## 1. Intro - Preparing the stack for analysis

**Note:** Execute all the commands in this lab session on a terminal in the Guacamole interface.

GIAnT is designed to work with outputs from multiple SAR/InSAR processors -e.g, ISCE, ROI_PAC etc. The very first stage of processing with GIAnT transforms InSAR products from their native formats (e.g, GMTSAR’s grd files, ISCE’’s binary files etc) to an internally consistent Hierarchical Data Format 5 (HDF5) format.

In this tutorial, we will describe the steps involved in transforming all the input data (described in the previous tutorial) into a HDF5 format needed by GIAnT. Again, we start with our test dataset located in the directory “synthetic”:

```
> cd /home/ubuntu/data/giant/kilaeau/GIAnT
> ls
example.xml  ifg.list  prepdataxml.py  prepsbasxml.py  userfn.py
```

From amongst the various python scripts in the directory - “userfn.py” and “prepdataxml.py” are needed for preparing our data stack for analysis. The other python scripts are related to the actual time-series analysis and will be discussed in later tutorials.

## 2. userfn.py - Translating pair information to actual files on disk

As described in the previous tutorial, “ifg.list” is a four column text file that describes our interferogram network in a simple fashion.

```
> less ifg.list
20110310   20101212   -90.8372435038   CSK
20110322   20101220   -53.7819100954   CSK
20110407   20110326   183.599267061   CSK
20101228   20101220   -259.347791029   CSK
20110322   20110318   -112.975218835   CSK
...
```

We also mentioned that we stored our unwrapped phase and coherence files in  individual sub-directories in a directory named “insar”. But we never provided the exact mapping between each line of “ifg.list” and the corresponding files in “insar”. This is accomplished through userfn.py .

```
>less userfn.py
def makefnames(dates1, dates2, sensor):
    dirname = '../insar'
    root = os.path.join(dirname, dates1+'_'+dates2)
    iname = os.path.join(root, 'filt_topophase.unw.geo')
    cname = os.path.join(root, 'topophase.cor.geo')
    return iname, cname
```

“userfn.py” should define a function named “makefnames” that takes the the master date, slave date and sensor name as inputs and returns two strings that represent the path to the unwrapped phase file and the coherence file. “userfn.py” should be located in your working directory.

This particular mechanism was devised to allow users to store InSAR outputs using their preferred directory and file name structure. Note that “userfn.py” should be considered as an user input, and each stack should be accompanied by its own “userfn.py”.

## 3. “prepdataxml.py” - setting up data characteristics.

“prepdataxml.py” is responsible for generating the input file “data.xml” which describes the characteristics of the dataset like dimensions, looks, formats etc.

```
> less prepdataxml.py
#!/usr/bin/env python

import tsinsar as ts
import argparse
import numpy as np

if __name__ == '__main__':

    ######Prepare the data.xml
    g = ts.TSXML('data')
    g.prepare_data_xml('example.xml', proc='ISCE',
                       xlim=[0,2118], ylim=[0, 1920],
                       rxlim = [1395,1405], rylim=[1420,1430],
                       latfile='', lonfile='', hgtfile='',
                       inc = 21., cohth=0.1, chgendian='False',
                       unwfmt='RMG', corfmt='RMG')
    g.writexml('data.xml')
```

We set up some basic parameters for processing our stack using “prepdataxml.py”. The complete list of all configurable parameters can be found in the GIAnT user manual. We describe the parameters that we have set up using prepdataxml.py below:

| | |
|---|---|
| example.xml | Example ISCE insarProc.xml file with dimensions and wavelength information |
| proc | Default is RPAC. We set it to ‘ISCE’ |
| xlim | X limits for cropping the image (Python convention). We use the full image here. |
| ylim | Y limits for cropping the image (Python convention). We use the full image here. |
| rxlim | X limits of reference region. Pixel 30-49 in range. (zero index) |
| rylim | Y limits of referenec region. Line 50-69 in azimuth. (zero |

| | |
|---|---|
| | index) |
| latfile, lonfile, hgtfile | Files for lat, lon, height in radar coordinates. This information is needed for atmospheric corrections, which are currently not used. These are described in the tutorial on advanced topics. |
| inc | Incidence angle (constant or file). Again only use for atmospheric corrections and GPS comparison. Not used in this tutorial. |
| cohth | Coherence threshold. All phase measurements with coherence less than this value are considered invalid. |
| chgendian | To the input files are in a different format than the native machine format. |
| unwfmt | FLT/RMG to indicate that the input is one or two channel file. |
| corfmt | FLT/RMG to indicate that the input is one or two channel file. |

The default data type for all files is float32. See GIAnT user manual for complete list of options and default values.

We will then generate our “data.xml” script as follows:

```
> python prepdataxml.py
```

To view the generated “data.xml” file,

```
> less data.xml
```

```
<data>
  <proc>
    <value>ISCE</value>
    <type>STR</type>
    <help>Processor used for generating the interferograms.</help>
  </proc>
  <master>
    <width>
      <value>2118</value>
      <type>INT</type>
      <help>WIDTH of the IFGs to be read in.</help>
    </width>
    <file_length>
      <value>1920</value>
      <type>INT</type>
      <help>FILE_LENGTH of the IFGS to be read in.</help>
    </file_length>
    <wavelength>
      <value>0.0312283810417</value>
      <type>FLOAT</type>
      <help>WAVELENGTH of the Stack. If combining sensors,ensure that
they are all converted to same units.</help>
    </wavelength>
...
```

Note that the generated XML file can be modified in a text editor, and we include a help string to describe each of the parameters in the file.

We are now ready to gather data into a HDF5 file readable by GIAnT.

## 4. PrepIgramStack.py - preparing the stack

From the GIAnT working directory, execute PrepIgramStack.py.

(NOTE: The PrepIgramStack.py command and many of the rest in this lab need to be run from the X11 windows in the Remote Desktop function of EarthKit.)

```
> cd /home/ubuntu/data/giant/kilaeau/GIAnT

> PrepIgramStack.py
<module> - INFO - No common mask defined
<module> - INFO - Output h5file: Stack/RAW-STACK.h5
<module> - INFO - PNG preview dir: Figs/Igrams
<module> - INFO - Deleting previous Stack/RAW-STACK.h5
<module> - INFO - Reading in IFGs
[========================= 59% =>                    ]    134s / 93s
```

As indicated by the screen output, the program generates a file named “Stack/RAW-STACK.h5” in the Stack directory and another directory called “Figs/Igrams”.

```
> ls
data.xml     Figs      prepdataxml.py  Stack      userfn.pyc
example.xml  ifg.list  prepsbasxml.py  userfn.py


> ls Stack
RAW-STACK.h5

> ls Figs
Igrams

> ls Figs/Igrams
I001-20110310-20101212-CSK.png  I024-20110404-20110214-CSK.png
I002-20110322-20101220-CSK.png  I025-20110302-20101228-CSK.png
I003-20110407-20110326-CSK.png  I026-20110404-20110403-CSK.png
I004-20101228-20101220-CSK.png  I027-20110129-20101228-CSK.png
...
```

HDF5 outputs of all GIAnT programs are stored in the “Stack” directory and associated PNG previews are generated in a directory named “Figs”.

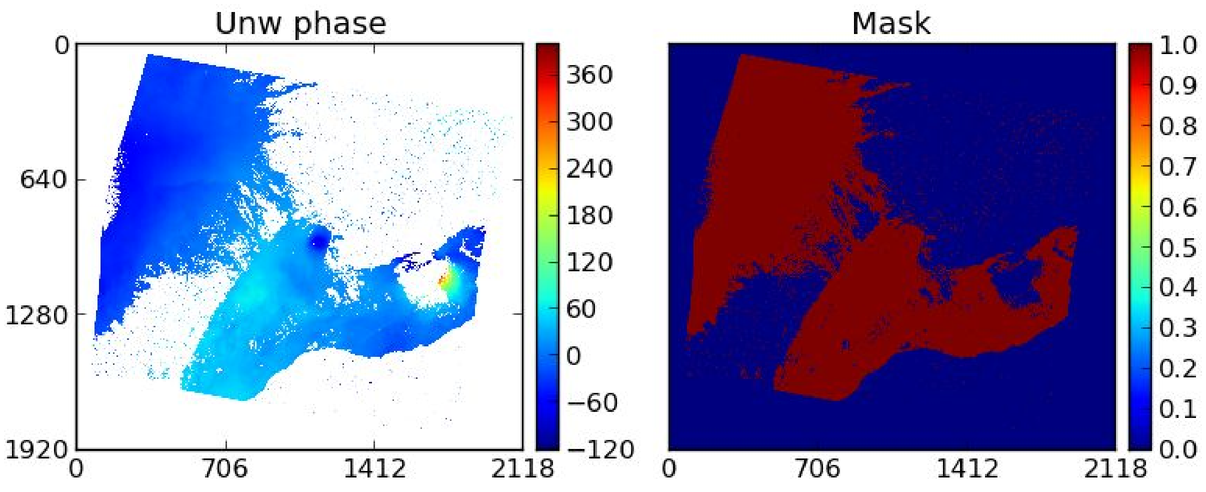
## 5. PNG previews - What does our data look like?

The directory Figs/Igrams contains PNG previews of all unwrapped interferograms listed in ifg.list . The PNG files are numbered in sequence. The PNG preview corresponding to the 80th interferogram in our test data set.

To preview the PNG files, run the following command: (NOTE: you will need to run this image preview command from the Remote Desktop as it is graphical in nature)

```
> cd Figs/Igrams
> eog *.png
```

Notice that a coherence threshold has been applied to the interferograms depending on the user inputs in data.xml. The unwrapped phase has been converted to mm at this stage.



## 6. Listing contents of RAW-STACK.h5

In this section, we will try to understand the structure of the HDF5 file `Stack/RAW-STACK.h5` created by “PrepIgramStack.py”. We can summarize the contents of this file using `h5ls`

```
> cd ../..
> h5ls Stack/RAW-STACK.h5
Jmat                     Dataset {46, 17}
bperp                    Dataset {46}
cmask                    Dataset {1920, 2118}
dates                    Dataset {17}
igram                    Dataset {46, 1920, 2118}
tims                     Dataset {17}
usat                     Dataset {17}
```

This lists the various arrays stored in the HDF5 file and their corresponding sizes.

The details on a particular variable, say Jmat, can be obtained using `h5dump`

```
> h5dump -a /Jmat/help Stack/RAW-STACK.h5
HDF5 "RAW-STACK.h5" {
ATTRIBUTE "/Jmat/help" {
   DATATYPE  H5T_STRING {
         STRSIZE 28;
         STRPAD H5T_STR_NULLPAD;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      }
   DATASPACE  SCALAR
   DATA {
   (0): "Connectivity matrix [-1,1,0]"
   }
}
}
```

Every HDF5 dataset created by GIAnT includes a self-explanatory “help” attribute which is listed in the “Data{}” section of the output from the `h5dump` command.

RAW-STACK.h5 has all the data we need to proceed to the next stage of time-series processing, stored in a convenient and easily accessible format.