1. Exploring the test data set

The primary requirement for analyzing interferogram stacks with GIAnT, is that all the input data should be coregistered on a common grid. We could design our interferogram processing workflows to generated coregistered output products in radar coordinates (i.e, along-track, slant range space) or in geo-coordinates (i.e, latitude, longitude space). For this exercise, we will consider actual UAVSAR RPI SLC stack acquired over Napa, California, and processed to a common grid in radar coordinates by the UAVSAR project. Coregistered interferograms (in radar coordinates) were generated using ISCE as outlined in Lab 11. Let us peruse the contents of the directory containing the test dataset, to try and understand the inputs needed by GIAnT.

We'll start in this directory:

```
> cd /home/ubuntu/data/giant/napa
> ls
GIAnT insar preprocessed prepGIAnT.py
```

"insar" directory contains all the interferograms processed using the approach described in Lab 11. We will proceed with our time-series analysis in "GIAnT" directory. "prepGIAnT.py" is a simple python script that was used to set up a GIAnT analysis run using outputs from ISCE. We will now describe the contents of the "insar" and "GIAnT" directories in detail.

2. “insar” - directory with unwrapped phase and coherence files

In the “insar” directory, we have compiled all the required interferograms and coherence files as float32 binary files. Information corresponding to each pair is stored in an individual subdirectory.

```
> cd insar
> ls
isceProc_20141011061450.xml  uav3__uav5  uav7__uav10
uav10__uav11                 uav3__uav6  uav7__uav8
uav10__uav12                 uav4__uav5  uav7__uav9
uav11__uav12                 uav4__uav6  uav8__uav10
uav1__uav2                   uav4__uav7  uav8__uav11
…
```

Each individual interferogram sub-directory contains the geocoded unwrapped phase and coherence files. Remember that we generated these interferogram products directly from the coregistered RPI_SLC products delivered by the UAVSAR project. Let us look at the contents of one of the interferogram directories.

```
> cd uav1__uav2
> ls
uav1__uav2_hh.filt_topophase.unw
uav1__uav2_hh.filt_topophase.unw.xml
uav1__uav2_hh.phsig.cor
uav1__uav2_hh.phsig.cor.xml
```

GIAnT needs the following conditions to be satisfied:

- All the unwrapped phase files should be of the same size. All the coherence files should be of the same size.
- The top left pixel of each file lines up exactly.
- A consistent format, i.e, data type, interleaving scheme, should be followed for unwrapped phase and coherence files.

You can view the contents of each of these files using mdx.py. NOTE: you will need to run mdx.py from the graphical desktop.

```
> cd insar/uav1__uav2
> mdx.py uav1__uav2_hh.filt_topophase.unw uav1__uav2_hh.phsig.cor
```

The metadata associated with coregistered stack of interferograms is available in the file named `isceProc_20141011061450.xml`.

3. “GIAnT” - directory for time-series analysis

We will now proceed to the GIAnT analysis directory.

```
> cd ../../GIAnT
> ls
example.rsc  map.json        prepsbasxml.py
ifg.list     prepdataxml.py  userfn.py
```

This lab will focus on describing the two files: “ifg.list” and “example.rsc” . The python scripts are used in the actual time-series analysis and will be explained in detail in Lab 12.1

Note:

The XML file format used by GIAnT is different from that used by ISCE. Maybe in the future, we will converge on a common format.

4. ifg.list - Interferogram network description

“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
20091117   20110713   0.0 UAVSAR
20100511   20101028   0.0 UAVSAR
20100511   20110713   0.0 UAVSAR
20100511   20111103   0.0 UAVSAR
20101028   20110713   0.0 UAVSAR
20101028   20111103   0.0 UAVSAR
20101028   20120418   0.0 UAVSAR
...
```

Each line of ifg.list is of the following form:
Masterdate   Slavedate   Bperp_in_m   Sensor

- Master and slave dates can be provided in yyyymmdd or yymmdd format. We used the yyyymmdd format for this example.
- Perpendicular baseline is provided in meters. This can be used to generate the time-baseline plots. The baseline values can be found in isceProc.xml files generated by ISCE. For UAVSAR RPI_SLC products this is 0.0.
- Sensor information is used if users want to build stacks that combines information from multiple sensors. This is particularly useful if your dataset includes acquisitions from multiple sensors on the same date.
- GIAnT does not insist on a particular temporal ordering of the master-slave acquisition in an interferogram. GIAnT can decode the order and build time-series design matrices accordingly. However it would be easier for the users if they stuck to one consistent order - earliest-latest or latest-earliest.

Note:
The presented scheme will not work for UAVSAR stacks that include multiple acquisitions on the same date. Tropospheric phase delay components between such acquisitions can be highly correlated and default setup in GIAnT is not set up to handle such cases. Python scripts can be easily customized for such stacks.

5. “example.rsc” - common metadata

ISCE generates a detailed progress log file (isceProc.xml) that contains information related to each of the processing steps. Our “prepGIAnT.py” script parses this file and stores the bare minimum common metadata needed to run a GIAnT analysis in a ROI_PAC style text file named “example.rsc”.

Let’s look at our example.rsc file:

```
> less example.rsc
CENTER_LINE_UTC            82112.0
WIDTH                      1650
FILE_LENGTH                4165
HEADING_DEG                0.0
WAVELENGTH                 0.238403545
```

Note that only some of these fields, that are common to the stack of interferograms, are used in the processing. In particular, the following fields are gathered:

Parameter	**Source**
Width of image	filt_topophase.unw.xml
Length of image	filt_topophase.unw.xml
Wavelength	isceProc.xml
UTC (Used for tropospheric corrections)	isceProc.xml
Heading (Used for GPS-based deramping)	isceProc.xml

We have now gathered all the information associated with the interferogram stack and can proceed to the time-series analysis labs.

Note:
GIAnT was primarily developed for analyzing space-borne InSAR data which are typically acquired from platforms in sun synchronous orbits. Hence, a single value for UTC acquisition time of the day is sufficient to search for relevant weather model data for applying tropospheric corrections. This assumption may not be valid for airborne data. However, the swath width from airborne sensors tend to be much smaller than those of spaceborne sensors and there may not be significant improvement in results when atmospheric corrections (usually derived from meteorological datasets at 0.5 degrees resolution) are applied.

6. “map.json” - mapping between dates and ISCE keys

In Lab 10, we used specific keys to refer to various available SAR acquisitions(uav1, uav2, …, uav12) for generating our stack. We need to communicate the mapping between these keys and actual SAR acquisition dates to GIAnT. We do this through a simple text file named “map.json” that was automatically generated for you by the prepGIAnT.py script.

```
> less map.json
{
    "uav1": "20090218",
    "uav10": "20131031",
    "uav11": "20140529",
    "uav12": "20140829",
    "uav2": "20091117",
    "uav3": "20100511",
    "uav4": "20101028",
    "uav5": "20110713",
    "uav6": "20111103",
    "uav7": "20120418",
    "uav8": "20121105",
    "uav9": "20130508"
}
```

Note:
This mapping would not be necessary if we only acquired one SAR acquisition per day and directly used the acquisition date in the product file names, as is common with space-borne data. For this example, we are only going to use one SAR acquisition from any given date to generate the stacks.