folder_name)
- def [build_parser](#) ()
- def [main](#) ()

Variables

- dictionary [needed_attributes](#)

19.93.1 Function Documentation

19.93.1.1 build_parser()

```
def pysar.unavco2json_mbtiles.build_parser ( )
```

19.93.1.2 convert_data()

```
def pysar.unavco2json_mbtiles.convert_data (
    attributes,
    decimal_dates,
    timeseries_datasets,
    dataset_keys,
    json_path,
    folder_name )
```

19.93.1.3 get_date()

```
def pysar.unavco2json_mbtiles.get_date (
    date_string )
```

19.93.1.4 get_decimal_date()

```
def pysar.unavco2json_mbtiles.get_decimal_date (
    d )
```

19.93.1.5 main()

```
def pysar.unavco2json_mbtiles.main ( )
```

19.93.1.6 make_json_file()

```
def pysar.unavco2json_mbtiles.make_json_file (
    chunk_num,
    points,
    dataset_keys,
    json_path,
    folder_name )
```

19.93.1.7 region_name_from_project_name()

```
def pysar.unavco2json_mbtiles.region_name_from_project_name (
    project_name )
```

19.93.1.8 serialize_dictionary()

```
def pysar.unavco2json_mbtiles.serialize_dictionary (
    dictionary,
    fileName )
```

19.93.2 Variable Documentation

19.93.2.1 needed_attributes

dictionary needed_attributes

Initial value:

```
1 = {
2     "prf", "first_date", "mission", "WIDTH", "X_STEP", "processing_software",
3     "wavelength", "processing_type", "beam_swath", "Y_FIRST", "look_direction",
4     "flight_direction", "last_frame", "post_processing_method", "min_baseline_perp"
5     "unwrap_method", "relative_orbit", "beam_mode", "FILE_LENGTH", "max_baseline_perp",
6     "X_FIRST", "atmos_correct_method", "last_date", "first_frame", "frame", "Y_STEP", "history",
7     "scene_footprint", "data_footprint", "downloadUnavcoUrl", "referencePdfUrl", "areaName", "referenceText"
8 }
```

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

Variables

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

19.94.1 Function Documentation

19.94.1.1 [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, lied as: x_ref, x, x_ref, x

Output:

data : 2D np.array, phase corrected matrix

19.94.1.2 [cmdLineParse\(\)](#)

```
def pysar.unwrap_error.cmdLineParse ( )
```

19.94.1.3 [main\(\)](#)

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

19.94.1.4 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')
```

19.94.1.5 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')
```

19.94.2 Variable Documentation

19.94.2.1 DESCRIPTION

string DESCRIPTION

19.94.2.2 EXAMPLE

string EXAMPLE

Initial value:

```
1 = '''example:
2 Phase Closure:
3  unwrap_error.py  Seeded_unwrapIfgram.h5  mask.h5
4 Bridging:
5  unwrap_error.py  Seeded_unwrapIfgram.h5  mask.h5  -t ShikokuT417F650_690AlosA.template
6  unwrap_error.py  Seeded_unwrapIfgram.h5  mask.h5  -x 283 305 -y 1177 1247
7  unwrap_error.py  Seeded_081018_090118.unw  mask_all.h5 -x 283 305 -y 1177 1247 --ramp quadratic
8 '''
```

19.94.2.3 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 '''
```

19.95 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)
- 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

- string [EXAMPLE](#)
- string [PLOT_TEMPLATE](#)

19.95.1 Function Documentation

19.95.1.1 add_inner_title()

```
def pysar.view.add_inner_title (
    ax,
    title,
    loc,
    size = None,
    kwargs )
```

19.95.1.2 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')
```

19.95.1.3 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

19.95.1.4 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
```

19.95.1.5 check_colormap_input()

```
def pysar.view.check_colormap_input (
    atr_dict,
    colormap = None )
```

19.95.1.6 check_multilook_input()

```
def pysar.view.check_multilook_input (
    pixel_box,
    row_num,
    col_num )
```

19.95.1.7 cmdLineParse()

```
def pysar.view.cmdLineParse (
    argv )
```

19.95.1.8 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

19.95.1.9 main()

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

Main Function #####.

19.95.1.10 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)
```

19.95.1.11 plot_dem_yx()

```
def pysar.view.plot_dem_yx (
    ax,
    dem,
    inps_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)
```

19.95.1.12 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()
```

19.95.1.13 round_to_1()

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

Return the most significant digit of input number

19.95.1.14 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

19.95.1.15 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

19.95.1.16 update_matrix_with_plot_inps()

```
def pysar.view.update_matrix_with_plot_inps (
    data,
    meta_dict,
    inps )
```

19.95.1.17 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

19.95.1.18 update_plot_inps_with_meta_dict()

```
def pysar.view.update_plot_inps_with_meta_dict (
    inps,
    meta_dict )
```

19.95.2 Variable Documentation

19.95.2.1 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 '''
```

19.95.2.2 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 '''
```

19.96 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](#)

19.96.1 Function Documentation

19.96.1.1 cmdLineParse()

```
def troposphere_uncertainty.cmdLineParse ( )
```

19.96.1.2 download()

```
def troposphere_uncertainty.download (
    inps )
```

19.96.1.3 estimate_seasonal()

```
def troposphere_uncertainty.estimate_seasonal (
    inps )
```

19.96.1.4 main()

```
def troposphere_uncertainty.main (
    argv )
```

19.96.1.5 statistics()

```
def troposphere_uncertainty.statistics (
    inps )
```

19.96.1.6 velocity_uncertainty()

```
def troposphere_uncertainty.velocity_uncertainty (
    relative_std_file,
    inps )
```

19.96.1.7 velocity_uncertainty_vs_distance()

```
def troposphere_uncertainty.velocity_uncertainty_vs_distance (
    inps )
```

19.96.2 Variable Documentation

19.96.2.1 EXAMPLE

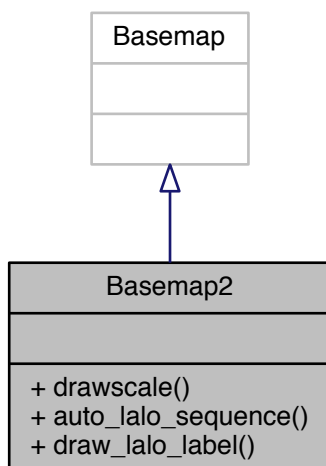
EXAMPLE

20 Class Documentation

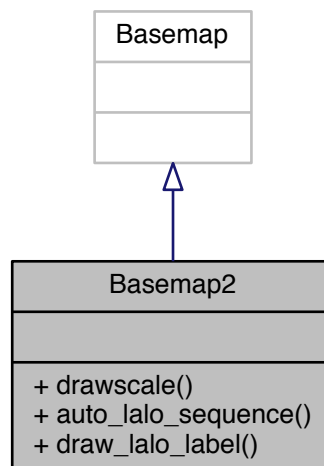
20.1 Basemap2 Class Reference

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)
- def [auto_lalo_sequence](#) (self, geo_box, max_tick_num=4, step_candidate=[1])
- def [draw_lalo_label](#) (self, geo_box, ax=None, labels=[1, font_size=12])

20.1.1 Detailed Description

Class #####.

20.1.2 Member Function Documentation

20.1.2.1 auto_lalo_sequence()

```
def auto_lalo_sequence (
    self,
    geo_box,
    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)
```

20.1.2.2 draw_lalo_label()

```
def draw_lalo_label (
    self,
    geo_box,
    ax = None,
    labels = [1,
    font_size = 12 )
```

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)
```

20.1.2.3 drawscale()

```
def drawscale (
    self,
    lat_c,
    lon_c,
    distance,
    ax = None,
    font_size = 12,
    yoffset = None )
```

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)
```

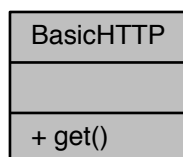
ref_link: <http://matplotliblib.1069221.n5.nabble.com/basemap-scalebar-td14133.html>

The documentation for this class was generated from the following file:

- [view.py](#)

20.2 BasicHTTP Class Reference

Collaboration diagram for BasicHTTP:



Static Public Member Functions

- def [get](#) (url)

20.2.1 Detailed Description

20.2.2 Member Function Documentation

20.2.2.1 get()

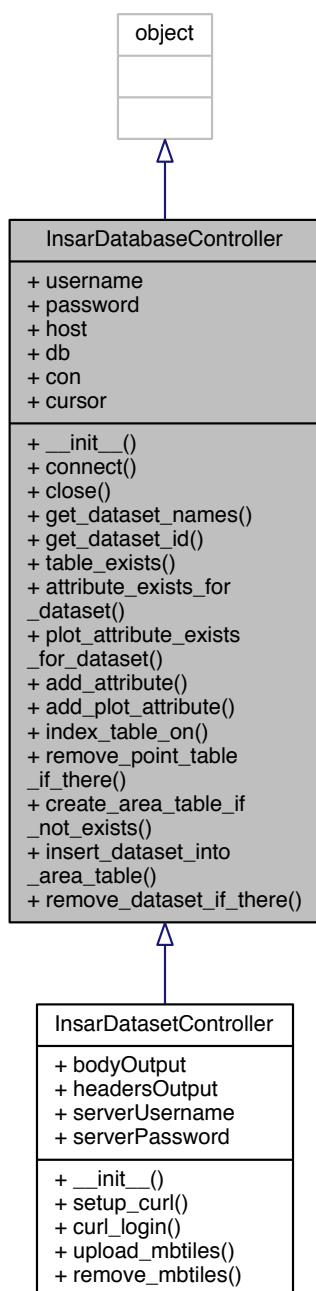
```
def get (
    url ) [static]
```

The documentation for this class was generated from the following file:

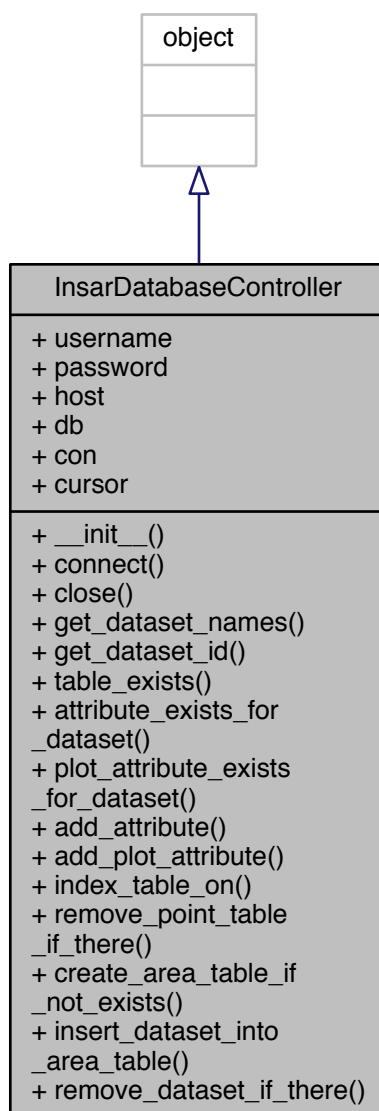
- [insarmaps_query.py](#)

20.3 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 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 [remove_point_table_if_there](#) (self, unavco_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](#)

20.3.1 Detailed Description

20.3.2 Constructor & Destructor Documentation

20.3.2.1 `__init__()`

```
def __init__ (
    self,
    username,
    password,
    host,
    db )
```

20.3.3 Member Function Documentation

20.3.3.1 `add_attribute()`

```
def add_attribute (
    self,
    dataset,
    attributekey,
    attributevalue )
```

20.3.3.2 add_plot_attribute()

```
def add_plot_attribute (
    self,
    dataset,
    attributekey,
    plotAttributeJSON )
```

20.3.3.3 attribute_exists_for_dataset()

```
def attribute_exists_for_dataset (
    self,
    dataset,
    attributekey )
```

20.3.3.4 close()

```
def close (
    self )
```

20.3.3.5 connect()

```
def connect (
    self )
```

20.3.3.6 create_area_table_if_not_exists()

```
def create_area_table_if_not_exists (
    self )
```

20.3.3.7 get_dataset_id()

```
def get_dataset_id (
    self,
    dataset )
```

20.3.3.8 get_dataset_names()

```
def get_dataset_names (
    self )
```

20.3.3.9 index_table_on()

```
def index_table_on (
    self,
    table,
    on,
    index_name )
```

20.3.3.10 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 )
```

20.3.3.11 plot_attribute_exists_for_dataset()

```
def plot_attribute_exists_for_dataset (
    self,
    dataset,
    attributekey )
```

20.3.3.12 remove_dataset_if_there()

```
def remove_dataset_if_there (
    self,
    unavco_name )
```

20.3.3.13 remove_point_table_if_there()

```
def remove_point_table_if_there (
    self,
    unavco_name )
```


20.3.3.14 table_exists()

```
def table_exists (
    self,
    table )
```

20.3.4 Member Data Documentation

20.3.4.1 con

con

20.3.4.2 cursor

cursor

20.3.4.3 db

db

20.3.4.4 host

host

20.3.4.5 password

password

20.3.4.6 username

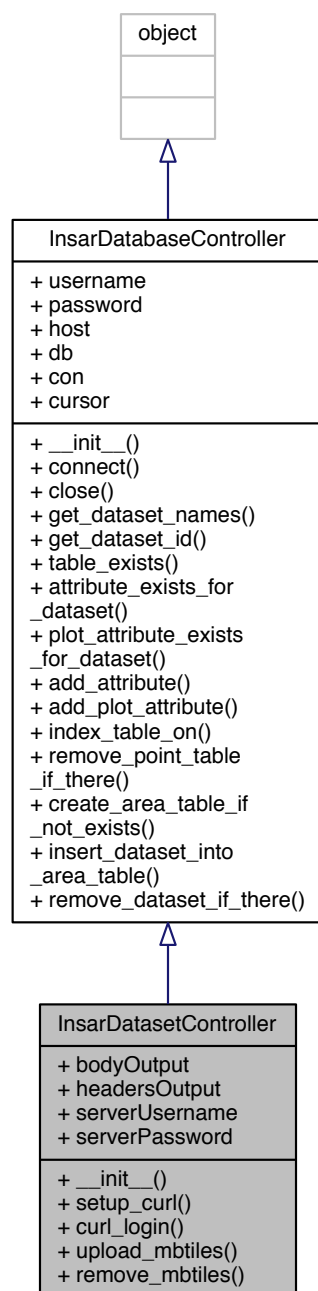
username

The documentation for this class was generated from the following file:

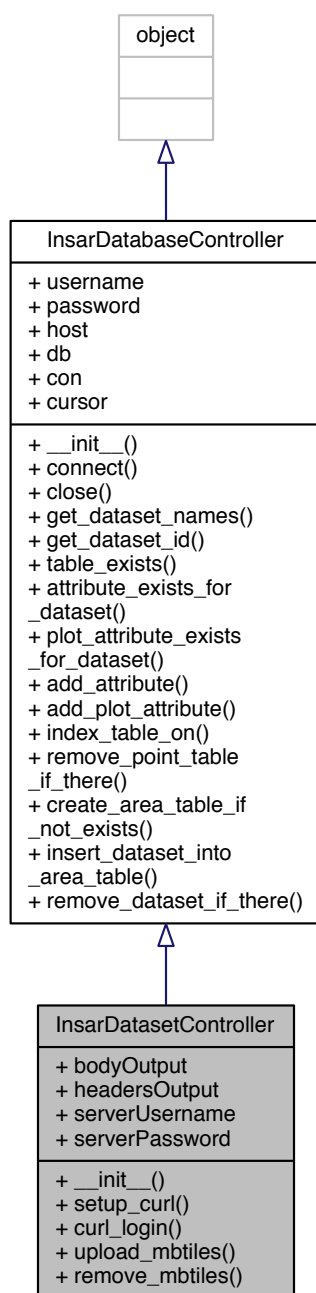
- [add_attribute_insarmaps.py](#)

20.4 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_mbtiles (self, fileName)`
- `def remove_mbtiles (self, fileName)`

Public Attributes

- [bodyOutput](#)
- [headersOutput](#)
- [serverUsername](#)
- [serverPassword](#)

20.4.1 Detailed Description

20.4.2 Constructor & Destructor Documentation

20.4.2.1 `__init__()`

```
def __init__ (
    self,
    username,
    password,
    host,
    db,
    serverUsername,
    serverPassword )
```

20.4.3 Member Function Documentation

20.4.3.1 `curl_login()`

```
def curl_login (
    self,
    username,
    password )
```

20.4.3.2 `remove_mbtiles()`

```
def remove_mbtiles (
    self,
    fileName )
```

20.4.3.3 `setup_curl()`

```
def setup_curl (
    self )
```

20.4.3.4 upload_mbtiles()

```
def upload_mbtiles (
    self,
    fileName )
```

20.4.4 Member Data Documentation

20.4.4.1 bodyOutput

bodyOutput

20.4.4.2 headersOutput

headersOutput

20.4.4.3 serverPassword

serverPassword

20.4.4.4 serverUsername

serverUsername

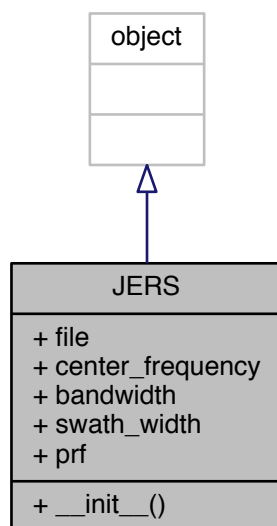
The documentation for this class was generated from the following file:

- [add_attribute_insarmaps.py](#)

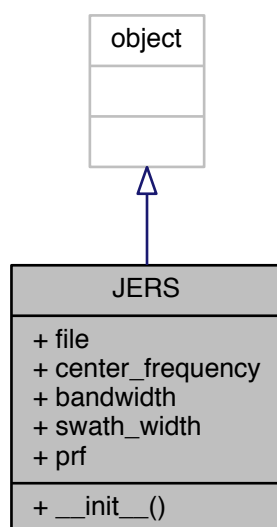
20.5 JERS Class Reference

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`

20.5.1 Detailed Description

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

Recommended Usage: import `pysar._sensor` as sensor

20.5.2 Constructor & Destructor Documentation

20.5.2.1 `__init__()`

```
def __init__ (
    self,
    file = None )
```

20.5.3 Member Data Documentation

20.5.3.1 `bandwidth`

`bandwidth`

20.5.3.2 `center_frequency`

`center_frequency`

20.5.3.3 `file`

`file`

20.5.3.4 prf

prf

20.5.3.5 swath_width

swath_width

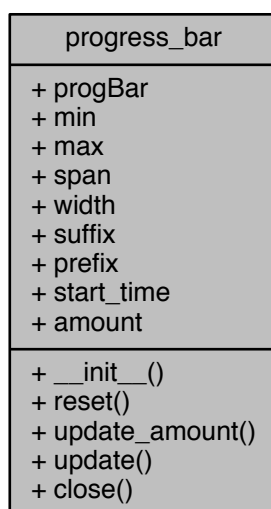
The documentation for this class was generated from the following file:

- [_sensor.py](#)

20.6 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](#)

20.6.1 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()
```

20.6.2 Constructor & Destructor Documentation**20.6.2.1 __init__()**

```
def __init__ (
    self,
    maxValue = 100,
    prefix = '',
    minValue = 0,
    totalWidth = 60 )
```

20.6.3 Member Function Documentation

20.6.3.1 close()

```
def close (
    self )
```

Prints a blank space at the end to ensure proper printing of future statements.

20.6.3.2 reset()

```
def reset (
    self )
```

20.6.3.3 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.

20.6.3.4 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).

20.6.4 Member Data Documentation

20.6.4.1 amount

amount

20.6.4.2 max

max

20.6.4.3 min

min

20.6.4.4 prefix

prefix

20.6.4.5 progBar

progBar

20.6.4.6 span

span

20.6.4.7 start_time

start_time

20.6.4.8 suffix

suffix

20.6.4.9 width

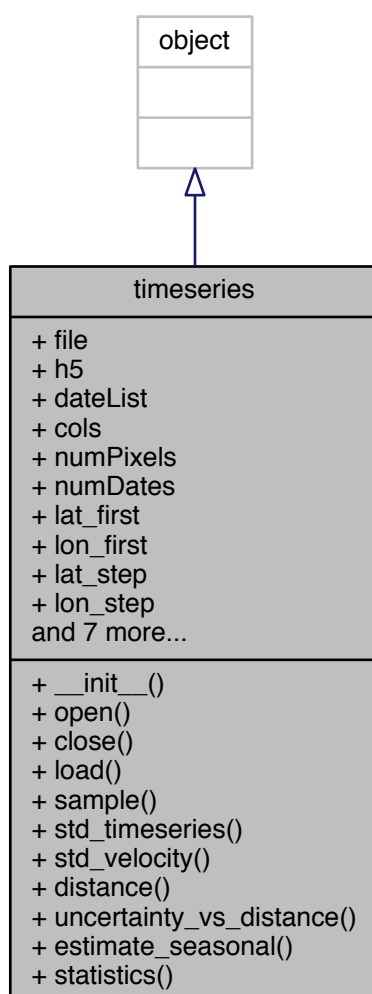
width

The documentation for this class was generated from the following file:

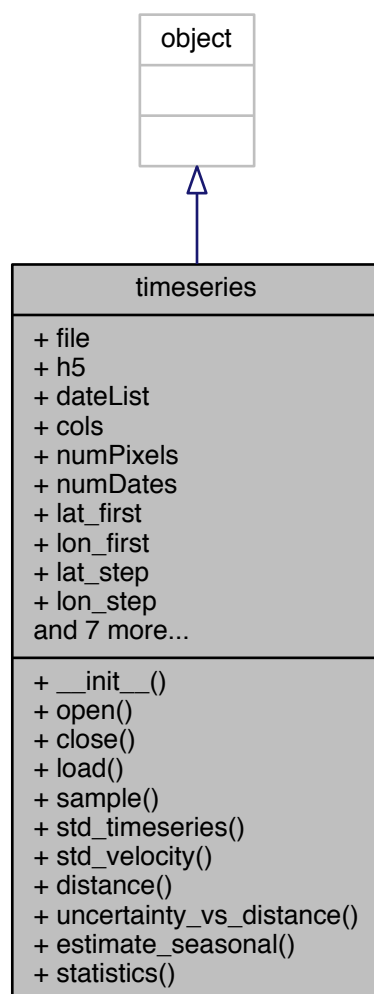
- [_datetime.py](#)

20.7 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](#)

20.7.1 Detailed Description

20.7.2 Constructor & Destructor Documentation

20.7.2.1 `__init__()`

```
def __init__ (
    self,
    file = None )
```

20.7.3 Member Function Documentation

20.7.3.1 `close()`

```
def close (
    self )
```

20.7.3.2 `distance()`

```
def distance (
    self,
    i )
```

20.7.3.3 estimate_seasonal()

```
def estimate_seasonal (
    self,
    inps )
```

20.7.3.4 load()

```
def load (
    self )
```

20.7.3.5 open()

```
def open (
    self )
```

20.7.3.6 sample()

```
def sample (
    self,
    numSamples = 500,
    mask = None )
```

20.7.3.7 statistics()

```
def statistics (
    self,
    inps )
```

20.7.3.8 std_timeseries()

```
def std_timeseries (
    self,
    ref )
```

20.7.3.9 std_velocity()

```
def std_velocity (
    self,
    sar_dates )
```

20.7.3.10 uncertainty_vs_distance()

```
def uncertainty_vs_distance (
    self,
    sar_dates )
```

20.7.4 Member Data Documentation**20.7.4.1 cols**

cols

20.7.4.2 Data

Data

20.7.4.3 dateList

dateList

20.7.4.4 dist

dist

20.7.4.5 file

file

20.7.4.6 h5

h5

20.7.4.7 idx

idx

20.7.4.8 lat

lat

20.7.4.9 lat_first

lat_first

20.7.4.10 lat_step

lat_step

20.7.4.11 lon

lon

20.7.4.12 lon_first

lon_first

20.7.4.13 lon_step

lon_step

20.7.4.14 numDates

numDates

20.7.4.15 numPixels

numPixels

20.7.4.16 relative_std

relative_std

20.7.4.17 `relative_std_velocity`

`relative_std_velocity`

The documentation for this class was generated from the following file:

- [delayTimeseries.py](#)

21 File Documentation

21.1 `__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](#) = [20.0, 12.0]

21.2 `_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)
- def [list_ifgram2date12](#) (ifgram_list)

21.3 `_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')

21.4 `_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)
- def [igram_perp_baseline_list](#) (File)
- def [azimuth_bandwidth](#) (sensor)
- def [range_bandwidth](#) (sensor)
- def [wavelength](#) (sensor)
- def [incidence_angle](#) (sensor)
- def [signal2noise_ratio](#) (sensor)
- def [critical_perp_baseline](#) (sensor)
- def [calculate_doppler_overlap](#) (dop_a, dop_b, bandwidth_az)
- 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)
- 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=[])
- def [plot_perp_baseline_hist](#) (ax, date8_list, pbase_list, plot_dict={}, date8_list_drop=[])
- def [plot_coherence_matrix](#) (ax, date12_list, coherence_list, 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](#)

21.5 [_plot.py](#) File Reference

Namespaces

- [pysar._plot](#)

Functions

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

21.6 [_pysar_utilities.py](#) File Reference

Namespaces

- [pysar._pysar_utilities](#)

Functions

- def [check_loaded_dataset](#) (work_dir='.', inps=None, print_message=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)
- def [perp_baseline_timeseries](#) (atr, dimension=1)
- def [range_distance](#) (atr, dimension=2)
- def [incidence_angle](#) (atr, dimension=2, print_message=True)
- def [which](#) (program)
- def [get_file_stack](#) (File, maskFile=None)
- def [check_drop_ifgram](#) (h5, atr, ifgram_list, print_message=True)
- def [nonzero_mask](#) (File, outFile='mask.h5')
- def [get_spatial_average](#) (File, maskFile=None, box=None, saveList=True)
- def [spatial_average](#) (File, mask=None, box=None, saveList=False)
- def [temporal_average](#) (File, outFile=None)
- def [get_file_list](#) (fileList, abspath=False)

- def `check_file_size` (fname_list, mode_width=None, mode_length=None)
- def `mode` (thelist)
- def `range_resolution` (atr, print_message=True)
- def `azimuth_resolution` (atr)
- def `glob2radar` (lat, lon, transFile='geomap *.trans', atr_rdr=dict(), print_message=True)
- def `radar2glob` (az, rg, transFile='geomap *.trans', atr_rdr=dict(), print_message=True)
- def `check_variable_name` (path)
- def `hillshade` (data, scale)
- def `date_list` (h5file)
- def `design_matrix` (ifgramFile=None, date12_list=[])
- def `timeseries_inversion` (ifgramFile, timeseriesFile)
- 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 `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)

21.7 `_readfile.py` File Reference

Namespaces

- `pysar._readfile`

Functions

- def `read` (File, box=(), epoch=None)
- 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)
- def `attribute_isce2roipac` (xml_dict)
- def `read_float32` (File, box=None)
- def `read_complex_float32` (fname, byteorder=None, real_imag=False)
- def `read_real_float32` (fname, byteorder=None)
- def `read_complex_int16` (File, box=None, real_imag=False)
- def `read_dem` (File)
- def `read_real_int16` (File)
- def `read_flag` (File)
- 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']
- list [single_dataset_hdf5_file](#) = ['dem','mask','rmse','temporal_coherence', 'velocity']

21.8 [_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)

21.9 [_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](#)

21.10 [_Sidebar.md](#) File Reference21.11 [_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)

21.12 `_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)

21.13 `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](#)

21.14 `add_attribute.py` File Reference

Namespaces

- [pysar.add_attribute](#)

Functions

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

21.15 add_attribute_insarmaps.py File Reference

Classes

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

Namespaces

- [pysar.add_attribute_insarmaps](#)

Functions

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

21.16 asc_desc.py File Reference

Namespaces

- [pysar.asc_desc](#)

Functions

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

Variables

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

21.17 Attributes.md File Reference

21.18 baseline_error.py File Reference

Namespaces

- [pysar.baseline_error](#)

Functions

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

21.19 baseline_trop.py File Reference

Namespaces

- [pysar.baseline_trop](#)

Functions

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

21.20 Bibliography.md File Reference

21.21 coord_glob2radar.py File Reference

Namespaces

- [pysar.coord_glob2radar](#)

Functions

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

21.22 coord_radar2glob.py File Reference

Namespaces

- [pysar.coord_radar2glob](#)

Functions

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

21.23 Coordinate.md File Reference

21.24 correct_dem.py File Reference

Namespaces

- [pysar.correct_dem](#)

Functions

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

21.25 correlation_with_dem.py File Reference

Namespaces

- [pysar.correlation_with_dem](#)

Functions

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

21.26 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)

21.27 DEM.md File Reference

21.28 dem_error.py File Reference

Namespaces

- [pysar.dem_error](#)

Functions

- def [read_template2inps](#) (template_file, inps=None)
- def [get_exclude_date](#) (inps, date_list_all)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

21.29 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)

21.30 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)

21.31 download_ecmwf.py File Reference

Namespaces

- [pysar.download_ecmwf](#)

Variables

- [start_date](#)
- [end_date](#)
- [hour](#)
- [step](#)
- [days](#)
- [dateListFile](#)
- [f](#)
- [date](#)
- [date_str](#)
- [tropCmd](#)
- [runFile](#)
- [maxJobNum](#)
- [jobCmd](#)

21.32 epoch_coherence.py File Reference

Namespaces

- [pysar.epoch_coherence](#)

Functions

- def [epoch_coherence_file](#) (inFile, maskFile='maskTempCoh.h5', outFile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

21.33 Example.md File Reference

21.34 File-Descriptions.md File Reference

21.35 Gamma-File-Decription.md File Reference

21.36 gamma_view.py File Reference

Namespaces

- [pysar.gamma_view](#)

Functions

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

21.37 generate_mask.py File Reference

Namespaces

- [pysar.generate_mask](#)

Functions

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

Variables

- [EXAMPLE](#)

21.38 geocode.py File Reference

Namespaces

- [pysar.geocode](#)

Functions

- def [update_attribute4isce](#) (atr_rdr, inps, geo_data)
- def [geocode_attribute_with_geo_lut](#) (atr_rdr, atr_lut, print_msg=True)
- def [geocode_file_with_geo_lut](#) (fname, lut_file=None, method='nearest', fill_value=np.nan, fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

21.39 geocode_orig.py File Reference

Namespaces

- [pysar.geocode_orig](#)

Functions

- def [update_attribute4isce](#) (atr_rdr, inps, geo_data)
- def [geocode_attribute_with_geo_lookup_table](#) (atr_rdr, atr_lut, print_message=True)
- def [geocode_file_with_geo_lookup_table](#) (fname, lookup_file=None, interp_method='nearest', fname_out=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

21.40 `get_modis_v3.py` File Reference

Namespaces

- [get_modis_v3](#)

Functions

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

Variables

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

21.41 `Google-Earth.md` File Reference

21.42 `Home.md` File Reference

21.43 `ifgram_closure.py` File Reference

Namespaces

- [pysar.ifgram_closure](#)

Functions

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

21.44 `ifgram_inversion.py` File Reference

Namespaces

- [pysar.ifgram_inversion](#)

Functions

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

Variables

- [EXAMPLE](#)

21.45 ifgram_reconstruction.py File Reference

Namespaces

- [pysar.ifgram_reconstruction](#)

Functions

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

21.46 ifgram_simulation.py File Reference

Namespaces

- [pysar.ifgram_simulation](#)

Functions

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

Variables

- [EXAMPLE](#)

21.47 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](#)

21.48 incidence_angle.py File Reference

Namespaces

- [pysar.incidence_angle](#)

Functions

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

21.49 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)

21.50 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)

21.51 insarmaps_query.py File Reference

Classes

- class [BasicHTTP](#)

Namespaces

- [pysar.insarmaps_query](#)

Functions

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

21.52 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](#)

21.53 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](#)

21.54 load_data.py File Reference

Namespaces

- [pysar.load_data](#)

Functions

- def [auto_path_miami](#) (inps, template={})
Sub Functions #####.
- def [mode](#) (thelist)
- def [check_file_size](#) (fileList, mode_width=None, mode_length=None)
- def [check_existed_hdf5_file](#) (roipacFileList, hdf5File)
- def [load_multi_group_hdf5](#) (fileType, fileList, hdf5File='unwraplfggram.h5', extra_meta_dict=dict())
- def [load_single_dataset_hdf5](#) (file_type, infile, outfile, extra_meta_dict=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

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

21.55 load_data_bak.py File Reference

Namespaces

- [pysar.load_data_bak](#)

Functions

- def [auto_path_miami](#) (inps, template={})
Sub Functions #####.
- def [mode](#) (thelist)
- def [check_file_size](#) (fileList, mode_width=None, mode_length=None)
- def [check_existed_hdf5_file](#) (roipacFileList, hdf5File)
- def [roipac2multi_group_hdf5](#) (fileType, fileList, hdf5File='unwraplfggram.h5', extra_meta_dict=dict())
- def [roipac_nonzero_mask](#) (unwFileList, maskFile='mask.h5')
- def [roipac2single_dataset_hdf5](#) (file_type, infile, outfile, extra_meta_dict=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

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

21.56 load_dem.py File Reference

Namespaces

- [pysar.load_dem](#)

Variables

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

21.57 lod.py File Reference

Namespaces

- [pysar.lod](#)

Functions

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

21.58 look_angle.py File Reference

Namespaces

- [pysar.look_angle](#)

Functions

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

21.59 los2enu.py File Reference

Namespaces

- [pysar.los2enu](#)

Functions

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

21.60 mask.py File Reference

Namespaces

- [pysar.mask](#)

Functions

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

Variables

- [EXAMPLE](#)

21.61 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](#)

21.62 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](#)

21.63 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](#)

21.64 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_message=True)
- def [multilook_file](#) (infile, lks_y, lks_x, outfile=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

21.65 perp_baseline.py File Reference

Namespaces

- [pysar.perp_baseline](#)

Functions

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

21.66 plot_atmDrop.py File Reference

Namespaces

- [pysar.plot_atmDrop](#)

Variables

- [projectList](#)
- [projectDir](#)
- [numProject](#)
- [fig](#)
- [figsize](#)
- [ax1](#)
- [ax2](#)
- [offset](#)
- [fl](#)

Read txt file.

- [lines](#)
- [lineNum](#)
- [dateList6](#)
- [meanList](#)
- [pixList](#)
- [line_s](#)
- [dateList](#)
- [dates](#)
- [datevector](#)
- [idxMean](#)
- [key](#)
- [idxPix](#)
- [sc1](#)

Plot.

- [c](#)
- [s](#)
- [alpha](#)
- [vmin](#)
- [vmax](#)
- [sc2](#)
- [fontsize](#)
- [cbar](#)
- [bbox_inches](#)
- [transparent](#)

21.67 plot_network.py File Reference

Namespaces

- [pysar.plot_network](#)

Functions

- def [cmdLineParse](#) ()
- def [main](#) (argv)
Main Function #####.

Variables

- [BL_LIST](#)
- [DATE12_LIST](#)
- [EXAMPLE](#)

21.68 [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](#)

21.69 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](#)

21.70 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](#)

21.71 prep_roipac.py File Reference

Namespaces

- [pysar.prep_roipac](#)

Functions

- def [extract_attribute](#) (fname)
Sub Functions #####
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [EXAMPLE](#)
- [DESCRIPTION](#)

21.72 pysarApp.py File Reference

Namespaces

- [pysar.pysarApp](#)

Functions

- def [check_subset_file](#) (File, inps_dict, outFile=None, overwrite=False)
- def [check_geocode_file](#) (geomapFile, File, outFile=None)
- def [subset_dataset](#) (inps, geo_box4geo, pix_box4rdr)
- def [create_subset_dataset](#) (inps, pix_box=None, geo_box=None)
- def [multilook_dataset](#) (inps, lks_y=None, lks_x=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

21.73 quality_map.py File Reference

Namespaces

- [pysar.quality_map](#)

Functions

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

21.74 range_distance.py File Reference

Namespaces

- [pysar.range_distance](#)

Functions

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

21.75 README.md File Reference

21.76 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](#)

21.77 remove_plane.py File Reference

Namespaces

- [pysar.remove_plane](#)

Functions

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

Variables

- [EXAMPLE](#)

21.78 rewrap.py File Reference

Namespaces

- [pysar.rewrap](#)

Functions

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

21.79 SAR-Sensor-Parameter.md File Reference

21.80 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](#)

21.81 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](#)

21.82 save_mat.py File Reference

Namespaces

- [pysar.save_mat](#)

Functions

- def [usage](#) ()
- def [yyyymmdd2years](#) (date)
- def [main](#) (argv)

21.83 save_mat_orig.py File Reference

Namespaces

- [pysar.save_mat_orig](#)

Functions

- def [usage](#) ()
- def [yyyymmdd2years](#) (date)
- def [main](#) (argv)

21.84 save_roipac.py File Reference

Namespaces

- [pysar.save_roipac](#)

Functions

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

21.85 save_unavco.py File Reference

Namespaces

- [pysar.save_unavco](#)

Functions

- def [get_mission_name](#) (meta_dict)
- def [metadata_pysar2unavco](#) (pysar_meta_dict, dateList)
- def [get_unavco_filename](#) (timeseriesFile)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

- [INT_ZERO](#)
- [FLOAT_ZERO](#)
- [CPX_ZERO](#)
- [EXAMPLE](#)

21.86 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_message=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](#)

21.87 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](#)
- [METHOD](#)
- [EXAMPLE](#)
- [TEMPLATE](#)

21.88 spatial_average.py File Reference

Namespaces

- [pysar.spatial_average](#)

Functions

- `def cmdLineParse ()`
- `def main (argv)`
Main Function #####.

Variables

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

21.89 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](#)

21.90 subset.py File Reference

Namespaces

- [pysar.subset](#)

Functions

- def [coord_geo2radar](#) (geoCoord, atr, coordType)
Example: 300 = coord_geo2radar(32.104990, atr,'lat') [1000,1500] = coord_geo2radar([130.5,131.4],atr,'lon')
- def [coord_radar2geo](#) (radarCoord, 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_message=True)
- def [get_coverage_box](#) (atr)
- def [read_subset_template2box](#) (templateFile)
- def [bbox_geo2radar](#) (geo_box, atr_rdr=dict(), transFile='geomap *.trans')
- def [bbox_radar2geo](#) (pix_box, atr_rdr=dict(), transFile='geomap *.trans')
- 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](#)

21.91 sum_epochs.py File Reference

Namespaces

- [pysar.sum_epochs](#)

Functions

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

21.92 temporal_average.py File Reference

Namespaces

- [pysar.temporal_average](#)

Functions

- def [usage](#) ()
Usage #####.
- def [main](#) (argv)
Main Function #####.

21.93 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](#)

21.94 temporal_derivative.py File Reference

Namespaces

- [pysar.temporal_derivative](#)

Functions

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

21.95 temporal_filter.py File Reference

Namespaces

- [pysar.temporal_filter](#)

Functions

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

Variables

- [EXAMPLE](#)

21.96 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](#)

21.97 timeseries_rms.py File Reference

Namespaces

- [pysar.timeseries_rms](#)

Functions

- def [read_template2inps](#) (templateFile, inps=None)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

21.98 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](#)

21.99 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) minorLocator = 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](#)

21.100 tropcor_phase_elevation.py File Reference

Namespaces

- [pysar.tropcor_phase_elevation](#)

Functions

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

Variables

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

21.101 tropcor_pyaps.py File Reference

Namespaces

- [pysar.tropcor_pyaps](#)

Functions

- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')
- def [get_delay](#) (grib_file, atr, inps_dict)
- def [dload_grib](#) (date_list, hour, grib_source='ECMWF', weather_dir='.')
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

21.102 tropcor_pyaps_orig.py File Reference

Namespaces

- [pysar.tropcor_pyaps_orig](#)

Functions

- def [closest_weather_product_time](#) (sar_acquisition_time, grib_source='ECMWF')
- def [get_delay](#) (grib_file, atr, inps_dict)
- def [cmdLineParse](#) ()
- def [main](#) (argv)

Variables

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

21.103 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](#) (relative_std_file, inps)
- def [download](#) (inps)
- def [main](#) (argv)

Variables

- [EXAMPLE](#)

21.104 tsvviewer.py File Reference

Namespaces

- [pysar.tsvviewer](#)

Functions

- def [read_timeseries_yx](#) (timeseries_file, y, x)
- 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](#)
- [length](#)
- [width](#)
- [ullon](#)
- [ullat](#)
- [lon_step](#)
- [lat_step](#)
- [lrlon](#)
- [lrlat](#)
- [y](#)
- [x](#)
- [yx](#)
- [ref_yx](#)
- [unit_fac](#)
- [flip_ud](#)
- [left_lr](#)
- [mask](#)
- [d_v](#)

- [ref_d_v](#)
- [data_lim](#)
- [ylim](#)
- [fig_v](#)

Fig 1 - Cumulative Displacement Map.

- [ax_v](#)
- [img](#)
- [cmap](#)
- [colormap](#)
- [clim](#)
- [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.

21.105 UNAVCO-InSAR-Archive.md File Reference

21.106 unavco2insarmaps.py File Reference

Namespaces

- [pysar.unavco2insarmaps](#)

Functions

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

21.107 unavco2json_mbtiles.py File Reference

Namespaces

- [pysar.unavco2json_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, dataset_keys, json_path, folder_name)
- def [make_json_file](#) (chunk_num, points, dataset_keys, json_path, folder_name)
- def [build_parser](#) ()
- def [main](#) ()

Variables

- dictionary [needed_attributes](#)

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

Variables

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

21.109 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)
- 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

- string [EXAMPLE](#)
- string [PLOT_TEMPLATE](#)

21.110 Web-Viewer.md File Reference

Index

- [_Sidebar.md](#), [258](#)
 - [_MOSEK](#)
 - [pysar::l1](#), [95](#)
 - [__init__](#)
 - [delayTimeseries::timeseries](#), [250](#)
 - [pysar::_datetime::progress_bar](#), [245](#)
 - [pysar::_sensor::JERS](#), [243](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), [234](#)
 - [pysar::add_attribute_insarmaps::InsarDataset](#)↔
[Controller](#), [240](#)
 - [__init__.py](#), [254](#)
 - [_datetime.py](#), [254](#)
 - [_gmt.py](#), [255](#)
 - [_network.py](#), [255](#)
 - [_plot.py](#), [256](#)
 - [_pysar_utilities.py](#), [256](#)
 - [_readfile.py](#), [257](#)
 - [_remove_surface.py](#), [258](#)
 - [_sensor.py](#), [258](#)
 - [_variance.py](#), [258](#)
 - [_writefile.py](#), [259](#)
- [add.py](#), [259](#)
- [add_attribute](#)
 - [pysar::_pysar_utilities](#), [45](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), [234](#)
- [add_attribute.py](#), [259](#)
- [add_attribute_insarmaps.py](#), [260](#)
- [add_files](#)
 - [pysar::add](#), [68](#)
- [add_inner_title](#)
 - [pysar::view](#), [221](#)
- [add_matrix](#)
 - [pysar::add](#), [68](#)
- [add_plot_attribute](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), [234](#)
- [alpha](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::plot_atmDrop](#), [135](#)
 - [pysar::transect_legacy](#), [185](#)
- [amount](#)
 - [pysar::_datetime::progress_bar](#), [246](#)
- [amp](#)
 - [pysar::load_dem](#), [102](#)
- [asc_desc.py](#), [260](#)
- [atr](#)
 - [plot_tropcor_phase_elevation](#), [22](#)
 - [pysar::tsviewer](#), [206](#)
- [atr2](#)
 - [plot_tropcor_phase_elevation](#), [23](#)
- [atr3](#)
 - [plot_tropcor_phase_elevation](#), [23](#)
- [atr4](#)
 - [plot_tropcor_phase_elevation](#), [23](#)
- [attribute_exists_for_dataset](#)
 - [pysar::add_attribute_insarmaps::InsarDatabase](#)↔
[Controller](#), [235](#)
- [attribute_gamma2roipac](#)
 - [pysar::_readfile](#), [58](#)
- [attribute_isce2roipac](#)
 - [pysar::_readfile](#), [58](#)
- [Attributes.md](#), [260](#)
- [auto_adjust_xaxis_date](#)
 - [pysar::_datetime](#), [30](#)
- [auto_adjust_yaxis](#)
 - [pysar::_network](#), [33](#)
- [auto_figure_title](#)
 - [pysar::view](#), [221](#)
- [auto_flip_direction](#)
 - [pysar::view](#), [221](#)
- [auto_lalo_sequence](#)
 - [pysar::view::Basemap2](#), [229](#)
- [auto_path_miami](#)
 - [pysar::load_data](#), [96](#)
 - [pysar::load_data_bak](#), [99](#)
- [auto_row_col_num](#)
 - [pysar::view](#), [221](#)
- [avglnSAR](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::transect_legacy](#), [185](#)
- [ax](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::transect_legacy](#), [185](#)
- [ax1](#)
 - [pysar::plot_atmDrop](#), [135](#)
- [ax2](#)
 - [pysar::plot_atmDrop](#), [135](#)
- [ax_time](#)
 - [pysar::tsviewer](#), [206](#)
- [ax_ts](#)
 - [pysar::tsviewer](#), [206](#)
- [ax_v](#)
 - [pysar::tsviewer](#), [207](#)
- [axes](#)
 - [plot_tropcor_phase_elevation](#), [23](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::transect_legacy](#), [185](#)
- [axes2](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::transect_legacy](#), [185](#)
- [axis](#)
 - [pysar::multi_transect](#), [118](#)
 - [pysar::transect_legacy](#), [185](#)
- [axisbg](#)
 - [pysar::tsviewer](#), [207](#)
- [azimuth_bandwidth](#)
 - [pysar::_network](#), [34](#)

- azimuth_resolution
 - pysar::_pysar_utilities, 45
- BASELINE_LIST_FILE
 - pysar::_network, 43
- BL_LIST
 - pysar::plot_network, 139
- bandwidth
 - pysar::_sensor::JERS, 243
- baseline_error.py, 260
- baseline_trop.py, 261
- baselineTimeseries
 - pysar::prep_isce, 142
- Basemap2, 228
- BasicHTTP, 231
- bbox_geo2radar
 - pysar::subset, 165
- bbox_inches
 - plot_tropcor_phase_elevation, 23
 - pysar::plot_atmDrop, 135
 - pysar::tsviewer, 207
- bbox_radar2geo
 - pysar::subset, 165
- Bh_Bv_timeseries
 - pysar::_pysar_utilities, 45
- Bibliography.md, 261
- bin_variance
 - pysar::_variance, 65
- bodyOutput
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 241
- box_geo2pixel
 - pysar::subset, 166
- box_pixel2geo
 - pysar::subset, 166
- bridging_data
 - pysar::unwrap_error, 218
- build_parser
 - pysar::add_attribute_insarmaps, 70
 - pysar::insarmaps_query, 92
 - pysar::json_mbtiles2insarmaps, 92
 - pysar::unavco2insarmaps, 215
 - pysar::unavco2json_mbtiles, 216
- buildURL
 - pysar::insarmaps_query, 92
- c
 - pysar::multi_transect, 118
 - pysar::plot_atmDrop, 135
 - pysar::transect_legacy, 186
- c_prof_edge
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- CPX_ZERO
 - pysar::save_unavco, 157
- calculate_doppler_overlap
 - pysar::_network, 34
- cbar
 - pysar::plot_atmDrop, 135
- pysar::tsviewer, 207
- center_frequency
 - pysar::_sensor::JERS, 243
- cf
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- check_box_within_data_coverage
 - pysar::subset, 166
- check_colormap_input
 - pysar::view, 222
- check_drop_ifgram
 - pysar::_pysar_utilities, 45
- check_existed_hdf5_file
 - pysar::load_data, 96
 - pysar::load_data_bak, 99
- check_file_size
 - pysar::_pysar_utilities, 45
 - pysar::load_data, 96
 - pysar::load_data_bak, 99
- check_geocode_file
 - pysar::pysarApp, 146
- check_loaded_dataset
 - pysar::_pysar_utilities, 46
- check_multilook_input
 - pysar::view, 222
- check_parallel
 - pysar::_pysar_utilities, 46
- check_result
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- check_result2
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- check_st_in_box
 - pysar::multi_transect, 114
 - pysar::transect_legacy, 181
- check_st_in_box2
 - pysar::multi_transect, 114
 - pysar::transect_legacy, 182
- check_subset_file
 - pysar::pysarApp, 146
- check_variable_name
 - pysar::_pysar_utilities, 46
 - pysar::_readfile, 58
- cid
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
 - pysar::tsviewer, 207
- circle_index
 - pysar::_pysar_utilities, 46
- clim
 - pysar::tsviewer, 207
- close
 - delayTimeseries::timeseries, 250
 - pysar::_datetime::progress_bar, 245
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 235
- closest_weather_product_time

- pysar::tropcor_pyaps, 200
- pysar::tropcor_pyaps_orig, 201
- cmap
 - pysar::tsviewer, 207
- cmdLineParse
 - pysar::add, 69
 - pysar::asc_desc, 71
 - pysar::dem_error, 75
 - pysar::diff, 76
 - pysar::epoch_coherence, 79
 - pysar::generate_mask, 81
 - pysar::geocode, 82
 - pysar::geocode_orig, 83
 - pysar::ifgram_inversion, 86
 - pysar::ifgram_simulation, 87
 - pysar::image_math, 88
 - pysar::load_data, 96
 - pysar::load_data_bak, 100
 - pysar::mask, 106
 - pysar::match, 107
 - pysar::modify_network, 109
 - pysar::multilook, 132
 - pysar::plot_network, 139
 - pysar::prep_gamma, 140
 - pysar::prep_isce, 142
 - pysar::prep_roipac, 144
 - pysar::pysarApp, 146
 - pysar::reference_epoch, 149
 - pysar::remove_plane, 150
 - pysar::save_gmt, 152
 - pysar::save_kml, 153
 - pysar::save_unavco, 156
 - pysar::seed_data, 158
 - pysar::select_network, 161
 - pysar::spatial_average, 163
 - pysar::spatial_filter, 163
 - pysar::subset, 166
 - pysar::temporal_filter, 173
 - pysar::timeseries2velocity, 174
 - pysar::timeseries_rms, 175
 - pysar::transect, 177
 - pysar::tropcor_phase_elevation, 199
 - pysar::tropcor_pyaps, 200
 - pysar::tropcor_pyaps_orig, 202
 - pysar::tsviewer, 204
 - pysar::unwrap_error, 218
 - pysar::view, 222
 - troposphere_uncertainty, 227
- coherence_matrix
 - pysar::_network, 34
- color
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- colormap
 - pysar::tsviewer, 207
- cols
 - delayTimeseries::timeseries, 252
- compression
 - pysar::load_dem, 102
- con
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 237
- connect
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 235
- convert_data
 - pysar::unavco2json_mbtiles, 216
- coord_geo2radar
 - pysar::subset, 167
- coord_glob2radar.py, 261
- coord_radar2geo
 - pysar::subset, 167
- coord_radar2glob.py, 261
- Coordinate.md, 261
- copy_file
 - pysar::load_data, 96
 - pysar::load_data_bak, 100
- corners
 - pysar::match, 107
- correct_dem.py, 261
- correct_lod_file
 - pysar::lod, 104
- correlation_with_dem.py, 262
- create_area_table_if_not_exists
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 235
- create_subset_dataset
 - pysar::pysarApp, 146
- createParser
 - pysar::prep_isce, 143
- critical_perp_baseline
 - pysar::_network, 34
- curl_login
 - pysar::add_attribute_insarmaps::InsarDataset←
Controller, 240
- cursor
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, 237
- D
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 186
- d_ts
 - pysar::tsviewer, 208
- d_v
 - pysar::tsviewer, 208
- DATE12_LIST
 - pysar::plot_network, 139
- dBh_dBv_timeseries
 - pysar::_pysar_utilities, 47
- DEM.md, 262
- DESCRIPTION
 - pysar::prep_gamma, 142
 - pysar::prep_roipac, 145
 - pysar::temporal_coherence, 172
 - pysar::unwrap_error, 219
- DROP_DATE_TXT

- pysar::timeseries2velocity, 175
- Data
 - delayTimeseries::timeseries, 252
- data
 - plot_tropcor_phase_elevation, 23
 - pysar::load_dem, 103
- data2
 - plot_tropcor_phase_elevation, 23
- data_lim
 - pysar::tsviewer, 208
- data_operation
 - pysar::image_math, 88
- dataList
 - plot_tropcor_phase_elevation, 23
- dataset
 - pysar::multi_transect, 119
 - pysar::transect_legacy, 187
- date
 - pysar::download_ecmwf, 78
- date12_list2index
 - pysar::_network, 35
- date_index
 - pysar::_datetime, 30
- date_list
 - pysar::_pysar_utilities, 47
- date_list2tbase
 - pysar::_datetime, 30
- date_list2vector
 - pysar::_datetime, 30
- date_num
 - pysar::tsviewer, 208
- date_str
 - pysar::download_ecmwf, 78
- dateList
 - delayTimeseries::timeseries, 252
 - pysar::plot_atmDrop, 135
 - pysar::tsviewer, 208
- dateList6
 - pysar::plot_atmDrop, 135
- dateListFile
 - pysar::download_ecmwf, 78
- daterange
 - dloadUtil, 20
- dates
 - pysar::plot_atmDrop, 135
 - pysar::tsviewer, 208
- datevector
 - pysar::plot_atmDrop, 136
- days
 - pysar::download_ecmwf, 78
- db
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 237
- dbHost
 - pysar::json_mbtiles2insarmaps, 93
- dbPassword
 - pysar::json_mbtiles2insarmaps, 93
- dbUsername
 - pysar::json_mbtiles2insarmaps, 93
- delayTimeseries, 19
 - nearest_valid, 19
 - write_to_h5, 19
- delayTimeseries.py, 262
- delayTimeseries::timeseries
 - __init__, 250
 - close, 250
 - cols, 252
 - Data, 252
 - dateList, 252
 - dist, 252
 - distance, 250
 - estimate_seasonal, 250
 - file, 252
 - h5, 252
 - idx, 252
 - lat, 252
 - lat_first, 253
 - lat_step, 253
 - load, 251
 - lon, 253
 - lon_first, 253
 - lon_step, 253
 - numDates, 253
 - numPixels, 253
 - open, 251
 - relative_std, 253
 - relative_std_velocity, 253
 - sample, 251
 - statistics, 251
 - std_timeseries, 251
 - std_velocity, 251
 - uncertainty_vs_distance, 251
- delimiter
 - pysar::tsviewer, 208
- dem
 - plot_tropcor_phase_elevation, 23
 - pysar::load_dem, 103
- dem_atr
 - plot_tropcor_phase_elevation, 24
- dem_error.py, 262
- demFile
 - plot_tropcor_phase_elevation, 24
 - pysar::load_dem, 103
- demRsc
 - pysar::load_dem, 103
- design_matrix
 - pysar::_pysar_utilities, 47
- df0
 - pysar::transect_legacy, 187
- df0_km
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 187
- df1
 - pysar::transect_legacy, 187
- dg
 - pysar::multi_transect, 120

- pysar::transect_legacy, 187
- diff.py, 263
- diff_data
 - pysar::diff, 76
- diff_file
 - pysar::diff, 77
- dist
 - delayTimeseries::timeseries, 252
- dist_point_from_line
 - pysar::multi_transect, 114
 - pysar::transect_legacy, 182
- DistGPS
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 187
- distance
 - delayTimeseries::timeseries, 250
- dload_grib
 - pysar::tropcor_pyaps, 200
- dloadUtil, 20
 - daterange, 20
 - download_atmosphereModel, 20
 - download_modis, 20
 - get_date, 20
 - pwv2zwd, 20
 - read_modis, 20
 - zwd2swd, 21
- dloadUtil.py, 263
- dms2d
 - pysar::multi_transect, 114
 - pysar::transect_legacy, 182
- download
 - troposphere_uncertainty, 227
- download_atmosphereModel
 - dloadUtil, 20
- download_ecmwf.py, 263
- download_modis
 - dloadUtil, 20
- dp
 - pysar::multi_transect, 120
- dpi
 - plot_tropcor_phase_elevation, 24
 - pysar::tsviewer, 208
- draw_lalo_label
 - pysar::view::Basemap2, 229
- drawscale
 - pysar::view::Basemap2, 230
- dset
 - pysar::load_dem, 103
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 187
- dtype
 - pysar::tsviewer, 208
- DX
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 187
- dx
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 187

- DY
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 188
- dy
 - pysar::multi_transect, 120
 - pysar::transect_legacy, 188
- e_ts
 - pysar::tsviewer, 209
- EXAMPLE
 - pysar::add, 69
 - pysar::asc_desc, 71
 - pysar::dem_error, 76
 - pysar::epoch_coherence, 80
 - pysar::generate_mask, 81
 - pysar::geocode, 83
 - pysar::geocode_orig, 85
 - pysar::ifgram_inversion, 86
 - pysar::ifgram_simulation, 87
 - pysar::image_math, 89
 - pysar::load_data, 98
 - pysar::load_data_bak, 102
 - pysar::mask, 107
 - pysar::match, 108
 - pysar::modify_network, 110
 - pysar::multilook, 133
 - pysar::plot_network, 139
 - pysar::prep_gamma, 142
 - pysar::prep_roipac, 145
 - pysar::pysarApp, 147
 - pysar::reference_epoch, 150
 - pysar::remove_plane, 151
 - pysar::save_gmt, 153
 - pysar::save_kml, 154
 - pysar::save_unavco, 157
 - pysar::seed_data, 160
 - pysar::select_network, 162
 - pysar::spatial_average, 163
 - pysar::spatial_filter, 164
 - pysar::subset, 170
 - pysar::temporal_coherence, 172
 - pysar::temporal_filter, 173
 - pysar::timeseries2velocity, 175
 - pysar::timeseries_rms, 176
 - pysar::transect, 179
 - pysar::tropcor_phase_elevation, 199
 - pysar::tropcor_pyaps, 201
 - pysar::tropcor_pyaps_orig, 202
 - pysar::tsviewer, 210
 - pysar::unwrap_error, 219
 - pysar::view, 225
 - troposphere_uncertainty, 228
- earth_radius
 - pysar::transect_legacy, 188
- ecmwf
 - plot_tropcor_phase_elevation, 24
- ecmwfFile
 - plot_tropcor_phase_elevation, 24
- ecolor

- pysar::tsviewer, 209
- end_date
 - pysar::download_ecmwf, 78
- epoch
 - plot_tropcor_phase_elevation, 24
- epoch_coherence.py, 264
- epoch_coherence_file
 - pysar::epoch_coherence, 79
- error_file
 - pysar::tsviewer, 209
- error_fileContent
 - pysar::tsviewer, 209
- error_ts
 - pysar::tsviewer, 209
- estimate_seasonal
 - delayTimeseries::timeseries, 250
 - troposphere_uncertainty, 227
- ex_date
 - pysar::tsviewer, 209
- ex_date_list
 - pysar::tsviewer, 209
- ex_dates
 - pysar::tsviewer, 209
- ex_error_ts
 - pysar::tsviewer, 209
- ex_idx_list
 - pysar::tsviewer, 210
- Example.md, 264
- ext
 - pysar::load_dem, 103
- extract_attribute
 - pysar::prep_roipac, 144
- extract_attribute_dem_geo
 - pysar::prep_gamma, 140
- extract_attribute_dem_radar
 - pysar::prep_gamma, 140
- extract_attribute_interferogram
 - pysar::prep_gamma, 140
- extract_attribute_lookup_table
 - pysar::prep_gamma, 141
- extractIsceMetadata
 - pysar::prep_isce, 143
- f
 - pysar::download_ecmwf, 78
- FLOAT_ZERO
 - pysar::save_unavco, 157
- facecolor
 - pysar::tsviewer, 210
- Fault_lat
 - pysar::multi_transect, 121
- fault_loc
 - pysar::transect_legacy, 188
- Fault_lon
 - pysar::multi_transect, 121
- FaultCoords
 - pysar::multi_transect, 121
- FaultLine
 - pysar::multi_transect, 121
- pysar::transect_legacy, 188
- fig
 - plot_tropcor_phase_elevation, 24
 - pysar::multi_transect, 121
 - pysar::plot_atmDrop, 136
 - pysar::transect_legacy, 188
- fig2
 - pysar::multi_transect, 121
 - pysar::transect_legacy, 188
- fig_base
 - pysar::tsviewer, 210
- fig_ts
 - pysar::tsviewer, 210
- fig_v
 - pysar::tsviewer, 210
- figName
 - pysar::multi_transect, 121
 - pysar::transect_legacy, 188
- figsize
 - plot_tropcor_phase_elevation, 24
 - pysar::plot_atmDrop, 136
 - pysar::tsviewer, 210
- figsize_multi
 - pysar, 28
- figsize_single_max
 - pysar, 29
- figsize_single_min
 - pysar, 29
- file
 - delayTimeseries::timeseries, 252
 - pysar::_sensor::JERS, 243
- File-Descriptions.md, 264
- file_operation
 - pysar::image_math, 88
- fileExtension
 - pysar::multi_transect, 121
 - pysar::transect_legacy, 189
- fileName
 - pysar::multi_transect, 122
 - pysar::transect_legacy, 189
- filter_data
 - pysar::spatial_filter, 164
- filter_file
 - pysar::spatial_filter, 164
- find_row_column
 - pysar::insar_vs_gps, 90
 - pysar::multi_transect, 114
 - pysar::transect_legacy, 182
- fl
 - pysar::plot_atmDrop, 136
- flip_ud
 - pysar::tsviewer, 210
- fmt
 - pysar::tsviewer, 211
- fontsize
 - pysar::multi_transect, 122
 - pysar::plot_atmDrop, 136
 - pysar::transect_legacy, 189

- format_coord
 - pysar::tsviewer, [205](#), [211](#)
- four_corners
 - pysar::_pysar_utilities, [47](#)
- GDAL2NUMPY_DATATYPE
 - pysar::prep_isce, [144](#)
- GPS_in_bound
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [189](#)
- GPS_in_bound_st
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [189](#)
- GPS_lat
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [189](#)
- GPS_lon
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [189](#)
- GPS_station
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [190](#)
- GPSx
 - pysar::multi_transect, [123](#)
 - pysar::transect_legacy, [190](#)
- GPSxx
 - pysar::multi_transect, [123](#)
 - pysar::transect_legacy, [190](#)
- GPSy
 - pysar::multi_transect, [123](#)
 - pysar::transect_legacy, [190](#)
- GPSyy
 - pysar::multi_transect, [123](#)
 - pysar::transect_legacy, [190](#)
- GPS
 - pysar::multi_transect, [122](#)
 - pysar::transect_legacy, [189](#)
- Gamma-File-Decription.md, [264](#)
- gamma_view.py, [264](#)
- generate_curls
 - pysar::_pysar_utilities, [48](#)
- generate_mask.py, [265](#)
- geocode.py, [265](#)
- geocode_attribute_with_geo_lookup_table
 - pysar::geocode_orig, [84](#)
- geocode_attribute_with_geo_lut
 - pysar::geocode, [82](#)
- geocode_file_with_geo_lookup_table
 - pysar::geocode_orig, [84](#)
- geocode_file_with_geo_lut
 - pysar::geocode, [82](#)
- geocode_orig.py, [265](#)
- get
 - pysar::insarmaps_query::BasicHTTP, [231](#)
- get_H5_filename
 - pysar::unavco2insarmaps, [215](#)
- get_box_overlap_index
 - pysar::subset, [167](#)
- get_coverage_box
 - pysar::subset, [167](#)
- get_dataset_id
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, [235](#)
- get_dataset_names
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, [235](#)
- get_date
 - dloadUtil, [20](#)
 - pysar::unavco2json_mbtiles, [216](#)
- get_date12_list
 - pysar::_network, [35](#)
- get_decimal_date
 - pysar::unavco2json_mbtiles, [216](#)
- get_delay
 - pysar::tropcor_pyaps, [200](#)
 - pysar::tropcor_pyaps_orig, [202](#)
- get_distance
 - pysar::_variance, [65](#)
- get_epoch_full_list_from_input
 - pysar::view, [222](#)
- get_exclude_date
 - pysar::dem_error, [75](#)
 - pysar::timeseries2velocity, [174](#)
- get_file_list
 - pysar::_pysar_utilities, [48](#)
- get_file_stack
 - pysar::_pysar_utilities, [48](#)
- get_geo_lat_lon
 - pysar::save_gmt, [152](#)
- get_intersect
 - pysar::multi_transect, [115](#)
 - pysar::transect_legacy, [183](#)
- get_lalo_ref
 - pysar::prep_gamma, [141](#)
- get_lat_lon
 - pysar::_variance, [65](#)
 - pysar::multi_transect, [115](#)
 - pysar::transect_legacy, [183](#)
- get_mission_name
 - pysar::save_unavco, [156](#)
- get_modis_v3, [21](#)
 - main, [21](#)
 - out, [21](#)
 - start_time_main, [21](#)
 - time_elapsed, [22](#)
 - usage, [21](#)
- get_modis_v3.py, [266](#)
- get_overlap_lalo
 - pysar::asc_desc, [71](#)
- get_perp_baseline
 - pysar::prep_gamma, [141](#)
- get_residual_rms
 - pysar::_pysar_utilities, [48](#)
- get_residual_std
 - pysar::_pysar_utilities, [49](#)
- get_scale_from_disp_unit
 - pysar::transect, [177](#)

- get_spatial_average
 - pysar::_pysar_utilities, 49
- get_start_end_point
 - pysar::multi_transect, 115
- get_transect
 - pysar::multi_transect, 115
 - pysar::transect_legacy, 183
- get_triangles
 - pysar::_pysar_utilities, 49
- get_unavco_filename
 - pysar::save_unavco, 156
- get_unavco_name
 - pysar::json_mbtiles2insarmaps, 92
- get_velocity_filename
 - pysar::timeseries2velocity, 174
- glob2radar
 - pysar::_pysar_utilities, 50
- Google-Earth.md, 266
- gps_to_LOS
 - pysar::multi_transect, 115
 - pysar::transect_legacy, 183
- gpsFile
 - pysar::multi_transect, 122
 - pysar::transect_legacy, 190
- gpsLOS_ref
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 190
- gpsLOS
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 190
- group
 - pysar::load_dem, 103
- gx
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 190
- gy
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- h5
 - delayTimeseries::timeseries, 252
 - pysar::load_dem, 103
 - pysar::tsviewer, 211
- h5file_theta
 - pysar::multi_transect, 123
 - pysar::transect_legacy, 191
- hbound
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- header_info
 - pysar::tsviewer, 211
- headersOutput
 - pysar::add_attribute_insarmaps::InsarDataset↔
Controller, 241
- heading
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- hillshade
 - pysar::_pysar_utilities, 50
- Home.md, 266
- host
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 237
- hour
 - pysar::download_ecmwf, 78
- i
 - plot_tropcor_phase_elevation, 24
- IDXref
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- IDYref
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 192
- IDX
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- IDY
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- IFGRAM_LIST_FILE
 - pysar::_network, 43
- INT_ZERO
 - pysar::save_unavco, 157
- idx
 - delayTimeseries::timeseries, 252
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- idxMean
 - pysar::plot_atmDrop, 136
- idxPix
 - pysar::plot_atmDrop, 136
- idxRef
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 191
- ifgram_closure.py, 266
- ifgram_date_list
 - pysar::_datetime, 30
- ifgram_inversion.py, 266
- ifgram_reconstruction.py, 267
- ifgram_simulation.py, 267
- igram_perp_baseline_list
 - pysar::_network, 35
- image_math.py, 267
- img
 - pysar::tsviewer, 211
- incidence_angle
 - pysar::_network, 35
 - pysar::_pysar_utilities, 50
- incidence_angle.py, 268
- index_table_on
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 235
- info.py, 268
- Info_aboutFault
 - pysar::transect_legacy, 192
- inps
 - pysar::tsviewer, 211

- input_ex_date
 - pysar::tsviewer, 211
- insar_vs_gps.py, 268
- insarData
 - pysar::multi_transect, 124
 - pysar::transect_legacy, 192
- InsarDatabaseController, 232
- InsarDatasetController, 238
- insarmaps_query.py, 268
- insert_dataset_into_area_table
 - pysar::add_attribute_insarmaps::InsarDatabaseController, 236
- is_file_exist
 - pysar::_pysar_utilities, 50
- is_plot_attribute
 - pysar::_readfile, 58
- JERS, 242
- jobCmd
 - pysar::download_ecmwf, 78
- json_mbtiles2insarmaps.py, 269
- k
 - pysar::tsviewer, 211
- key
 - pysar::plot_atmDrop, 136
- l1
 - pysar::l1, 94
- l1.py, 269
- l1blas
 - pysar::l1, 94
- l1mosek
 - pysar::l1, 94
- l1mosek2
 - pysar::l1, 94
- LOGO
 - pysar::pysarApp, 147
- Lat
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
- lat
 - delayTimeseries::timeseries, 252
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
 - pysar::tsviewer, 212
- Lat0
 - pysar::multi_transect, 125
- Lat1
 - pysar::multi_transect, 125
- lat_all
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
- lat_first
 - delayTimeseries::timeseries, 253
- lat_step
 - delayTimeseries::timeseries, 253
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
- pysar::tsviewer, 212
- lat_transect
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
- lbound
 - pysar::multi_transect, 125
 - pysar::transect_legacy, 192
- left_lr
 - pysar::tsviewer, 212
- Length
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
- length
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
 - pysar::tsviewer, 212
- line
 - pysar::multi_transect, 116
 - pysar::transect_legacy, 183
- line_s
 - pysar::plot_atmDrop, 137
- lineNum
 - pysar::plot_atmDrop, 137
- lines
 - pysar::plot_atmDrop, 137
- linewidth
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
- list_ifgram2date12
 - pysar::_datetime, 31
- load
 - delayTimeseries::timeseries, 251
- load_data.py, 270
- load_data_bak.py, 270
- load_data_from_template
 - pysar::load_data, 97
 - pysar::load_data_bak, 100
- load_dem.py, 271
- load_file
 - pysar::load_data, 97
 - pysar::load_data_bak, 100
- load_multi_group_hdf5
 - pysar::load_data, 97
- load_single_dataset_hdf5
 - pysar::load_data, 97
- lod.py, 271
- log
 - pysar::select_network, 161
- Lon
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
- lon
 - delayTimeseries::timeseries, 253
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
 - pysar::tsviewer, 212
- Lon0
 - pysar::multi_transect, 126

- Lon1
 - pysar::multi_transect, 126
- lon_all
 - pysar::multi_transect, 126
 - pysar::transect_legacy, 193
- lon_first
 - delayTimeseries::timeseries, 253
- lon_step
 - delayTimeseries::timeseries, 253
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 193
 - pysar::tsviewer, 212
- lon_transect
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 193
- look_angle.py, 271
- los2enu.py, 272
- lrlat
 - pysar::tsviewer, 212
- lrlon
 - pysar::tsviewer, 212
- m
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- m1
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- m_prof_edge
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- METHOD
 - pysar::select_network, 162
- main
 - get_modis_v3, 21
 - pysar::add, 69
 - pysar::add_attribute, 69
 - pysar::add_attribute_insarmaps, 70
 - pysar::asc_desc, 71
 - pysar::baseline_error, 72
 - pysar::baseline_trop, 72
 - pysar::coord_glob2radar, 73
 - pysar::coord_radar2glob, 73
 - pysar::correct_dem, 74
 - pysar::correlation_with_dem, 74
 - pysar::dem_error, 75
 - pysar::diff, 77
 - pysar::epoch_coherence, 80
 - pysar::gamma_view, 80
 - pysar::generate_mask, 81
 - pysar::geocode, 82
 - pysar::geocode_orig, 84
 - pysar::ifgram_closure, 85
 - pysar::ifgram_inversion, 86
 - pysar::ifgram_reconstruction, 86
 - pysar::ifgram_simulation, 87
 - pysar::image_math, 88
 - pysar::incidence_angle, 89
 - pysar::info, 89
 - pysar::insar_vs_gps, 91
 - pysar::insarmaps_query, 92
 - pysar::json_mbtiles2insarmaps, 93
 - pysar::load_data, 98
 - pysar::load_data_bak, 100
 - pysar::lod, 104
 - pysar::look_angle, 104
 - pysar::los2enu, 105
 - pysar::mask, 106
 - pysar::match, 107
 - pysar::modify_network, 109
 - pysar::multi_transect, 116
 - pysar::multilook, 133
 - pysar::perp_baseline, 134
 - pysar::plot_network, 139
 - pysar::prep_gamma, 141
 - pysar::prep_isce, 143
 - pysar::prep_roipac, 145
 - pysar::pysarApp, 146
 - pysar::quality_map, 148
 - pysar::range_distance, 148
 - pysar::reference_epoch, 149
 - pysar::remove_plane, 151
 - pysar::rewrap, 151
 - pysar::save_gmt, 152
 - pysar::save_kml, 153
 - pysar::save_mat, 154
 - pysar::save_mat_orig, 155
 - pysar::save_roipac, 155
 - pysar::save_unavco, 156
 - pysar::seed_data, 158
 - pysar::select_network, 161
 - pysar::spatial_average, 163
 - pysar::spatial_filter, 164
 - pysar::subset, 167
 - pysar::sum_epochs, 170
 - pysar::temporal_average, 170
 - pysar::temporal_coherence, 171
 - pysar::temporal_derivative, 172
 - pysar::temporal_filter, 173
 - pysar::timeseries2velocity, 174
 - pysar::timeseries_rms, 176
 - pysar::transect, 177
 - pysar::transect_legacy, 184
 - pysar::tropcor_phase_elevation, 199
 - pysar::tropcor_pyaps, 200
 - pysar::tropcor_pyaps_orig, 202
 - pysar::unavco2insarmaps, 215
 - pysar::unavco2json_mbtiles, 216
 - pysar::unwrap_error, 218
 - pysar::view, 222
 - troposphere_uncertainty, 227
- make_json_file
 - pysar::unavco2json_mbtiles, 217
- make_triangle
 - pysar::_pysar_utilities, 51
- manual_offset_estimate
 - pysar::match, 107

- manual_select_pairs_to_remove
 - pysar::modify_network, 109
- manual_select_reference_yx
 - pysar::seed_data, 158
- manual_select_start_end_point
 - pysar::transect, 177
- markededgecolor
 - pysar::tsviewer, 212
- mask
 - plot_tropcor_phase_elevation, 25
 - pysar::tsviewer, 213
- mask.py, 272
- mask_file
 - pysar::mask, 106
- mask_matrix
 - pysar::mask, 106
- maskFile
 - plot_tropcor_phase_elevation, 25
- matFile
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- match.py, 272
- match_two_files
 - pysar::match, 108
- max
 - pysar::datetime::progress_bar, 246
- maxJobNum
 - pysar::download_ecmwf, 78
- meanList
 - pysar::plot_atmDrop, 137
- metadata_pysar2unavco
 - pysar::save_unavco, 157
- mf
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- mfc
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
- miami_path
 - pysar, 29
- min
 - pysar::datetime::progress_bar, 247
- mode
 - pysar::_network, 35
 - pysar::_pysar_utilities, 51
 - pysar::load_data, 98
 - pysar::load_data_bak, 101
- modify_file_date12_list
 - pysar::modify_network, 109
- modify_network.py, 273
- mp
 - pysar::transect_legacy, 194
- ms
 - plot_tropcor_phase_elevation, 25
 - pysar::multi_transect, 127
 - pysar::transect_legacy, 194
 - pysar::tsviewer, 213
- msk_atr
 - plot_tropcor_phase_elevation, 25
- multi_dataset_hdf5_file
 - pysar::_readfile, 63
- multi_group_hdf5_file
 - pysar::_readfile, 63
- multi_transect.py, 273
- multilook.py, 276
- multilook_attribute
 - pysar::multilook, 133
- multilook_dataset
 - pysar::pysarApp, 147
- multilook_file
 - pysar::multilook, 133
- multilook_matrix
 - pysar::multilook, 133
- NOTE
 - pysar::seed_data, 160
- nanmean
 - pysar::multi_transect, 116
 - pysar::transect_legacy, 184
- nanstd
 - pysar::multi_transect, 116
 - pysar::transect_legacy, 184
- ncols
 - plot_tropcor_phase_elevation, 25
- ndx
 - plot_tropcor_phase_elevation, 25
- nearest
 - pysar::insar_vs_gps, 91
 - pysar::match, 108
 - pysar::multi_transect, 116
 - pysar::seed_data, 158
 - pysar::transect_legacy, 184
- nearest_neighbor
 - pysar::modify_network, 109
- nearest_valid
 - delayTimeseries, 19
- needed_attributes
 - pysar::unavco2json_mbtils, 217
- NoInSAR
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- nonzero_mask
 - pysar::_pysar_utilities, 51
- normalize_timeseries
 - pysar::_pysar_utilities, 51
- normalize_timeseries_old
 - pysar::_pysar_utilities, 51
- nrows
 - plot_tropcor_phase_elevation, 25
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- Num_profiles
 - pysar::multi_transect, 128
- numDates
 - delayTimeseries::timeseries, 253
- numPixels
 - delayTimeseries::timeseries, 253

- numProject
 - pysar::plot_atmDrop, [137](#)
- offset
 - pysar::plot_atmDrop, [137](#)
- onclick
 - pysar::multi_transect, [116](#)
 - pysar::transect_legacy, [184](#)
- open
 - delayTimeseries::timeseries, [251](#)
- orientation
 - pysar::tsviewer, [213](#)
- out
 - get_modis_v3, [21](#)
- outName
 - pysar::load_dem, [103](#)
 - pysar::tsviewer, [213](#)
- PLOT_TEMPLATE
 - pysar::view, [226](#)
- pair_merge
 - pysar::_network, [36](#)
- pair_sort
 - pysar::_network, [36](#)
- parallel_num
 - pysar, [29](#)
- password
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, [237](#)
- perp_baseline.py, [276](#)
- perp_baseline_ifgram2timeseries
 - pysar::_pysar_utilities, [52](#)
- perp_baseline_timeseries
 - pysar::_pysar_utilities, [52](#)
- pixList
 - pysar::plot_atmDrop, [137](#)
- plot_atmDrop.py, [277](#)
- plot_attribute_exists_for_dataset
 - pysar::add_attribute_insarmaps::InsarDatabase←
Controller, [236](#)
- plot_bar_std
 - pysar::_plot, [43](#)
- plot_coherence_history
 - pysar::_network, [36](#)
- plot_coherence_matrix
 - pysar::_network, [36](#)
- plot_dem_lalo
 - pysar::view, [223](#)
- plot_dem_yx
 - pysar::view, [223](#)
- plot_matrix
 - pysar::view, [223](#)
- plot_network
 - pysar::_network, [36](#)
- plot_network.py, [277](#)
- plot_perp_baseline_hist
 - pysar::_network, [37](#)
- plot_timeseries_errorbar
 - pysar::tsviewer, [205](#)
- plot_timeseries_event
 - pysar::tsviewer, [205](#)
- plot_timeseries_scatter
 - pysar::tsviewer, [205](#)
- plot_tropcor_phase_elevation, [22](#)
 - atr, [22](#)
 - atr2, [23](#)
 - atr3, [23](#)
 - atr4, [23](#)
 - axes, [23](#)
 - bbox_inches, [23](#)
 - data, [23](#)
 - data2, [23](#)
 - dataList, [23](#)
 - dem, [23](#)
 - dem_atr, [24](#)
 - demFile, [24](#)
 - dpi, [24](#)
 - ecmwf, [24](#)
 - ecmwfFile, [24](#)
 - epoch, [24](#)
 - fig, [24](#)
 - figsize, [24](#)
 - i, [24](#)
 - mask, [25](#)
 - maskFile, [25](#)
 - ms, [25](#)
 - msk_atr, [25](#)
 - ncols, [25](#)
 - ndx, [25](#)
 - nrows, [25](#)
 - sharex, [25](#)
 - sharey, [25](#)
 - timeseriesFile, [26](#)
 - timeseriesFile2, [26](#)
 - tropHgt, [26](#)
 - tropHgtFile, [26](#)
 - True, [26](#)
 - workDir, [26](#)
- plot_tropcor_phase_elevation.py, [278](#)
- point_on_line_with_distance_from_beginning
 - pysar::multi_transect, [117](#)
- point_with_distance_from_line
 - pysar::multi_transect, [117](#)
- prefix
 - pysar::_datetime::progress_bar, [247](#)
- prep_gamma.py, [279](#)
- prep_isce.py, [279](#)
- prep_roipac.py, [279](#)
- prepare_geometry
 - pysar::prep_isce, [143](#)
- prepare_stack
 - pysar::prep_isce, [143](#)
- prf
 - pysar::_sensor::JERS, [243](#)
- print_attributes
 - pysar::info, [90](#)
- print_hdf5_structure

- pysar::info, 90
- print_timseries_date_info
 - pysar::info, 90
- print_warning
 - pysar::seed_data, 158
- progBar
 - pysar::datetime::progress_bar, 247
- progress_bar, 244
- project_name2sensor
 - pysar::select_network, 161
- projectDir
 - pysar::plot_atmDrop, 137
- projectList
 - pysar::plot_atmDrop, 137
- pwv2zwd
 - dloadUtil, 20
- pysar, 27
 - figsize_multi, 28
 - figsize_single_max, 29
 - figsize_single_min, 29
 - miami_path, 29
 - parallel_num, 29
- pysar._datetime, 29
- pysar._gmt, 32
- pysar._network, 33
- pysar._plot, 43
- pysar._pysar_utilities, 44
- pysar._readfile, 57
- pysar._remove_surface, 63
- pysar._sensor, 64
- pysar._variance, 64
- pysar._writefile, 66
- pysar.add, 68
- pysar.add_attribute, 69
- pysar.add_attribute_insarmaps, 70
- pysar.asc_desc, 70
- pysar.baseline_error, 72
- pysar.baseline_trop, 72
- pysar.coord_glob2radar, 73
- pysar.coord_radar2glob, 73
- pysar.correct_dem, 74
- pysar.correlation_with_dem, 74
- pysar.dem_error, 75
- pysar.diff, 76
- pysar.download_ecmwf, 77
- pysar.epoch_coherence, 79
- pysar.gamma_view, 80
- pysar.generate_mask, 81
- pysar.geocode, 81
- pysar.geocode_orig, 83
- pysar.ifgram_closure, 85
- pysar.ifgram_inversion, 86
- pysar.ifgram_reconstruction, 86
- pysar.ifgram_simulation, 87
- pysar.image_math, 88
- pysar.incidence_angle, 89
- pysar.info, 89
- pysar.insar_vs_gps, 90
- pysar.insarmaps_query, 91
- pysar.json_mbtiles2insarmaps, 92
- pysar.l1, 93
- pysar.load_data, 95
- pysar.load_data_bak, 99
- pysar.load_dem, 102
- pysar.lod, 104
- pysar.look_angle, 104
- pysar.los2enu, 105
- pysar.mask, 105
- pysar.match, 107
- pysar.modify_network, 108
- pysar.multi_transect, 111
- pysar.multilook, 132
- pysar.perp_baseline, 134
- pysar.plot_atmDrop, 134
- pysar.plot_network, 138
- pysar.prep_gamma, 140
- pysar.prep_isce, 142
- pysar.prep_roipac, 144
- pysar.pysarApp, 145
- pysar.quality_map, 148
- pysar.range_distance, 148
- pysar.reference_epoch, 149
- pysar.remove_plane, 150
- pysar.rewrap, 151
- pysar.save_gmt, 152
- pysar.save_kml, 153
- pysar.save_mat, 154
- pysar.save_mat_orig, 154
- pysar.save_roipac, 155
- pysar.save_unavco, 156
- pysar.seed_data, 157
- pysar.select_network, 161
- pysar.spatial_average, 162
- pysar.spatial_filter, 163
- pysar.subset, 165
- pysar.sum_epochs, 170
- pysar.temporal_average, 170
- pysar.temporal_coherence, 171
- pysar.temporal_derivative, 172
- pysar.temporal_filter, 173
- pysar.timeseries2velocity, 173
- pysar.timeseries_rms, 175
- pysar.transect, 176
- pysar.transect_legacy, 179
- pysar.tropcor_phase_elevation, 198
- pysar.tropcor_pyaps, 199
- pysar.tropcor_pyaps_orig, 201
- pysar.tsviewer, 203
- pysar.unavco2insarmaps, 215
- pysar.unavco2json_mbtiles, 216
- pysar.unwrap_error, 218
- pysar.view, 220
- pysar::datetime
 - auto_adjust_xaxis_date, 30
 - date_index, 30
 - date_list2tbase, 30

- date_list2vector, [30](#)
- ifgram_date_list, [30](#)
- list_ifgram2date12, [31](#)
- read_date_list, [31](#)
- yymmdd, [31](#)
- yymmdd2yyyymmdd, [31](#)
- yyyymmdd, [31](#)
- yyyymmdd2years, [32](#)
- pysar::_datetime::progress_bar
 - __init__, [245](#)
 - amount, [246](#)
 - close, [245](#)
 - max, [246](#)
 - min, [247](#)
 - prefix, [247](#)
 - progBar, [247](#)
 - reset, [246](#)
 - span, [247](#)
 - start_time, [247](#)
 - suffix, [247](#)
 - update, [246](#)
 - update_amount, [246](#)
 - width, [247](#)
- pysar::_gmt
 - write_gmt_simple, [32](#)
- pysar::_network
 - auto_adjust_yaxis, [33](#)
 - azimuth_bandwidth, [34](#)
 - BASELINE_LIST_FILE, [43](#)
 - calculate_doppler_overlap, [34](#)
 - coherence_matrix, [34](#)
 - critical_perp_baseline, [34](#)
 - date12_list2index, [35](#)
 - get_date12_list, [35](#)
 - IFGRAM_LIST_FILE, [43](#)
 - igram_perp_baseline_list, [35](#)
 - incidence_angle, [35](#)
 - mode, [35](#)
 - pair_merge, [36](#)
 - pair_sort, [36](#)
 - plot_coherence_history, [36](#)
 - plot_coherence_matrix, [36](#)
 - plot_network, [36](#)
 - plot_perp_baseline_hist, [37](#)
 - range_bandwidth, [37](#)
 - read_baseline_file, [38](#)
 - read_igram_pairs, [38](#)
 - read_pairs_list, [38](#)
 - select_master_date, [38](#)
 - select_master_interferogram, [39](#)
 - select_pairs_all, [39](#)
 - select_pairs_delaunay, [39](#)
 - select_pairs_hierarchical, [39](#)
 - select_pairs_mst, [40](#)
 - select_pairs_sequential, [40](#)
 - select_pairs_star, [40](#)
 - signal2noise_ratio, [41](#)
 - threshold_coherence_based_mst, [41](#)
 - threshold_doppler_overlap, [41](#)
 - threshold_perp_baseline, [41](#)
 - threshold_temporal_baseline, [42](#)
 - wavelength, [42](#)
 - write_pairs_list, [42](#)
- pysar::_plot
 - plot_bar_std, [43](#)
- pysar::_pysar_utilities
 - add_attribute, [45](#)
 - azimuth_resolution, [45](#)
 - Bh_Bv_timeseries, [45](#)
 - check_drop_ifgram, [45](#)
 - check_file_size, [45](#)
 - check_loaded_dataset, [46](#)
 - check_parallel, [46](#)
 - check_variable_name, [46](#)
 - circle_index, [46](#)
 - dBh_dBv_timeseries, [47](#)
 - date_list, [47](#)
 - design_matrix, [47](#)
 - four_corners, [47](#)
 - generate_curls, [48](#)
 - get_file_list, [48](#)
 - get_file_stack, [48](#)
 - get_residual_rms, [48](#)
 - get_residual_std, [49](#)
 - get_spatial_average, [49](#)
 - get_triangles, [49](#)
 - glob2radar, [50](#)
 - hillshade, [50](#)
 - incidence_angle, [50](#)
 - is_file_exist, [50](#)
 - make_triangle, [51](#)
 - mode, [51](#)
 - nonzero_mask, [51](#)
 - normalize_timeseries, [51](#)
 - normalize_timeseries_old, [51](#)
 - perp_baseline_ifgram2timeseries, [52](#)
 - perp_baseline_timeseries, [52](#)
 - radar2glob, [52](#)
 - range_distance, [52](#)
 - range_resolution, [53](#)
 - spatial_average, [53](#)
 - stacking, [53](#)
 - temporal_average, [54](#)
 - timeseries_coherence, [54](#)
 - timeseries_inversion, [54](#)
 - timeseries_inversion_FGLS, [54](#)
 - timeseries_inversion_L1, [55](#)
 - timeseries_rms, [55](#)
 - timeseries_std, [55](#)
 - update_attribute_or_not, [55](#)
 - update_file, [56](#)
 - update_template_file, [56](#)
 - which, [56](#)
 - yymmdd, [56](#)
 - yymmdd2YYYYMMDD, [57](#)
 - yyyymmdd, [57](#)

pysar::_readfile
 attribute_gamma2roipac, 58
 attribute_isce2roipac, 58
 check_variable_name, 58
 is_plot_attribute, 58
 multi_dataset_hdf5_file, 63
 multi_group_hdf5_file, 63
 read, 58
 read_GPS_USGS, 61
 read_attribute, 59
 read_complex_float32, 59
 read_complex_int16, 60
 read_dem, 60
 read_flag, 60
 read_float32, 60
 read_gamma_par, 61
 read_isce_xml, 61
 read_multiple, 62
 read_real_float32, 62
 read_real_int16, 62
 read_roipac_rsc, 62
 read_template, 63
 single_dataset_hdf5_file, 63
 pysar::_remove_surface
 remove_data_multiple_surface, 64
 remove_data_surface, 64
 remove_surface, 64
 pysar::_sensor::JERS
 __init__, 243
 bandwidth, 243
 center_frequency, 243
 file, 243
 prf, 243
 swath_width, 244
 pysar::_variance
 bin_variance, 65
 get_distance, 65
 get_lat_lon, 65
 sample_data, 65
 structure_function, 65
 pysar::_writefile
 write, 66
 write_complex64, 66
 write_complex_int16, 67
 write_dem, 67
 write_float32, 67
 write_real_float32, 67
 write_real_int16, 67
 write_roipac_rsc, 68
 pysar::add
 add_files, 68
 add_matrix, 68
 cmdLineParse, 69
 EXAMPLE, 69
 main, 69
 pysar::add_attribute
 main, 69
 usage, 70
 pysar::add_attribute_insarmaps
 build_parser, 70
 main, 70
 pysar::add_attribute_insarmaps::InsarDatabaseController
 __init__, 234
 add_attribute, 234
 add_plot_attribute, 234
 attribute_exists_for_dataset, 235
 close, 235
 con, 237
 connect, 235
 create_area_table_if_not_exists, 235
 cursor, 237
 db, 237
 get_dataset_id, 235
 get_dataset_names, 235
 host, 237
 index_table_on, 235
 insert_dataset_into_area_table, 236
 password, 237
 plot_attribute_exists_for_dataset, 236
 remove_dataset_if_there, 236
 remove_point_table_if_there, 236
 table_exists, 236
 username, 237
 pysar::add_attribute_insarmaps::InsarDatasetController
 __init__, 240
 bodyOutput, 241
 curl_login, 240
 headersOutput, 241
 remove_mbtils, 240
 serverPassword, 241
 serverUsername, 241
 setup_curl, 240
 upload_mbtils, 240
 pysar::asc_desc
 cmdLineParse, 71
 EXAMPLE, 71
 get_overlap_lalo, 71
 main, 71
 REFERENCE, 71
 pysar::baseline_error
 main, 72
 to_percent, 72
 usage, 72
 pysar::baseline_trop
 main, 72
 to_percent, 72
 usage, 73
 pysar::coord_glob2radar
 main, 73
 usage, 73
 pysar::coord_radar2glob
 main, 73
 usage, 74
 pysar::correct_dem
 main, 74

- usage, 74
- pysar::correlation_with_dem
 - main, 74
 - usage, 75
- pysar::dem_error
 - cmdLineParse, 75
 - EXAMPLE, 76
 - get_exclude_date, 75
 - main, 75
 - REFERENCE, 76
 - read_template2inps, 75
 - TEMPLATE, 76
- pysar::diff
 - cmdLineParse, 76
 - diff_data, 76
 - diff_file, 77
 - main, 77
 - usage, 77
- pysar::download_ecmwf
 - date, 78
 - date_str, 78
 - dateListFile, 78
 - days, 78
 - end_date, 78
 - f, 78
 - hour, 78
 - jobCmd, 78
 - maxJobNum, 78
 - runFile, 79
 - start_date, 79
 - step, 79
 - tropCmd, 79
- pysar::epoch_coherence
 - cmdLineParse, 79
 - EXAMPLE, 80
 - epoch_coherence_file, 79
 - main, 80
- pysar::gamma_view
 - main, 80
 - usage, 80
- pysar::generate_mask
 - cmdLineParse, 81
 - EXAMPLE, 81
 - main, 81
- pysar::geocode
 - cmdLineParse, 82
 - EXAMPLE, 83
 - geocode_attribute_with_geo_lut, 82
 - geocode_file_with_geo_lut, 82
 - main, 82
 - update_attribute4isce, 83
- pysar::geocode_orig
 - cmdLineParse, 83
 - EXAMPLE, 85
 - geocode_attribute_with_geo_lookup_table, 84
 - geocode_file_with_geo_lookup_table, 84
 - main, 84
 - update_attribute4isce, 84
- pysar::ifgram_closure
 - main, 85
 - usage, 85
- pysar::ifgram_inversion
 - cmdLineParse, 86
 - EXAMPLE, 86
 - main, 86
- pysar::ifgram_reconstruction
 - main, 86
 - usage, 87
- pysar::ifgram_simulation
 - cmdLineParse, 87
 - EXAMPLE, 87
 - main, 87
- pysar::image_math
 - cmdLineParse, 88
 - data_operation, 88
 - EXAMPLE, 89
 - file_operation, 88
 - main, 88
- pysar::incidence_angle
 - main, 89
 - usage, 89
- pysar::info
 - main, 89
 - print_attributes, 90
 - print_hdf5_structure, 90
 - print_timseries_date_info, 90
 - usage, 90
- pysar::insar_vs_gps
 - find_row_column, 90
 - main, 91
 - nearest, 91
 - readGPSfile, 91
 - usage, 91
- pysar::insarmaps_query
 - build_parser, 92
 - buildURL, 92
 - main, 92
- pysar::insarmaps_query::BasicHTTP
 - get, 231
- pysar::json_mbtiles2insarmaps
 - build_parser, 92
 - dbHost, 93
 - dbPassword, 93
 - dbUsername, 93
 - get_unavco_name, 92
 - main, 93
 - upload_insarmaps_metadata, 93
 - upload_json, 93
- pysar::l1
 - __MOSEK, 95
 - l1, 94
 - l1blas, 94
 - l1mosek, 94
 - l1mosek2, 94
 - task, 95
 - x, 95

pysar::load_data
 auto_path_miami, 96
 check_existed_hdf5_file, 96
 check_file_size, 96
 cmdLineParse, 96
 copy_file, 96
 EXAMPLE, 98
 load_data_from_template, 97
 load_file, 97
 load_multi_group_hdf5, 97
 load_single_dataset_hdf5, 97
 main, 98
 mode, 98
 TEMPLATE, 98

pysar::load_data_bak
 auto_path_miami, 99
 check_existed_hdf5_file, 99
 check_file_size, 99
 cmdLineParse, 100
 copy_file, 100
 EXAMPLE, 102
 load_data_from_template, 100
 load_file, 100
 main, 100
 mode, 101
 roipac2multi_group_hdf5, 101
 roipac2single_dataset_hdf5, 101
 roipac_nonzero_mask, 101
 TEMPLATE, 102

pysar::load_dem
 amp, 102
 compression, 102
 data, 103
 dem, 103
 demFile, 103
 demRsc, 103
 dset, 103
 ext, 103
 group, 103
 h5, 103
 outName, 103

pysar::lod
 correct_lod_file, 104
 main, 104
 usage, 104

pysar::look_angle
 main, 104
 usage, 105

pysar::los2enu
 main, 105
 usage, 105

pysar::mask
 cmdLineParse, 106
 EXAMPLE, 107
 main, 106
 mask_file, 106
 mask_matrix, 106
 update_mask, 106

pysar::match
 cmdLineParse, 107
 corners, 107
 EXAMPLE, 108
 main, 107
 manual_offset_estimate, 107
 match_two_files, 108
 nearest, 108

pysar::modify_network
 cmdLineParse, 109
 EXAMPLE, 110
 main, 109
 manual_select_pairs_to_remove, 109
 modify_file_date12_list, 109
 nearest_neighbor, 109
 read_template2inps, 110
 reset_pairs, 110
 TEMPLATE, 110

pysar::multi_transect
 alpha, 118
 avglnSAR, 118
 ax, 118
 axes, 118
 axes2, 118
 axis, 118
 c, 118
 c_prof_edge, 119
 cf, 119
 check_result, 119
 check_result2, 119
 check_st_in_box, 114
 check_st_in_box2, 114
 cid, 119
 color, 119
 D, 119
 dataset, 119
 df0_km, 120
 dg, 120
 dist_point_from_line, 114
 DistGPS, 120
 dms2d, 114
 dp, 120
 dset, 120
 DX, 120
 dx, 120
 DY, 120
 dy, 120
 Fault_lat, 121
 Fault_lon, 121
 FaultCoords, 121
 FaultLine, 121
 fig, 121
 fig2, 121
 figName, 121
 fileExtension, 121
 fileName, 122
 find_row_column, 114
 fontsize, 122

- GPS_in_bound, 122
- GPS_in_bound_st, 122
- GPS_lat, 122
- GPS_lon, 122
- GPS_station, 122
- GPSx, 123
- GPSxx, 123
- GPSy, 123
- GPSyy, 123
- GPS, 122
- get_intersect, 115
- get_lat_lon, 115
- get_start_end_point, 115
- get_transect, 115
- gps_to_LOS, 115
- gpsFile, 122
- gpsLOS_ref, 123
- gpsLOS, 123
- gx, 123
- gy, 123
- h5file_theta, 123
- hbound, 124
- heading, 124
- IDXref, 124
- IDYref, 124
- IDX, 124
- IDY, 124
- idx, 124
- idxRef, 124
- insarData, 124
- Lat, 125
- lat, 125
- Lat0, 125
- Lat1, 125
- lat_all, 125
- lat_step, 125
- lat_transect, 125
- lbound, 125
- Length, 126
- length, 126
- line, 116
- linewidth, 126
- Lon, 126
- lon, 126
- Lon0, 126
- Lon1, 126
- lon_all, 126
- lon_step, 127
- lon_transect, 127
- m, 127
- m1, 127
- m_prof_edge, 127
- main, 116
- matFile, 127
- mf, 127
- mfc, 127
- ms, 127
- nanmean, 116
- nanstd, 116
- nearest, 116
- NoInSAR, 128
- nrows, 128
- Num_profiles, 128
- onclick, 116
- point_on_line_with_distance_from_beginning, 117
- point_with_distance_from_line, 117
- read_fault_coords, 117
- readGPSfile, 117
- redGPSfile, 117
- redGPSfile_cmm4, 117
- Se, 128
- Sn, 128
- Stations, 128
- stationsList, 128
- stdInSAR, 128
- theta, 129
- transect, 129
- transect_lat, 129
- transect_lon, 129
- unitVec, 129
- usage, 118
- Ve, 129
- Vn, 129
- Width, 129
- x, 129
- X0, 130
- x0, 130
- X1, 130
- x1, 130
- XX0, 130
- XX1, 131
- xc, 130
- Xf0, 130
- Xf1, 130
- xlim, 130
- y, 131
- Y0, 131
- y0, 131
- Y1, 131
- y1, 131
- YY0, 132
- YY1, 132
- yc, 131
- Yf0, 131
- Yf1, 131
- ylim, 132
- zi, 132
- pysar::multilook
 - cmdLineParse, 132
 - EXAMPLE, 133
 - main, 133
 - multilook_attribute, 133
 - multilook_file, 133
 - multilook_matrix, 133
- pysar::perp_baseline
 - main, 134

- usage, 134
- pysar::plot_atmDrop
 - alpha, 135
 - ax1, 135
 - ax2, 135
 - bbox_inches, 135
 - c, 135
 - cbar, 135
 - dateList, 135
 - dateList6, 135
 - dates, 135
 - datevector, 136
 - fig, 136
 - figsize, 136
 - fl, 136
 - fontsize, 136
 - idxMean, 136
 - idxPix, 136
 - key, 136
 - line_s, 137
 - lineNum, 137
 - lines, 137
 - meanList, 137
 - numProject, 137
 - offset, 137
 - pixList, 137
 - projectDir, 137
 - projectList, 137
 - s, 138
 - sc1, 138
 - sc2, 138
 - transparent, 138
 - vmax, 138
 - vmin, 138
- pysar::plot_network
 - BL_LIST, 139
 - cmdLineParse, 139
 - DATE12_LIST, 139
 - EXAMPLE, 139
 - main, 139
- pysar::prep_gamma
 - cmdLineParse, 140
 - DESCRIPTION, 142
 - EXAMPLE, 142
 - extract_attribute_dem_geo, 140
 - extract_attribute_dem_radar, 140
 - extract_attribute_interferogram, 140
 - extract_attribute_lookup_table, 141
 - get_lalo_ref, 141
 - get_perp_baseline, 141
 - main, 141
- pysar::prep_isce
 - baselineTimeseries, 142
 - cmdLineParse, 142
 - createParser, 143
 - extractIsceMetadata, 143
 - GDAL2NUMPY_DATATYPE, 144
 - main, 143
 - prepare_geometry, 143
 - prepare_stack, 143
 - read_baseline, 143
 - write_rsc, 143
- pysar::prep_roipac
 - cmdLineParse, 144
 - DESCRIPTION, 145
 - EXAMPLE, 145
 - extract_attribute, 144
 - main, 145
- pysar::pysarApp
 - check_geocode_file, 146
 - check_subset_file, 146
 - cmdLineParse, 146
 - create_subset_dataset, 146
 - EXAMPLE, 147
 - LOGO, 147
 - main, 146
 - multilook_dataset, 147
 - subset_dataset, 147
 - TEMPLATE, 147
 - UM_FILE_STRUCT, 148
- pysar::quality_map
 - main, 148
 - usage, 148
- pysar::range_distance
 - main, 148
 - usage, 148
- pysar::reference_epoch
 - cmdLineParse, 149
 - EXAMPLE, 150
 - main, 149
 - read_template2inps, 149
 - ref_date_attribute, 149
 - ref_date_file, 150
 - TEMPLATE, 150
- pysar::remove_plane
 - cmdLineParse, 150
 - EXAMPLE, 151
 - main, 151
- pysar::rewrap
 - main, 151
 - rewrap, 151
 - usage, 151
- pysar::save_gmt
 - cmdLineParse, 152
 - EXAMPLE, 153
 - get_geo_lat_lon, 152
 - main, 152
 - write_grd_file, 152
- pysar::save_kml
 - cmdLineParse, 153
 - EXAMPLE, 154
 - main, 153
 - write_kmz_file, 153
- pysar::save_mat
 - main, 154
 - usage, 154

- yyyyymmdd2years, [154](#)
- pysar::save_mat_orig
 - main, [155](#)
 - usage, [155](#)
 - yyyyymmdd2years, [155](#)
- pysar::save_roipac
 - main, [155](#)
 - usage, [155](#)
- pysar::save_unavco
 - CPX_ZERO, [157](#)
 - cmdLineParse, [156](#)
 - EXAMPLE, [157](#)
 - FLOAT_ZERO, [157](#)
 - get_mission_name, [156](#)
 - get_unavco_filename, [156](#)
 - INT_ZERO, [157](#)
 - main, [156](#)
 - metadata_pysar2unavco, [157](#)
- pysar::seed_data
 - cmdLineParse, [158](#)
 - EXAMPLE, [160](#)
 - main, [158](#)
 - manual_select_reference_yx, [158](#)
 - NOTE, [160](#)
 - nearest, [158](#)
 - print_warning, [158](#)
 - random_select_reference_yx, [159](#)
 - read_seed_reference2inps, [159](#)
 - read_seed_template2inps, [159](#)
 - remove_reference_pixel, [159](#)
 - seed_attributes, [159](#)
 - seed_file_inps, [159](#)
 - seed_file_reference_value, [160](#)
 - select_max_coherence_yx, [160](#)
 - TEMPLATE, [160](#)
- pysar::select_network
 - cmdLineParse, [161](#)
 - EXAMPLE, [162](#)
 - log, [161](#)
 - METHOD, [162](#)
 - main, [161](#)
 - project_name2sensor, [161](#)
 - REFERENCE, [162](#)
 - read_template2inps, [161](#)
 - sar_sensor_list, [162](#)
 - TEMPLATE, [162](#)
- pysar::spatial_average
 - cmdLineParse, [163](#)
 - EXAMPLE, [163](#)
 - main, [163](#)
- pysar::spatial_filter
 - cmdLineParse, [163](#)
 - EXAMPLE, [164](#)
 - filter_data, [164](#)
 - filter_file, [164](#)
 - main, [164](#)
- pysar::subset
 - bbox_geo2radar, [165](#)
 - bbox_radar2geo, [165](#)
 - box_geo2pixel, [166](#)
 - box_pixel2geo, [166](#)
 - check_box_within_data_coverage, [166](#)
 - cmdLineParse, [166](#)
 - coord_geo2radar, [167](#)
 - coord_radar2geo, [167](#)
 - EXAMPLE, [170](#)
 - get_box_overlap_index, [167](#)
 - get_coverage_box, [167](#)
 - main, [167](#)
 - read_subset_template2box, [168](#)
 - subset_attribute, [168](#)
 - subset_box2inps, [168](#)
 - subset_file, [168](#)
 - subset_file_list, [169](#)
 - subset_input_dict2box, [169](#)
- pysar::sum_epochs
 - main, [170](#)
 - usage, [170](#)
- pysar::temporal_average
 - main, [170](#)
 - usage, [171](#)
- pysar::temporal_coherence
 - DESCRIPTION, [172](#)
 - EXAMPLE, [172](#)
 - main, [171](#)
 - REFERENCE, [172](#)
 - temporal_coherence, [171](#)
 - USAGE, [172](#)
 - usage, [171](#)
- pysar::temporal_derivative
 - main, [172](#)
 - usage, [172](#)
- pysar::temporal_filter
 - cmdLineParse, [173](#)
 - EXAMPLE, [173](#)
 - main, [173](#)
- pysar::timeseries2velocity
 - cmdLineParse, [174](#)
 - DROP_DATE_TXT, [175](#)
 - EXAMPLE, [175](#)
 - get_exclude_date, [174](#)
 - get_velocity_filename, [174](#)
 - main, [174](#)
 - read_template2inps, [174](#)
 - TEMPLATE, [175](#)
- pysar::timeseries_rms
 - cmdLineParse, [175](#)
 - EXAMPLE, [176](#)
 - main, [176](#)
 - read_template2inps, [176](#)
 - TEMPLATE, [176](#)
- pysar::transect
 - cmdLineParse, [177](#)
 - EXAMPLE, [179](#)
 - get_scale_from_disp_unit, [177](#)
 - main, [177](#)

- manual_select_start_end_point, 177
- read_lonlat_file, 177
- transect_lalo, 177
- transect_list, 178
- transect_yx, 178
- pysar::transect_legacy
 - alpha, 185
 - avglnSAR, 185
 - ax, 185
 - axes, 185
 - axes2, 185
 - axis, 185
 - c, 186
 - c_prof_edge, 186
 - cf, 186
 - check_result, 186
 - check_result2, 186
 - check_st_in_box, 181
 - check_st_in_box2, 182
 - cid, 186
 - color, 186
 - D, 186
 - dataset, 187
 - df0, 187
 - df0_km, 187
 - df1, 187
 - dg, 187
 - dist_point_from_line, 182
 - DistGPS, 187
 - dms2d, 182
 - dset, 187
 - DX, 187
 - dx, 187
 - DY, 188
 - dy, 188
 - earth_radius, 188
 - fault_loc, 188
 - FaultLine, 188
 - fig, 188
 - fig2, 188
 - figName, 188
 - fileExtension, 189
 - fileName, 189
 - find_row_column, 182
 - fontsize, 189
 - GPS_in_bound, 189
 - GPS_in_bound_st, 189
 - GPS_lat, 189
 - GPS_lon, 189
 - GPS_station, 190
 - GPSx, 190
 - GPSxx, 190
 - GPSy, 190
 - GPSyy, 190
 - GPS, 189
 - get_intersect, 183
 - get_lat_lon, 183
 - get_transect, 183
 - gps_to_LOS, 183
 - gpsFile, 190
 - gpsLOS_ref, 190
 - gpsLOS, 190
 - gx, 190
 - gy, 191
 - h5file_theta, 191
 - hbound, 191
 - heading, 191
 - IDXref, 191
 - IDYref, 192
 - IDX, 191
 - IDY, 191
 - idx, 191
 - idxRef, 191
 - Info_aboutFault, 192
 - insarData, 192
 - Lat, 192
 - lat, 192
 - lat_all, 192
 - lat_step, 192
 - lat_transect, 192
 - lbound, 192
 - Length, 193
 - length, 193
 - line, 183
 - linewidth, 193
 - Lon, 193
 - lon, 193
 - lon_all, 193
 - lon_step, 193
 - lon_transect, 193
 - m, 194
 - m1, 194
 - m_prof_edge, 194
 - main, 184
 - matFile, 194
 - mf, 194
 - mfc, 194
 - mp, 194
 - ms, 194
 - nanmean, 184
 - nanstd, 184
 - nearest, 184
 - NoInSAR, 195
 - nrows, 195
 - onclick, 184
 - readGPSfile, 184
 - redGPSfile, 184
 - redGPSfile_cmm4, 185
 - Se, 195
 - Sn, 195
 - Stations, 195
 - stationsList, 195
 - stdlnSAR, 195
 - theta, 195
 - transect, 196
 - transect_lat, 196

- transect_lon, 196
- unitVec, 196
- Usage, 185
- Ve, 196
- Vn, 196
- Width, 196
- x, 196
- X0, 197
- x0, 196
- X1, 197
- x1, 197
- XX0, 197
- XX1, 197
- xc, 197
- y, 197
- Y0, 197
- y0, 197
- Y1, 198
- y1, 198
- YY0, 198
- YY1, 198
- yc, 198
- ylim, 198
- zi, 198
- pysar::tropcor_phase_elevation
 - cmdLineParse, 199
 - EXAMPLE, 199
 - main, 199
 - REFERENCE, 199
- pysar::tropcor_pyaps
 - closest_weather_product_time, 200
 - cmdLineParse, 200
 - dload_grib, 200
 - EXAMPLE, 201
 - get_delay, 200
 - main, 200
 - REFERENCE, 201
 - TEMPLATE, 201
- pysar::tropcor_pyaps_orig
 - closest_weather_product_time, 201
 - cmdLineParse, 202
 - EXAMPLE, 202
 - get_delay, 202
 - main, 202
 - REFERENCE, 202
 - TEMPLATE, 202
- pysar::tsviewer
 - atr, 206
 - ax_time, 206
 - ax_ts, 206
 - ax_v, 207
 - axisbg, 207
 - bbox_inches, 207
 - cbar, 207
 - cid, 207
 - clim, 207
 - cmap, 207
 - cmdLineParse, 204
 - colormap, 207
 - d_ts, 208
 - d_v, 208
 - data_lim, 208
 - date_num, 208
 - dateList, 208
 - dates, 208
 - delimiter, 208
 - dpi, 208
 - dtype, 208
 - e_ts, 209
 - EXAMPLE, 210
 - ecolor, 209
 - error_file, 209
 - error_fileContent, 209
 - error_ts, 209
 - ex_date, 209
 - ex_date_list, 209
 - ex_dates, 209
 - ex_error_ts, 209
 - ex_idx_list, 210
 - facecolor, 210
 - fig_base, 210
 - fig_ts, 210
 - fig_v, 210
 - figsize, 210
 - flip_ud, 210
 - fmt, 211
 - format_coord, 205, 211
 - h5, 211
 - header_info, 211
 - img, 211
 - inps, 211
 - input_ex_date, 211
 - k, 211
 - lat, 212
 - lat_step, 212
 - left_lr, 212
 - length, 212
 - lon, 212
 - lon_step, 212
 - lrlat, 212
 - lrlon, 212
 - markededgecolor, 212
 - mask, 213
 - ms, 213
 - orientation, 213
 - outName, 213
 - plot_timeseries_errorbar, 205
 - plot_timeseries_event, 205
 - plot_timeseries_scatter, 205
 - read_timeseries_lalo, 205
 - read_timeseries_yx, 205
 - ref_d_v, 213
 - ref_yx, 213
 - time_slider_update, 206
 - tims, 213
 - transparent, 213

- True, 213
- tslider, 214
- ullat, 214
- ullon, 214
- unit_fac, 214
- update_timeseries, 206
- valinit, 214
- width, 214
- x, 214
- y, 214
- ylim, 214
- yticks, 215
- yx, 215
- pysar::unavco2insarmaps
 - build_parser, 215
 - get_H5_filename, 215
 - main, 215
- pysar::unavco2json_mbtiles
 - build_parser, 216
 - convert_data, 216
 - get_date, 216
 - get_decimal_date, 216
 - main, 216
 - make_json_file, 217
 - needed_attributes, 217
 - region_name_from_project_name, 217
 - serialize_dictionary, 217
- pysar::unwrap_error
 - bridging_data, 218
 - cmdLineParse, 218
 - DESCRIPTION, 219
 - EXAMPLE, 219
 - main, 218
 - REFERENCE, 220
 - unwrap_error_correction_bridging, 218
 - unwrap_error_correction_phase_closure, 219
- pysar::view
 - add_inner_title, 221
 - auto_figure_title, 221
 - auto_flip_direction, 221
 - auto_row_col_num, 221
 - check_colormap_input, 222
 - check_multilook_input, 222
 - cmdLineParse, 222
 - EXAMPLE, 225
 - get_epoch_full_list_from_input, 222
 - main, 222
 - PLOT_TEMPLATE, 226
 - plot_dem_lalo, 223
 - plot_dem_yx, 223
 - plot_matrix, 223
 - round_to_1, 224
 - scale_data2disp_unit, 224
 - scale_data4disp_unit_and_rewrap, 224
 - update_matrix_with_plot_inps, 225
 - update_plot_inps_with_display_setting_file, 225
 - update_plot_inps_with_meta_dict, 225
- pysar::view::Basemap2
 - auto_lalo_sequence, 229
 - draw_lalo_label, 229
 - drawscale, 230
- pysarApp.py, 280
- quality_map.py, 280
- README.md, 281
- REFERENCE
 - pysar::asc_desc, 71
 - pysar::dem_error, 76
 - pysar::select_network, 162
 - pysar::temporal_coherence, 172
 - pysar::tropcor_phase_elevation, 199
 - pysar::tropcor_pyaps, 201
 - pysar::tropcor_pyaps_orig, 202
 - pysar::unwrap_error, 220
- radar2glob
 - pysar::_pysar_utilities, 52
- random_select_reference_yx
 - pysar::seed_data, 159
- range_bandwidth
 - pysar::_network, 37
- range_distance
 - pysar::_pysar_utilities, 52
- range_distance.py, 281
- range_resolution
 - pysar::_pysar_utilities, 53
- read
 - pysar::_readfile, 58
- read_GPS_USGS
 - pysar::_readfile, 61
- read_attribute
 - pysar::_readfile, 59
- read_baseline
 - pysar::prep_isce, 143
- read_baseline_file
 - pysar::_network, 38
- read_complex_float32
 - pysar::_readfile, 59
- read_complex_int16
 - pysar::_readfile, 60
- read_date_list
 - pysar::_datetime, 31
- read_dem
 - pysar::_readfile, 60
- read_fault_coords
 - pysar::multi_transect, 117
- read_flag
 - pysar::_readfile, 60
- read_float32
 - pysar::_readfile, 60
- read_gamma_par
 - pysar::_readfile, 61
- read_igram_pairs
 - pysar::_network, 38
- read_isce_xml
 - pysar::_readfile, 61
- read_lonlat_file

- pysar::transect, 177
- read_modis
 - dloadUtil, 20
- read_multiple
 - pysar::_readfile, 62
- read_pairs_list
 - pysar::_network, 38
- read_real_float32
 - pysar::_readfile, 62
- read_real_int16
 - pysar::_readfile, 62
- read_roipac_rsc
 - pysar::_readfile, 62
- read_seed_reference2inps
 - pysar::seed_data, 159
- read_seed_template2inps
 - pysar::seed_data, 159
- read_subset_template2box
 - pysar::subset, 168
- read_template
 - pysar::_readfile, 63
- read_template2inps
 - pysar::dem_error, 75
 - pysar::modify_network, 110
 - pysar::reference_epoch, 149
 - pysar::select_network, 161
 - pysar::timeseries2velocity, 174
 - pysar::timeseries_rms, 176
- read_timeseries_lalo
 - pysar::tsviewer, 205
- read_timeseries_yx
 - pysar::tsviewer, 205
- readGPSfile
 - pysar::insar_vs_gps, 91
 - pysar::multi_transect, 117
 - pysar::transect_legacy, 184
- redGPSfile
 - pysar::multi_transect, 117
 - pysar::transect_legacy, 184
- redGPSfile_cmm4
 - pysar::multi_transect, 117
 - pysar::transect_legacy, 185
- ref_d_v
 - pysar::tsviewer, 213
- ref_date_attribute
 - pysar::reference_epoch, 149
- ref_date_file
 - pysar::reference_epoch, 150
- ref_yx
 - pysar::tsviewer, 213
- reference_epoch.py, 281
- region_name_from_project_name
 - pysar::unavco2json_mbtiles, 217
- relative_std
 - delayTimeseries::timeseries, 253
- relative_std_velocity
 - delayTimeseries::timeseries, 253
- remove_data_multiple_surface
 - pysar::_remove_surface, 64
- remove_data_surface
 - pysar::_remove_surface, 64
- remove_dataset_if_there
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 236
- remove_mbtiles
 - pysar::add_attribute_insarmaps::InsarDataset↔
Controller, 240
- remove_plane.py, 281
- remove_point_table_if_there
 - pysar::add_attribute_insarmaps::InsarDatabase↔
Controller, 236
- remove_reference_pixel
 - pysar::seed_data, 159
- remove_surface
 - pysar::_remove_surface, 64
- reset
 - pysar::_datetime::progress_bar, 246
- reset_pairs
 - pysar::modify_network, 110
- rewrap
 - pysar::rewrap, 151
- rewrap.py, 282
- roipac2multi_group_hdf5
 - pysar::load_data_bak, 101
- roipac2single_dataset_hdf5
 - pysar::load_data_bak, 101
- roipac_nonzero_mask
 - pysar::load_data_bak, 101
- round_to_1
 - pysar::view, 224
- runFile
 - pysar::download_ecmwf, 79
- s
 - pysar::plot_atmDrop, 138
- SAR-Sensor-Parameter.md, 282
- sample
 - delayTimeseries::timeseries, 251
- sample_data
 - pysar::_variance, 65
- sar_sensor_list
 - pysar::select_network, 162
- save_gmt.py, 282
- save_kml.py, 282
- save_mat.py, 283
- save_mat_orig.py, 283
- save_roipac.py, 283
- save_unavco.py, 283
- sc1
 - pysar::plot_atmDrop, 138
- sc2
 - pysar::plot_atmDrop, 138
- scale_data2disp_unit
 - pysar::view, 224
- scale_data4disp_unit_and_rewrap
 - pysar::view, 224
- Se

- pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- seed_attributes
 - pysar::seed_data, 159
- seed_data.py, 284
- seed_file_inps
 - pysar::seed_data, 159
- seed_file_reference_value
 - pysar::seed_data, 160
- select_master_date
 - pysar::_network, 38
- select_master_interferogram
 - pysar::_network, 39
- select_max_coherence_yx
 - pysar::seed_data, 160
- select_network.py, 284
- select_pairs_all
 - pysar::_network, 39
- select_pairs_delaunay
 - pysar::_network, 39
- select_pairs_hierarchical
 - pysar::_network, 39
- select_pairs_mst
 - pysar::_network, 40
- select_pairs_sequential
 - pysar::_network, 40
- select_pairs_star
 - pysar::_network, 40
- serialize_dictionary
 - pysar::unavco2json_mbtiles, 217
- serverPassword
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 241
- serverUsername
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 241
- setup_curl
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 240
- sharex
 - plot_tropcor_phase_elevation, 25
- sharey
 - plot_tropcor_phase_elevation, 25
- signal2noise_ratio
 - pysar::_network, 41
- single_dataset_hdf5_file
 - pysar::_readfile, 63
- Sn
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- span
 - pysar::_datetime::progress_bar, 247
- spatial_average
 - pysar::_pysar_utilities, 53
- spatial_average.py, 285
- spatial_filter.py, 285
- stacking
 - pysar::_pysar_utilities, 53
- start_date
 - pysar::download_ecmwf, 79
- start_time
 - pysar::_datetime::progress_bar, 247
- start_time_main
 - get_modis_v3, 21
- Stations
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- stationsList
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- statistics
 - delayTimeseries::timeseries, 251
 - troposphere_uncertainty, 227
- std_timeseries
 - delayTimeseries::timeseries, 251
- std_velocity
 - delayTimeseries::timeseries, 251
- stdInSAR
 - pysar::multi_transect, 128
 - pysar::transect_legacy, 195
- step
 - pysar::download_ecmwf, 79
- structure_function
 - pysar::_variance, 65
- subset.py, 285
- subset_attribute
 - pysar::subset, 168
- subset_box2inps
 - pysar::subset, 168
- subset_dataset
 - pysar::pysarApp, 147
- subset_file
 - pysar::subset, 168
- subset_file_list
 - pysar::subset, 169
- subset_input_dict2box
 - pysar::subset, 169
- suffix
 - pysar::_datetime::progress_bar, 247
- sum_epochs.py, 286
- swath_width
 - pysar::_sensor::JERS, 244
- TEMPLATE
 - pysar::dem_error, 76
 - pysar::load_data, 98
 - pysar::load_data_bak, 102
 - pysar::modify_network, 110
 - pysar::pysarApp, 147
 - pysar::reference_epoch, 150
 - pysar::seed_data, 160
 - pysar::select_network, 162
 - pysar::timeseries2velocity, 175
 - pysar::timeseries_rms, 176
 - pysar::tropcor_pyaps, 201
 - pysar::tropcor_pyaps_orig, 202
- table_exists

pysar::add_attribute_insarmaps::InsarDatabase↔
 Controller, 236
 task
 pysar::l1, 95
 temporal_average
 pysar::_pysar_utilities, 54
 temporal_average.py, 286
 temporal_coherence
 pysar::temporal_coherence, 171
 temporal_coherence.py, 287
 temporal_derivative.py, 287
 temporal_filter.py, 287
 theta
 pysar::multi_transect, 129
 pysar::transect_legacy, 196
 threshold_coherence_based_mst
 pysar::_network, 41
 threshold_doppler_overlap
 pysar::_network, 41
 threshold_perp_baseline
 pysar::_network, 41
 threshold_temporal_baseline
 pysar::_network, 42
 time_elapsed
 get_modis_v3, 22
 time_slider_update
 pysar::tsviewer, 206
 timeseries, 248
 timeseries2velocity.py, 288
 timeseries_coherence
 pysar::_pysar_utilities, 54
 timeseries_inversion
 pysar::_pysar_utilities, 54
 timeseries_inversion_FGLS
 pysar::_pysar_utilities, 54
 timeseries_inversion_L1
 pysar::_pysar_utilities, 55
 timeseries_rms
 pysar::_pysar_utilities, 55
 timeseries_rms.py, 288
 timeseries_std
 pysar::_pysar_utilities, 55
 timeseriesFile
 plot_tropcor_phase_elevation, 26
 timeseriesFile2
 plot_tropcor_phase_elevation, 26
 tims
 pysar::tsviewer, 213
 to_percent
 pysar::baseline_error, 72
 pysar::baseline_trop, 72
 transect
 pysar::multi_transect, 129
 pysar::transect_legacy, 196
 transect.py, 288
 transect_lalo
 pysar::transect, 177
 transect_lat
 pysar::multi_transect, 129
 pysar::transect_legacy, 196
 transect_legacy.py, 289
 transect_list
 pysar::transect, 178
 transect_lon
 pysar::multi_transect, 129
 pysar::transect_legacy, 196
 transect_yx
 pysar::transect, 178
 transparent
 pysar::plot_atmDrop, 138
 pysar::tsviewer, 213
 tropCmd
 pysar::download_ecmwf, 79
 tropHgt
 plot_tropcor_phase_elevation, 26
 tropHgtFile
 plot_tropcor_phase_elevation, 26
 tropcor_phase_elevation.py, 292
 tropcor_pyaps.py, 292
 tropcor_pyaps_orig.py, 293
 troposphere_uncertainty, 226
 cmdLineParse, 227
 download, 227
 EXAMPLE, 228
 estimate_seasonal, 227
 main, 227
 statistics, 227
 velocity_uncertainty, 227
 velocity_uncertainty_vs_distance, 227
 troposphere_uncertainty.py, 293
 True
 plot_tropcor_phase_elevation, 26
 pysar::tsviewer, 213
 tslider
 pysar::tsviewer, 214
 tsviewer.py, 294
 UM_FILE_STRUCT
 pysar::pysarApp, 148
 UNAVCO-InSAR-Archive.md, 295
 USAGE
 pysar::temporal_coherence, 172
 ullat
 pysar::tsviewer, 214
 ullon
 pysar::tsviewer, 214
 unavco2insarmaps.py, 295
 unavco2json_mbtiles.py, 296
 uncertainty_vs_distance
 delayTimeseries::timeseries, 251
 unit_fac
 pysar::tsviewer, 214
 unitVec
 pysar::multi_transect, 129
 pysar::transect_legacy, 196
 unwrap_error.py, 296
 unwrap_error_correction_bridging

- pysar::unwrap_error, 218
- unwrap_error_correction_phase_closure
 - pysar::unwrap_error, 219
- update
 - pysar::_datetime::progress_bar, 246
- update_amount
 - pysar::_datetime::progress_bar, 246
- update_attribute4isce
 - pysar::geocode, 83
 - pysar::geocode_orig, 84
- update_attribute_or_not
 - pysar::_pysar_utilities, 55
- update_file
 - pysar::_pysar_utilities, 56
- update_mask
 - pysar::mask, 106
- update_matrix_with_plot_inps
 - pysar::view, 225
- update_plot_inps_with_display_setting_file
 - pysar::view, 225
- update_plot_inps_with_meta_dict
 - pysar::view, 225
- update_template_file
 - pysar::_pysar_utilities, 56
- update_timeseries
 - pysar::tsviewer, 206
- upload_insarmaps_metadata
 - pysar::json_mbtiles2insarmaps, 93
- upload_json
 - pysar::json_mbtiles2insarmaps, 93
- upload_mbtiles
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 240
- Usage
 - pysar::transect_legacy, 185
- usage
 - get_modis_v3, 21
 - pysar::add_attribute, 70
 - pysar::baseline_error, 72
 - pysar::baseline_trop, 73
 - pysar::coord_glob2radar, 73
 - pysar::coord_radar2glob, 74
 - pysar::correct_dem, 74
 - pysar::correlation_with_dem, 75
 - pysar::diff, 77
 - pysar::gamma_view, 80
 - pysar::ifgram_closure, 85
 - pysar::ifgram_reconstruction, 87
 - pysar::incidence_angle, 89
 - pysar::info, 90
 - pysar::insar_vs_gps, 91
 - pysar::lod, 104
 - pysar::look_angle, 105
 - pysar::los2enu, 105
 - pysar::multi_transect, 118
 - pysar::perp_baseline, 134
 - pysar::quality_map, 148
 - pysar::range_distance, 148
 - pysar::rewrap, 151
 - pysar::save_mat, 154
 - pysar::save_mat_orig, 155
 - pysar::save_roipac, 155
 - pysar::sum_epochs, 170
 - pysar::temporal_average, 171
 - pysar::temporal_coherence, 171
 - pysar::temporal_derivative, 172
- username
 - pysar::add_attribute_insarmaps::InsarResult←
Controller, 237
- valinit
 - pysar::tsviewer, 214
- Ve
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- velocity_uncertainty
 - troposphere_uncertainty, 227
- velocity_uncertainty_vs_distance
 - troposphere_uncertainty, 227
- view.py, 297
- vmax
 - pysar::plot_atmDrop, 138
- vmin
 - pysar::plot_atmDrop, 138
- Vn
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- wavelength
 - pysar::_network, 42
- Web-Viewer.md, 297
- which
 - pysar::_pysar_utilities, 56
- Width
 - pysar::multi_transect, 129
 - pysar::transect_legacy, 196
- width
 - pysar::_datetime::progress_bar, 247
 - pysar::tsviewer, 214
- workDir
 - plot_tropcor_phase_elevation, 26
- write
 - pysar::_writefile, 66
- write_complex64
 - pysar::_writefile, 66
- write_complex_int16
 - pysar::_writefile, 67
- write_dem
 - pysar::_writefile, 67
- write_float32
 - pysar::_writefile, 67
- write_gmt_simple
 - pysar::_gmt, 32
- write_grd_file
 - pysar::save_gmt, 152
- write_kmz_file
 - pysar::save_kml, 153

write_pairs_list
 pysar::_network, [42](#)
 write_real_float32
 pysar::_writefile, [67](#)
 write_real_int16
 pysar::_writefile, [67](#)
 write_roipac_rsc
 pysar::_writefile, [68](#)
 write_rsc
 pysar::prep_isce, [143](#)
 write_to_h5
 delayTimeseries, [19](#)

x
 pysar::l1, [95](#)
 pysar::multi_transect, [129](#)
 pysar::transect_legacy, [196](#)
 pysar::tsviewer, [214](#)

X0
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [197](#)

x0
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [196](#)

X1
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [197](#)

x1
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [197](#)

XX0
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [197](#)

XX1
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [197](#)

xc
 pysar::multi_transect, [130](#)
 pysar::transect_legacy, [197](#)

Xf0
 pysar::multi_transect, [130](#)

Xf1
 pysar::multi_transect, [130](#)

xlim
 pysar::multi_transect, [130](#)

y
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [197](#)
 pysar::tsviewer, [214](#)

Y0
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [197](#)

y0
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [197](#)

Y1
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [198](#)

y1
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [198](#)

YY0
 pysar::multi_transect, [132](#)
 pysar::transect_legacy, [198](#)

YY1
 pysar::multi_transect, [132](#)
 pysar::transect_legacy, [198](#)

yc
 pysar::multi_transect, [131](#)
 pysar::transect_legacy, [198](#)

Yf0
 pysar::multi_transect, [131](#)

Yf1
 pysar::multi_transect, [131](#)

ylim
 pysar::multi_transect, [132](#)
 pysar::transect_legacy, [198](#)
 pysar::tsviewer, [214](#)

yticks
 pysar::tsviewer, [215](#)

yx
 pysar::tsviewer, [215](#)

yymmdd
 pysar::_datetime, [31](#)
 pysar::_pysar_utilities, [56](#)

yymmdd2YYYYMMDD
 pysar::_pysar_utilities, [57](#)

yymmdd2yyyymmdd
 pysar::_datetime, [31](#)

yyyymmdd
 pysar::_datetime, [31](#)
 pysar::_pysar_utilities, [57](#)

yyyymmdd2years
 pysar::_datetime, [32](#)
 pysar::save_mat, [154](#)
 pysar::save_mat_orig, [155](#)

zi
 pysar::multi_transect, [132](#)
 pysar::transect_legacy, [198](#)

zwd2swd
 dloadUtil, [21](#)