ALMA data reimaging

Felix Stoehr
ALMA Science Archive

1. Introduction

idea

- Goal: create the complete set of PL Imaging products for all previous cycles

- Idea:
  - While redoing the calibration for all ES cycles seems impossible due to the very large amount of manual intervention required (redoing QA2)
  - Imaging, starting from the calibrated MS could actually work
  - Check the feasibility and cost

why

- It is clear that especially in interferometry it is not possible to create products that are suitable for all science cases (rebinning, tapering, change of resolution, weighting schemes, ...)
- In particular we believe that even if the ALMA archive had complete and science-grade data-products **still** a large fraction of ysers would want to reimage

why

- Why we nevertheless believe a full set of Images for ALMA is indispensable
  - Previews/Quick-look images can speed up the discovery by many orders of magnitude
  - We can measure fiducial rms and dynamic range from the images
  - The products can be sent to CARTA
  - We can do a generic post-analysis with ADMIT
  - We can offer the products to the VO
  - We can compute the HiPS and offer that in the AQ
  - If the products are too large for the users to create, we can create them for the users and provide cut-outs
  - The images can serve as starting point for small dedicated use reimaging
  - We can provide a homogeneous set of all ALMA images

how

1) Extract the data+scripts from the Archive
2) Discover the CASA version to use for the MS restore
3) run scriptForPI.py to get the calibrated MS
4) Apply known fixes to the data
5) Identify the MS(s) and run the ALMA Imaging PL **without changing any of the default parameters**
6) Run the Preview code
7) Run footprint extraction
8) Run ADMIT
9) Log the result of the run (errors, #products, ...) in a database

what this is *not*

- Evaluation of the scientific quality of the Imaging PL
- Creation of the best possible products
- Creation of multiple products for the different science cases

2. status

status

- Timeline
  - 2012-2013 First code to do PL calibration vs. manual calibration checking. Both were then imaged with the same parameters.
  - 2014-2016 Code to run the imaging PL on the restored MS
  - 2016-2017 Rewrote the code completely

status

- Code
  - Python
  - OO
  - Supports remote fetching of data throug the RH
  - Supports ORACLE and sqlite3 databases
  - Works around most known issues (README files, file names, …)
  - Cycle 0 as well as Cycle 1+
  - Runs with curreny bleeding edge Imaging PL
  - Uses hifatargets.py
  - Makes Previews, footprint, runs ADMIT
  - Can use parallel CASA

hifatargets.py

```
def hifatargets (vislist, importonly=False, pipelinemode='automatic', interactive=True):
...
try:
    h_init()                                                                    # Initialize the pipeline
    hifa_importdata (vis=vislist, dbservice=False, pipelinemode=pipelinemode)   # Load the data
    if importonly:
        raise Exception(IMPORT_ONLY)
    hif_mstransform (pipelinemode=pipelinemode)                                 # Split out the target data
    hifa_flagtargets (pipelinemode=pipelinemode)                                # Flag the target data
    hif_makeimlist (specmode='mfs', pipelinemode=pipelinemode)                  # Make a list of mfs targets
    hif_findcont(pipelinemode=pipelinemode)                                     # Find continuum frequency
    hif_uvcontfit(pipelinemode=pipelinemode)                                    # Fit the continuum
    hif_uvcontsub(pipelinemode=pipelinemode)                                    # Subtract the continuum fit
    hif_makeimages (pipelinemode=pipelinemode)                                  # Make clean cont images
    hif_makeimlist (specmode='cont', pipelinemode=pipelinemode)                 # Make a list of cont targets
    hif_makeimages (pipelinemode=pipelinemode)                                  # Make clean mfs images
    hif_makeimlist (width='', pipelinemode=pipelinemode)                        # Make a list ofcontsub
targets
    hif_makeimages (subcontms=False, pipelinemode=pipelinemode)                 # Make clean cont subtracted
    hif_exportdata(pipelinemode=pipelinemode)                                   # ARI: also export the
products
```

runs

- From 2015-2017
  - 921 runs (with various versions of the PL)
  - 470 runs went completely through to ADMIT

- Typical errors
  - 75 memory error
  - 51 problems with the CASA versiof for restore
  - 37 no visibility found
  - 21 core dumps
  - ...

products

- weblog
- FITS files
- Footprints: 1 per FTIS file
- Previews: 1 per FITS file
- ADMIT: 1 per 3D cube

products

casa_commands.log
casa_piperestorescript.py
casa_pipescript.py
cont.dat
flux.csv
oussid.NGC_1614_sci.spw0.cube.I.pb.fits.gz
oussid.NGC_1614_sci.spw0.cube.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw0.mfs.I.pb.fits.gz
oussid.NGC_1614_sci.spw0.mfs.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw0_1_2_3.cont.I.alpha.error.fits.gz
oussid.NGC_1614_sci.spw0_1_2_3.cont.I.alpha.fits.gz
oussid.NGC_1614_sci.spw0_1_2_3.cont.I.pb.tt0.fits.gz
oussid.NGC_1614_sci.spw0_1_2_3.cont.I.tt0.pbcor.fits.gz
oussid.NGC_1614_sci.spw0_1_2_3.cont.I.tt1.pbcor.fits.gz
oussid.NGC_1614_sci.spw1.cube.I.pb.fits.gz
oussid.NGC_1614_sci.spw1.cube.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw1.mfs.I.pb.fits.gz
oussid.NGC_1614_sci.spw1.mfs.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw2.cube.I.pb.fits.gz
oussid.NGC_1614_sci.spw2.cube.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw2.mfs.I.pb.fits.gz
oussid.NGC_1614_sci.spw2.mfs.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw3.cube.I.pb.fits.gz
oussid.NGC_1614_sci.spw3.cube.I.pbcor.fits.gz
oussid.NGC_1614_sci.spw3.mfs.I.pb.fits.gz
oussid.NGC_1614_sci.spw3.mfs.I.pbcor.fits.gz
pipeline_aquareport.xml
uid___A002_X8440e0_X4b44.ms.split.cal.calapply.txt
uid___A002_X8440e0_X4b44.ms.split.cal.flagversions.tgz
unknown.pipeline_manifest.xml
weblog.tgz

3. next steps

next steps

- Next steps
  - Build categories out of the holdings where we believe that the same results (success/failure & quality) will be reached
  - Run and verify e.g. 10 datasets of each category and validate/invalidate the approach
  - Look at the error-classes of the technical failures and see if they can be worked around.
  - Check fr fixes that need to be potentially applied (e.g. CSV2555)

Summary

- A basis of runs as well as much of the code are in place
- Now comes the hard work