LOFAR Data Storage and Handling

Ger van Diepen

ASTRON

On behalf of Imaging Pipeline Team

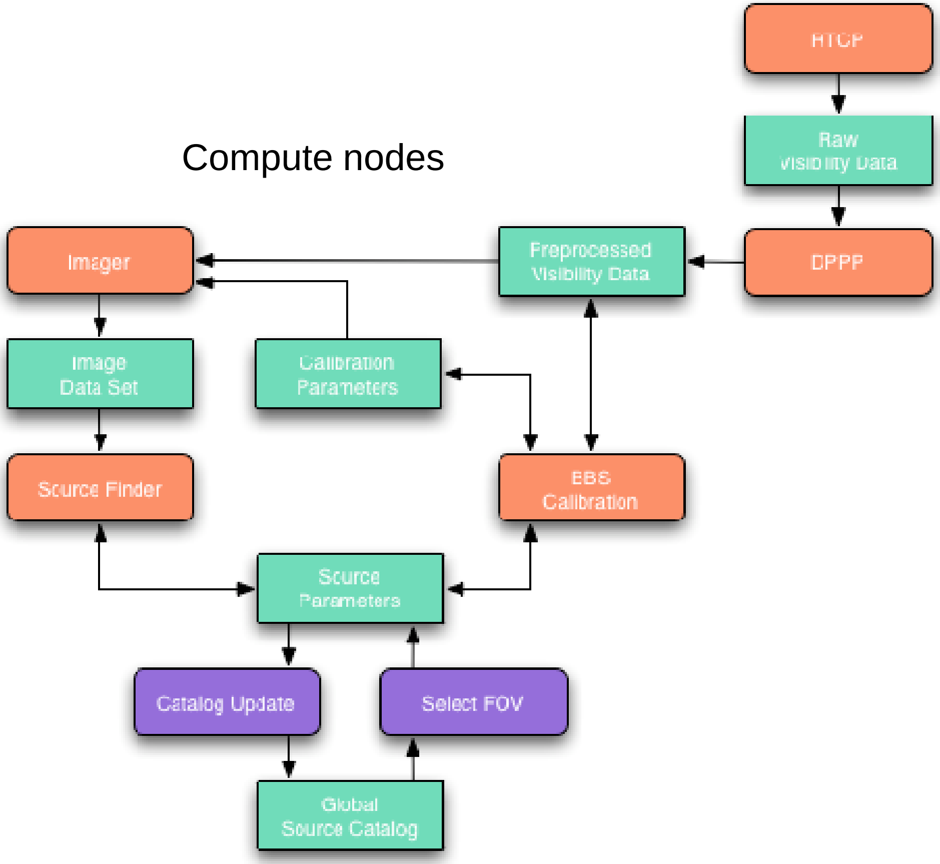
Outline

- LOFAR Imaging Pipeline data flow and data format
- Data Handling Tools

Imaging Pipeline data flow

Compute nodes

Storage nodes



Pipeline script executes all steps

Distributed Data Processing

- Data are distributed per subband (up to 248)
  - 256 channels, 64 stations --> up tp 4 GB/sec (already 24 TB observation done)
  - Meta file (.gvds file) keeps track of location of all parts
- First processing step (NDPPP):
  - auto-flagging
    - MADFlagger: median(abs(data-median)) thresholding
    - rficonsole: surface fit in time-frequency (see http://www.astro.rug.nl/rfi-software)
  - averages data (up to factor 10 in time and freq)
  - writes to compute nodes
- All further processing done on compute nodes
  - Calibration (BBS) solves for time/freq dependent parms; also DDE parms
  - Imaging (ASKAP's or CASA's imager)
  - Source finding
- Results written back to storage nodes to be archived

- All subbands are processed individually (embarrassingly parallel)
  - No real need for parallel file system
  - Map/Reduce type of processing
  - If needed (e.g. calibration over subbands) only normal equations are exchanged
  - Multi-threading where applicable

Pipeline

- Pipeline script executes all steps (single major cycle so far)
  - no archiving step yet
  - fully distributed
    - Uses .gvds file to determine where to start processes
- Job will be automatically started by M&C (almost done)
  - Observation specs should set the pipeline parameters
- Job will be automatically scheduled (work in progress)
  - uses resources like avaibable disk space and computer time
- Written in python
  - Uses ipython for communication
  - Extendible framework
    - Will be used for other pipelines

Data formats

- Casacore Table System (MeasurementSet) or HDF5
  - HDF5 is used by other LOFAR KSPs

- Casacore MeasurementSets used because:
  - Other packages (CASA, ASKAP) use MeasurementSets
    - Do not want to convert between formats
  - Dedicated storage managers are possible (from dynamically loadable library)
    - RTCP writes visibility data directly (no overhead, robust in case of crash)
    - LofarStMan maps it as a table
    - Note: ALMA/EVLA looking at mapping SDM format to a storage manager
  - Flat data space, thus no predefined ordering (i.e. hierarchy)
    - Easier to form arbitrary subsets
  - HDF5 is slow when accessing small hyperslabs
  - Versatile query language (TaQL) to select or modify data

- Special LOFAR subtables and columns added
  - E.g. Dipole positions per station

- Images can be written in HDF5 or FITS format

Data Handling Toolkit

casacore, pyrap, and CASA form Jan Noordam’s 3rd pillar

Data Handling

- casacore
  - C++ library
  - Arrays, Math, Tables, MeasurementSets, Measures, Images
- pyrap
  - Python interface to casacore
  - pyrap.quanta, measures, tables, images
  - uses numpy arrays
  - E.g. iterate over cross-correlation baselines in a MeasurementSet

```
t = table(‘my.ms’)
t1 = t.query(‘ANTENNA1 != ANTENNA2’)
for subset in t1.iter([“ANTENNA1”, “ANTENNA2”]):
     visdata = subset.getcol (‘DATA’)
```

- TaQL
  - SQL-like
  - Select, modify, or insert table data
  - E.g. subtract background noise using median in a 51x51 box around each pixel

```
update my.img set map = map - runningmedian(map, 25, 25)
```

Toolkit

- CASA tools
  - casabrowser   view and edit a table
  - tableplot   plot one column or expression against another
  - plotms
  - casaviewer (MS and image)

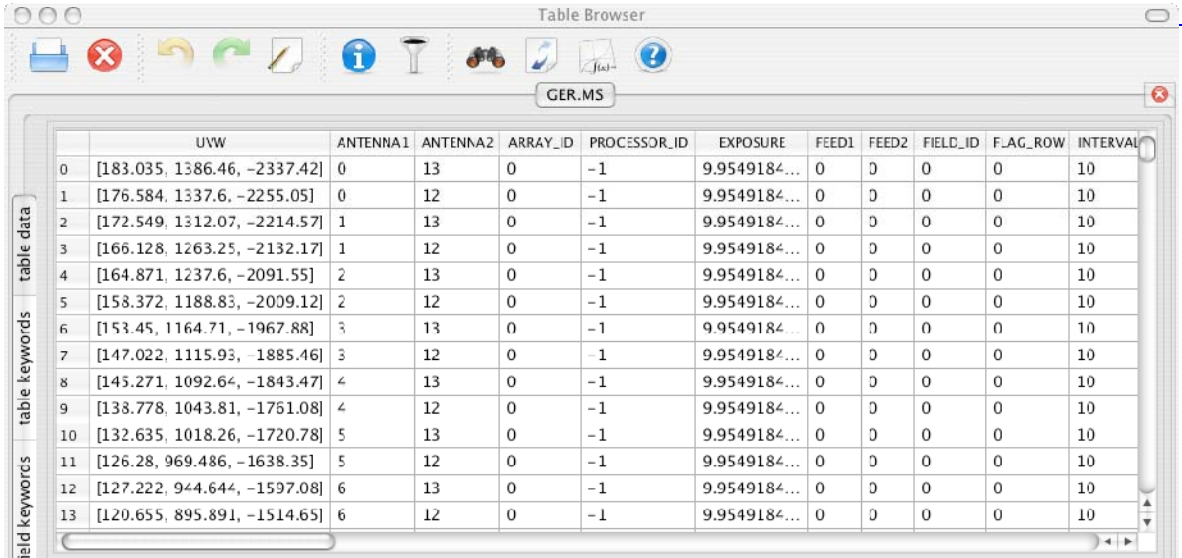
- Combining and selecting MeasurementSets

  e.g. select band 0 from an observation split in time (say MSSS)

```
t = table([‘ms1’,’ms2’,’ms3’])
t1 = t.query (‘DATA_DESC_ID==0’)
t1.copy (‘sel.ms’, deep=True)
```

- Averaging in time/freq
  - LOFAR’s NDPPP
  - Tools in CASA

casabrowser



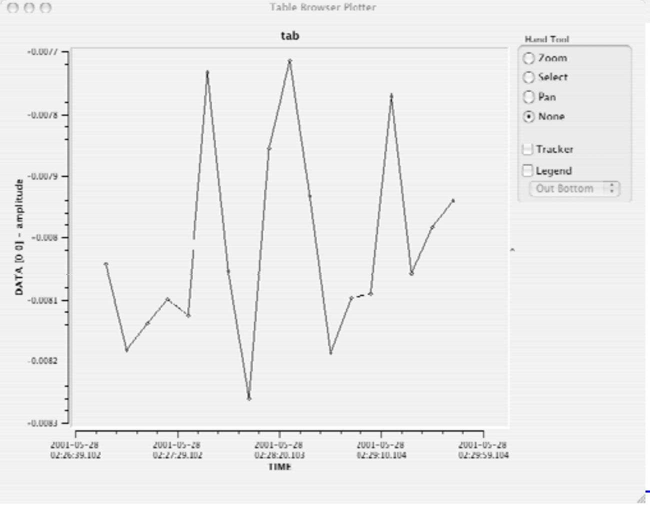
	UVW	ANTENNA1	ANTENNA2	ARRAY_ID	PROCESSOR_ID	EXPOSURE	FEED1	FEED2	FIELD_ID	FLAG_ROW	INTERVAL
0	[183.035, 1386.46, -2337.42]	0	13	0	-1	9.9549184...	0	0	0	0	10
1	[176.584, 1337.6, -2255.05]	0	12	0	-1	9.9549184...	0	0	0	0	10
2	[172.549, 1312.07, -2214.57]	1	13	0	-1	9.9549184...	0	0	0	0	10
3	[166.128, 1263.25, -2132.17]	1	12	0	-1	9.9549184...	0	0	0	0	10
4	[164.871, 1237.6, -2091.55]	2	13	0	-1	9.9549184...	0	0	0	0	10
5	[158.372, 1188.83, -2009.12]	2	12	0	-1	9.9549184...	0	0	0	0	10
6	[153.45, 1164.71, -1967.88]	3	13	0	-1	9.9549184	0	0	0	0	10
7	[147.022, 1115.93, -1885.46]	3	12	0	-1	9.9549184...	0	0	0	0	10
8	[145.271, 1092.64, -1843.47]	4	13	0	-1	9.9549184...	0	0	0	0	10
9	[138.778, 1043.81, -1761.08]	4	12	0	-1	9.9549184...	0	0	0	0	10
10	[132.635, 1018.26, -1720.78]	5	13	0	-1	9.9549184...	0	0	0	0	10
11	[126.28, 969.486, -1638.35]	5	12	0	-1	9.9549184...	0	0	0	0	10
12	[127.222, 944.644, -1597.08]	6	13	0	-1	9.9549184...	0	0	0	0	10
13	[120.655, 895.891, -1514.65]	6	12	0	-1	9.9549184...	0	0	0	0	10

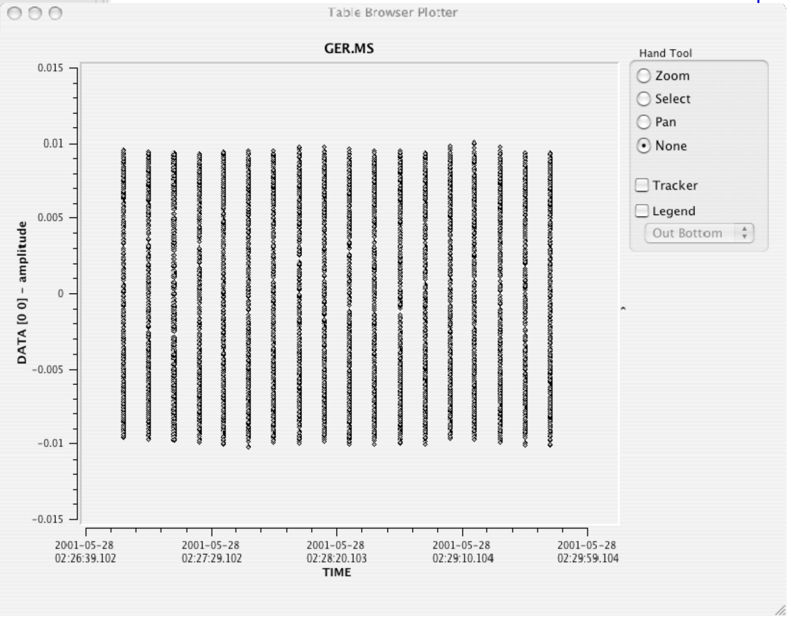
Restore Columns | Resize Headers

PAGE NAVIGATION First << [1 / 13] >> Last 1 Go Loading 1000

```
python
    from pyrap.tables import *
    t = table('GER.MS')
    t1 = t.query('DATA DESC ID=0')
    t1.browse()
```

```
casabrowser ('GER.MS')
```

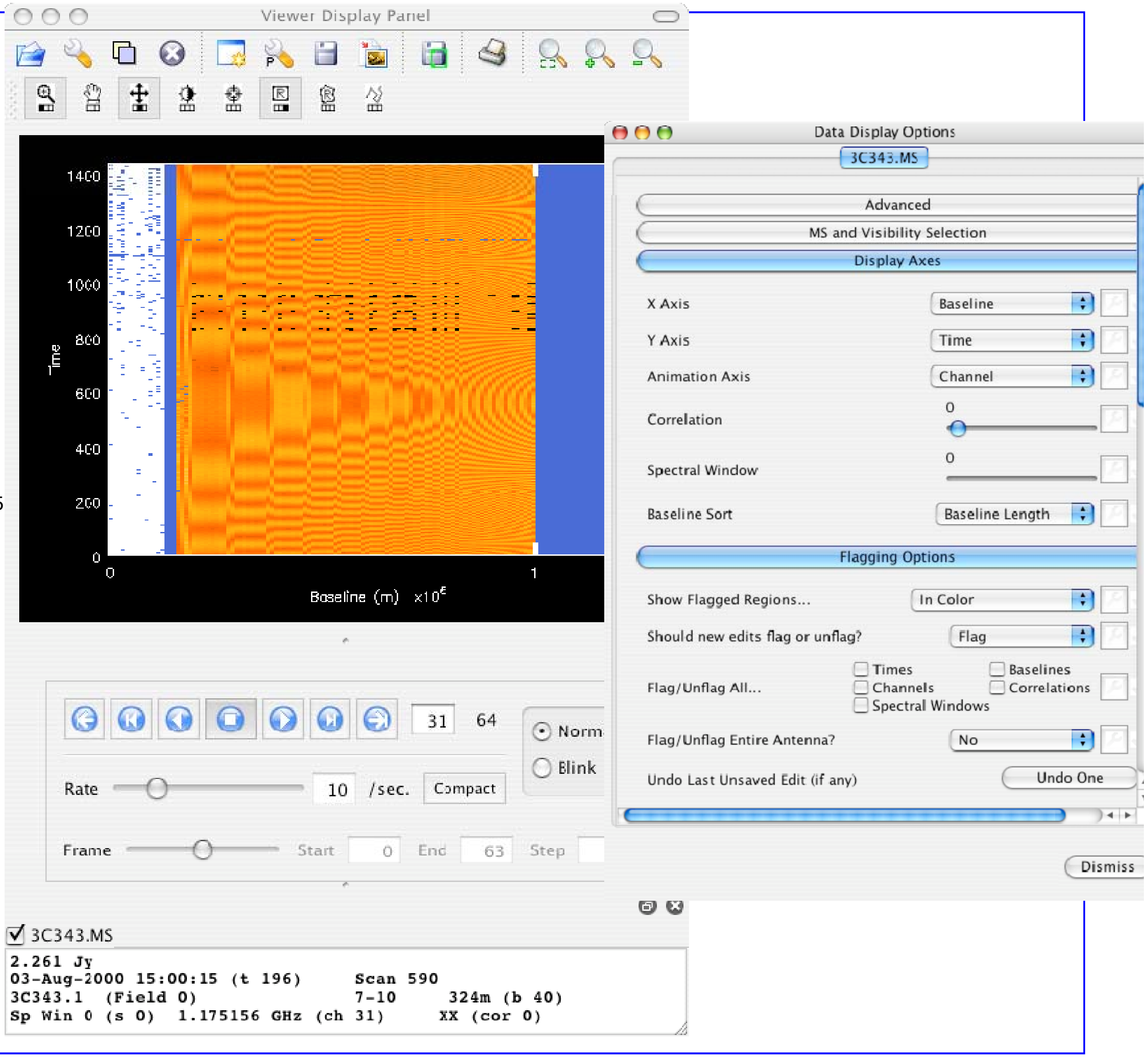




casaviewer (MS)

```
python
    from pyrap.tables import *
    t = table('GER.MS')
    t1 = t.query('DATA_DESC_ID=0')
    t1.view()



casaviewer ('GER.MS')
```

```
Sorting... Done.

/Users/diepen/3C343.MS
  Selected MS:  Time slots: 1437  Baselines (incl. gaps): 15
  Correlations: 4  Channels: 64  Spectral Windows: 1

Loading MS vis. data:  28%  47%  80%  89%  Done.


Resorting MS vis. data:  38%  95%  Done.
```



X Axis: Baseline
Y Axis: Time
Animation Axis: Channel
Correlation: 0
Spectral Window: 0
Baseline Sort: Baseline Length
Show Flagged Regions... In Color
Should new edits flag or unflag? Flag
Flag/Unflag All... □ Times □ Baselines □ Channels □ Correlations □ Spectral Windows
Flag/Unflag Entire Antenna? No
Undo Last Unsaved Edit (if any)

☑ 3C343.MS

```
2.261 Jy
03-Aug-2000 15:00:15 (t 196)      Scan 590
3C343.1  (Field 0)                7-10      324m (b 40)
Sp Win 0 (s 0)  1.175156 GHz (ch 31)     XX (cor 0)
```

casacore

C++ library for astronomical data handling

- Arrays templated N-dim arrays (STL-conforming)
- Tables storage mechanism with TaQL query language
- MeasurementSet visibility data storage and access (using Tables)
- Measures values in frame (direction, position, epoch, ...)
- Coordinates world coordinates for images
- Images N-dimensional image cubes with 0 or more masks
  □(also supports HDF5, FITS, Miriad, expressions (LEL))

- Used by LOFAR, ASKAP, ALMA, eVLA, MeqTrees, pyrap, pydal

- See

Download	casacore.googlecode.com
Classes	www.astron.nl/casacore/trunk/casacore/doc/html
TaQL	www.astron.nl/casacore/trunk/casacore/doc/notes/199.html
LEL	www.astron.nl/casacore/trunk/casacore/doc/notes/223.html
MS definition	www.astron.nl/casacore/trunk/casacore/doc/notes/229.html

Tables

- Collection of rows and columns in a flat data space (no hierarchy)
  - scalars and N-dim arrays of basic data types (incl. complex and string)
  - keywords (headers) to define subtables and units/reference frames

- Several table types:
  - Plain table
    contains the data columns and rows
  - Reference table
    view of another table (result of selection, sort, or iteration)
    only contains references to rows, columns, and subtables
  - Concat table
    virtual concatenation of similar tables
    subtables can be concatenated at will

- Persistent or transient
  persistent as a directory containing various files and optional subtables
- Concurrent access (one writer, multiple readers)

Table storage

- Various data (storage) managers
  - StandardStMan
    columnar storage; for scalars or small fixed arrays (like UVW)
  - IncrStMan
    rather constant data (gets compressed)
  - TiledStMan
    bulk data (tiling gives fast access for all axes)
  - Dynamically loadable dedicated data managers (e.g. LofarStMan)
    - can map an existing format to a table (if randomly accessible)
    - plans to implement mapping of the SDM format (ALMA/EVLA)
  - Virtual columns (calculated on the fly); e.g. PA, HA

MeasurementSets

Collection of Tables containing visibility data

- Described in note 229 (www.astron.nl/casacore/trunk/casacore/doc/notes/229.html)
- Predefined structure of subtables and columns
- Instrument-specific subtables and columns can be added

- Main table has data columns DATA and FLAG
  - Per row 2-dim array with axes frequency and polarisation
  - Indexed by columns containing baseline (antenna1 and antenna2), time, band, subarray, etc.

- Subtables define meta data

  ANTENNA, ARRAY, FEED, FIELD, SPECTRAL_WINDOW, ...

  Some subtables are optional

- C++ headers in ms/MeasurementSets
- No specific Python code; use pyrap.tables

Images

- N-dim cube (ra, dec, freq, pol, ...)
  with world coordinates

- Two native formats
  - Tables, HDF5
  - Tiled storage
  - data type float, double, complex, or dcomplex
  - zero or more masks
- Two external formats (readonly)
  - FITS and Miriad
  - Non-tiled storage (FITS proposal for tiling)
  - data type float only

- Virtual concatenation of images (e.g. to combine subbands)
- Image expressions using LEL (e.g. difference of images)

  See www.astron.nl/casacore/trunk/casacore/doc/notes/223.html

TaQL

Table Query Language www.astron.nl/casacore/trunk/casacore/doc/notes/199.html

- SQL-like with support for arrays and units
- Subqueries
- SELECT, UPDATE, INSERT, DELETE, CREATE TABLE, CALC
- No join, GROUPBY/HAVING
- Uses Python style (0-based indexing, C array order, end exclusive)

```
Select from my.ms orderby unique TIME
Select unique TIME from my.ms
    get unique times

Select from my.ms where ntrue(FLAG) == 0
Select from my.ms where not any(FLAG)        # faster way
    find rows without flagged data

Update my.ms set FLAG[,0]=FLAG[,0]||amplitude(DATA[,0]) > 3*median(amplitude(DATA[,0]))
    flag XX data based on a simple median filter (per row)

Update my.img set map = map - runningmedian(map, 25, 25)
    subtract background noise from image using median in a 51x51 box around each pixel

Insert into my.ms/HISTORY (TIME,MESSAGE) values (mjd(), “historystring”)
    add a line to the HISTORY table of a MeasurementSet (converts automatically to sec)

Calc date()-”1Jan2000”
    how many days since 1-1-2000
```

TaQL (cont’d)

Can be used from:

- C++

```
#include <tables/Tables/TableParse.h>
Table tab = tableCommand (‘command’);
```

- Python

```
import pyrap.tables as pt
t1 = pt.taql (‘select col1, col2 from sometable where col1=1
                      orderby col1,col2 giving outname’)
```

or

```
t = pt.table(‘sometable’)
t1 = t.query (‘col1=1’, sortlist=‘col1, col2’, columns=‘col1, col2’,
            name=‘outname’)
```

- shell

```
taql ‘select col1, col2 from sometable where col1=1 ...’
```

In C++ also

```
#include <tables/Tables/ExprNode.h>
Table t(‘sometable’)
Table t1 = t(t.col(‘col1’) == 1);
```

pyrap

python interface to casacore

- Manipulate tables, images from python
- data as numpy arrays in row major order (reverses axes!)

    pyfits does the same

pyrap.tables

create, read, write, selection, sort, iteration, display

pyrap.images

LEL, read, write, tofits, statistics, regrid, display

pyrap.quanta and pyrap.measures

measures and frames conversions

pyrap.functionals and pyrap.fitting

fitting of data to function parameters (1-dim or more)

See

Download	pyrap.googlecode.com
Modules	www.astron.nl/casacore/trunk/pyrap/docs

pyrap.tables

```
import numpy
import pylab
import pyrap.tables as pt

def plotbl (ms, band=0, ch=0, sep=5.0, fig=None):
    t  = pt.table(ms);
    t1 = t.query ('DATA_DESC_ID=%d' % band)
    pylab.figure(fig)
    pylab.clf()
    offset = 0.0
    for t2 in t1.iter(["ANTENNA1","ANTENNA2"]):
        # Get XX data of given channel
        ampl = numpy.absolute (t2.getcolslice("DATA", [ch,0], [ch,0]))
        sc = sep/numpy.mean(ampl)
        ampl = ampl.reshape(ampl.shape[0])
        pylab.plot(sc*ampl + offset)
        offset += sep
    pylab.show()
```

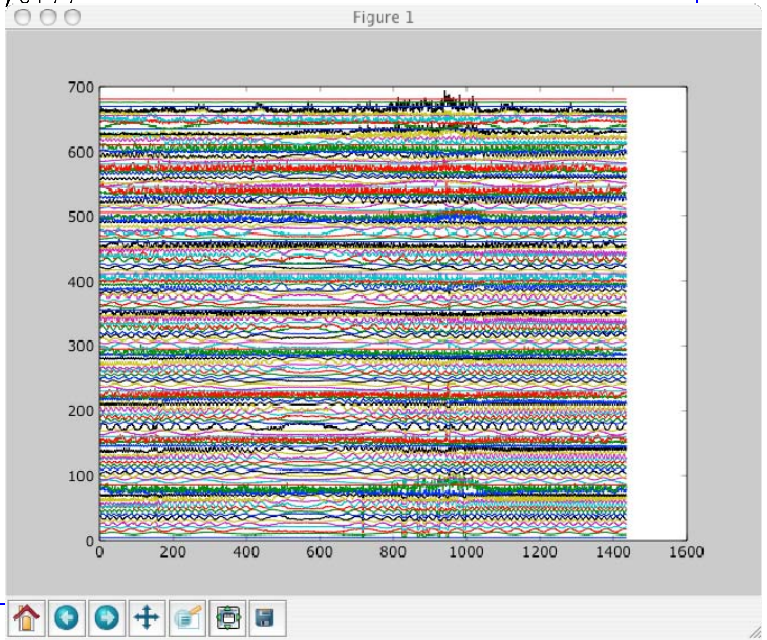


table manipulation

```
# select a spectral band from an MS

# in python
from pyrap.tables import *
t=table('3C343.MS')
t1 = table.query(‘DATA_DESC_ID=0’, name=‘3C343_SPW0.MS’)    # results in ref table
t1.copy (‘3C343_SPW0.MS’, deep=True)                        # results in plain table

Or

# in shell
taql ‘select from 3C343.MS where DATA_DESC_ID=0 giving 3C343_SPW0.MS as plain’


# Concatenate MSSS observations of same field
t=table([‘MSSS_p0.MS’,’MSSS_p1.MS’,’MSSS_p2.MS’], concatsubimage=‘SYSCAL’)
t.copy (‘MSSS.MS’)
```

pyrap.images

Uses numpy masked arrays

- for image mask=True means bad pixel; numpy opposite
- hence, getmask and putmask negate mask automatically

```
    import pyrap.images as pim
    im = pim.image(‘my.img’)
    print im.statistics()
    npmaskedarray = im.get()
    nparray = im.getdata()
    im.tofits (‘fitsfile’)
    im.view()

# image expression
    im = pim.image(“(image1 + image2 + image3) / 3”)
    im.saveas (‘avgimage’)

# image concatenation
    im = pim.image([‘image1’, ‘image2’, ‘image3’])
    im.saveas (‘concimage’)
```

CASA

- ALMA and EVLA data processing package
  - Based on casacore
- Has some useful tools:
  - casabrowser

    view data in tabular way

    editing, plotting, querying
  - casaviewer

    viewing images (also FITS, Miriad, HDF5, expressions)

    viewing and flagging MeasurementSets
  - plotms, tableplot
- Can be used from command line:

```
casaviewer ('my.img')
```

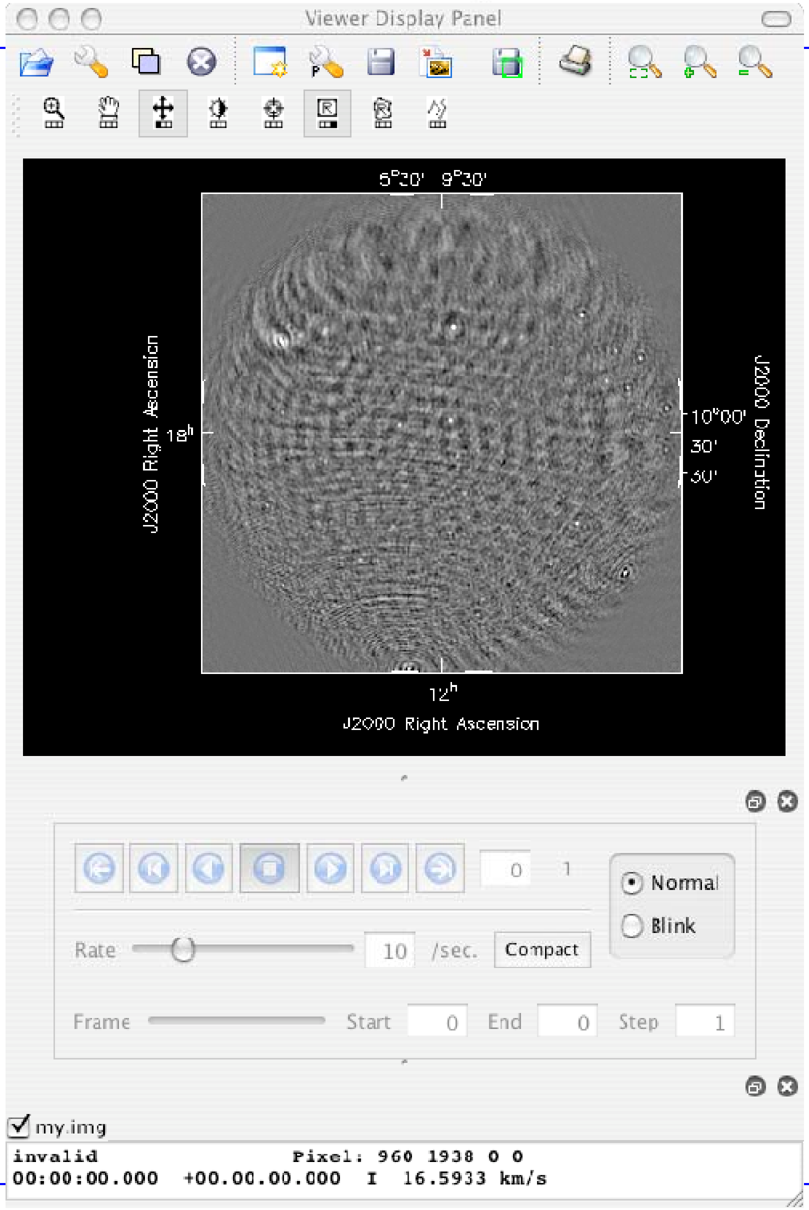
- Can be used from python (pyrap):

```
im = image('my.img')
im.view()
```

casaviewer (image)

```
python
    from pyrap.images import *
    im = image(‘my.img’)
    im.view()
```

```
casaviewer (‘my.img’)
```



```
invalid              Pixel: 960 1938 0 0
00:00:00.000  +00.00.00.000  I  16.5933 km/s
```

Thank you