1. Intro - Preparing the ISCE 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, ISCE's binary files, GMTSAR's grd 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/napa/GIAnT
> ls
example.rsc  map.json        prepsbasxml.py
ifg.list     prepdataxml.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 Lab 12.2.

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
20090218   20091117   0.0 UAVSAR
20090218   20100511   0.0 UAVSAR
20090218   20101028   0.0 UAVSAR
20091117   20100511   0.0 UAVSAR
20091117   20101028   0.0 UAVSAR
...
```

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'
    key1 = date2key(dates1)
    key2 = date2key(dates2)

    pre = key1+'__'+key2+'_hh.'
    root = os.path.join(dirname, key1+'__'+key2)
    iname = os.path.join(root, pre+'filt_topophase.unw')
    cname = os.path.join(root, pre+'phsig.cor')
    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. userfn.py and map.json

GIAnT only insists on the existence a function named “makefnames” within the script “userfn.py”. This function can in turn invoke other functions from other python libraries or look up databases to construct filenames. In this case, it invokes the “date2key” function which has also been defined in the same script.

```
> less userfn.py
...
def date2key(indate):
    fid = open('map.json', 'r')
    keymap = json.load(fid)
    fid.close()

    instr = str(indate)

    for key,value in keymap.items():
        if instr in value:
            return key

    raise Exception('Date not found')
    return
…
```

This is a simple python function that loads “map.json” and accepts a datestring in yyyymmdd format to return the corresponding user-defined ISCE key.

4. “prepdataxml.py” - setting up data properties.

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

```
> cd /home/ubuntu/data/giant/napa/GIAnT
> 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.rsc',
                       xlim=[0,1650], ylim=[0, 4165],
                       rxlim = [745,755], rylim=[3595,3605],
                       latfile='', lonfile='', hgtfile='',
                       inc = 21., cohth=0.4, chgendian='False',
                       unwfmt='RMG', corfmt='FLT')
    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.rsc	ROI_PAC style resource file with minimum common metadata.
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>RPAC</value>
    <type>STR</type>
    <help>Processor used for generating the interferograms.</help>
  </proc>
  <master>
    <width>
      <value>1650</value>
      <type>INT</type>
      <help>WIDTH of the IFGs to be read in.</help>
```

```
    </width>
    <file_length>
      <value>4165</value>
      <type>INT</type>
      <help>FILE_LENGTH of the IFGS to be read in.</help>
    </file_length>
    <wavelength>
      <value>0.238403545</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/napa/GIAnT

> PrepIgramStack.py
<module> - INFO - Number of interferograms = 30
<module> - INFO - Number of unique SAR scenes = 12
<module> - INFO - Number of connected components in network: 1
<module> - INFO - No common mask defined
<module> - INFO - Output h5file: Stack/RAW-STACK.h5
<module> - INFO - PNG preview dir: Figs/Igrams
<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

...
```

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

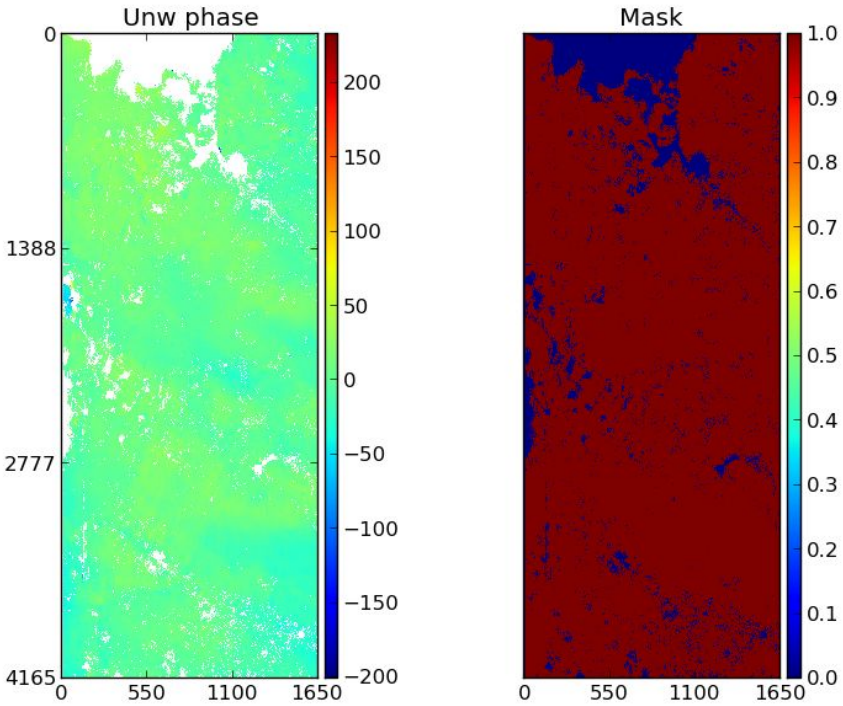
5. PNG previews - What does our data look like?

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 /home/ubuntu/data/giant/napa/GIAnT
> h5ls Stack/RAW-STACK.h5
Jmat                     Dataset {30, 12}
bperp                    Dataset {30}
cmask                    Dataset {4165, 1650}
dates                    Dataset {12}
igram                    Dataset {30, 4165, 1650}
tims                     Dataset {12}
usat                     Dataset {12}
```

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

HDF5 datasets are compatible with “gdal” and you can use gdalinfo to display help information about each dataset.

```
> gdalinfo Stack/RAW-STACK.h5
Driver: HDF5/Hierarchical Data Format Release 5
Files: RAW-STACK.h5
Size is 512, 512
Coordinate System is `'
Metadata:
  bperp_help=Array of baseline values.
  cmask_help=Common mask for pixels.
  dates_help=Ordinal values of SAR acquisition dates.
  help=All the raw data read from individual interferograms into a
single location for fast access.
  igram_help=Unwrapped IFGs read straight from files.
  Jmat_help=Connectivity matrix [-1,1,0]
  tims_help= Array of SAR acquisition times.
Subdatasets:
  SUBDATASET_1_NAME=HDF5:"RAW-STACK.h5"://Jmat
```

```
  SUBDATASET_1_DESC=[30x12] //Jmat (64-bit floating-point)
  SUBDATASET_2_NAME=HDF5:"RAW-STACK.h5"://cmask
  SUBDATASET_2_DESC=[4165x1650] //cmask (64-bit floating-point)
  SUBDATASET_3_NAME=HDF5:"RAW-STACK.h5"://igram
  SUBDATASET_3_DESC=[30x4165x1650] //igram (32-bit floating-point)
Corner Coordinates:
Upper Left  (    0.0,    0.0)
Lower Left  (    0.0,  512.0)
Upper Right (  512.0,    0.0)
Lower Right (  512.0,  512.0)
Center      (  256.0,  256.0)
```

Every HDF5 dataset created by GIAnT includes a self-explanatory “help” attribute which is listed in the “Metadata” section of the output from the `gdalinfo` 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.