ALMA Data Reduction with CASA

www.alma-allegro.nl/alma-data-reduction-casa-training-day-march-3-2017/

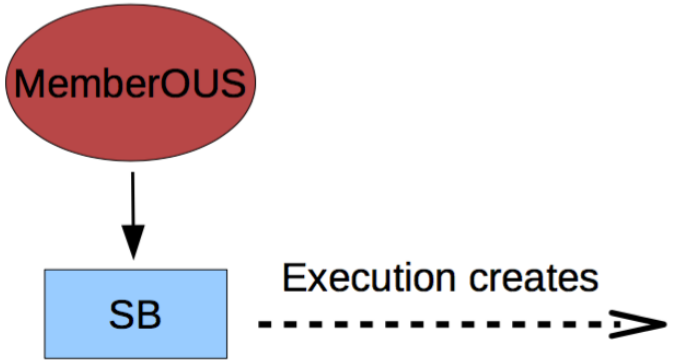
The script 'scriptForPI.py'

M.C. Toribio
Allegro

Leiden Observatory – 03/03/2017

Science Goal, SB, EB, OUS…?

PI defines **Science Goals** in the Observing Tool (OT)



Execution Block (EB):

- actual execution of an SB
- May need several EB to reach a science goal.

Scheduling Blocks (SBs) in **Observation Unit Sets (OUSs)**

- SB is a prototype of an atomic (c.a. 0.5h) observation to reach a science goal

Credit: D. Petry (ESO)

ALMA QA

QA consists on 3 (+1) steps:

- **QA0** At the time of data acquisition: *Atmosphere, Antennas, Front-ends, Connectivity, Back-ends…*
- **QA1** Monitor slowly varying array performance parameters: *arrays, antennas, calibration sources*

If OUS completed

- **QA2** Confirm that Science Goal was met;
  request additional data and iterate if not
  (implies full calibration + generation of standard science products)

If QA2 PASS

MemberOUS data is delivered to the PI

If problem reported by PI

- **QA3** re-reduction of the data, possibly replacing products in the archive

Credit: D. Petry (ESO)

ALMA Data Processing

TWO PROCESSING MODELS COEXIST:

1. **Semi-automatic calibration and imaging:**
   - The analyst edits the output of a script generator and processes data.

2. **Data calibration with automated pipeline + script generator for imaging:**
   - Calibration is performed by the pipeline and analysts take care of imaging.

FUTURE:

(1) (2) & Fully automated pipeline including imaging

ALMA Archive

http://almascience.eso.org/aq/

ALMA Science Archive Query

Query Form | Results Table

Search | Reset | Query Help

Position
- Source name (Resolver)
- Source name (ALMA)
- RA Dec
- Galactic
- Angular resolution
- Largest angular scale
- Field of view

Energy
- Frequency
- Bandwidth
- Spectral resolution
- Band

Time
- Observation date
- Integration time

Polarisation
- Polarisation type

Observation
- Line sensitivity (10 km/s)
- Continuum Sensitivity
- Water vapour

Project
- Project code
- Project title
- PI name
- Proposal authors
- Project abstract
- Publication count
- Science keyword

Publication
- Bibcode
- Title
- First author
- Authors
- Abstract
- Year

Options

View:
- (•) raw data
- ( ) project
- ( ) publication

[ ] public data only

[x] science observations only

More details at the Science Portal Documentation:
https://almascience.nrao.edu/alma-data/archive

The scripts in the PI data package

Example of directory structure after unpack:

```
2012.1.01234.S/science_goal.uid___A001_X12345_X123/
group.uid___A002_X6789ab_X6789member.uid___A002_Xcdef1_X234/
```

With subdirectories:

```
calibrated calibration log product qa raw script README
```

More details at the Science Portal Documentation:
https://almascience.nrao.edu/alma-data/archive

Navigate your folder

```
$ cd 20XX.1.0XXXX.S/sg_ouss_id/group_ouss_id/member_ouss_id/

$ ls
calibrated  calibration  log  product  qa  raw  README.txt script
```

The scripts in the PI data package

```
project_id/
└→ sg_ouss_id/
    └→ group_ouss_id/
        └→ member_ouss_id/
            ├→ README.txt
            ├→ product/
            ├→ calibration/
            ├→ qa/
            ├→ script/
            ├→ log/
            └→ raw/
```

README.txt — **READ THIS FIRST**

product/ — the FITS cubes of all images

calibration/ — calibration tables

qa/ — diagnostic summary and plots

script/ — calibration and imaging scripts

log/ — calibration and imaging log files

raw/ — created when ASDMs are unpacked

More details at the Science Portal Documentation:
https://almascience.eso.org/documents-and-tools/cycle3/ALMAQA2Products3.0.pdf

Credit: D. Petry (ESO)

The scripts in the PI data package

`project_id/`	
`└→ sg_ouss_id/`	
`└→ group_ouss_id/`	
`└→ member_ouss_id/`	
`├→ README.txt`	**READ THIS FIRST**
`├→ product/`	the FITS cubes of all images
`├→ calibration/`	calibration tables
`├→ qa/`	diagnostic summary and plots
`├→ script/`	calibration and imaging scripts
`├→ log/`	calibration and imaging log files
`└→ raw/`	created when ASDMs are unpacked
`calibrated/`	**created when `scriptForPI.py` is run**

More details at the Science Portal Documentation:
https://almascience.eso.org/documents-and-tools/cycle3/ALMAQA2Products3.0.pdf

Credit: D. Petry (ESO)

The scripts in `script` folder

Filename	Origin	Purpose
uid*.ms.scriptForCalibration.py (optional)	script-generator/ analyst	calibrates a single EB (ASDM); results in one uid*.ms.split.cal
PPR*.xml (optional)	ALMA Pipeline	controlled the run of the ALMA Pipeline; contains the list of ASDMs
casa_piperestorescript.py (optional)	ALMA Pipeline	calibrates all pipeline-processed EBs; results in one uid*.ms.split.cal per EB
casa_pipescript.py (optional)	ALMA Pipeline	enables user to rerun the Pipeline from scratch results in one uid*.ms.split.cal per EB
scriptForFluxCalibration.py (optional)	script-generator/ analyst	adjust the flux calibration of several EBs close in time which use same phase calibrator; prepare imaging; results is calibrated.ms
scriptForImaging.py	script-generator/ analyst	create all imaging products for the MOUS; results in (among others) *.fits files for all images
scriptForPI.py	added in packaging	Perform all necessary steps to create all uid*.ms.split.cal MSs

Credit: D. Petry (ESO)

scriptForImagingPrep.py

BEFORE IMAGING, USUALLY FOR:

- flux equalization
- concatenation of EBs
- splitting out science target

(In the past called: `scriptForFluxCalibration.py`)

scriptForPI.py

```
# ALMA Data Reduction Script
# $Id: scriptForPI.py,v 1.18 2016/10/18 17:33:15 dpetry Exp $

# Calibration application

import os
import sys
import glob

applyonly = True

os.environ["LANG"] = "C"

savingslevel=0
if globals().has_key("SPACESAVING"):
    print 'SPACESAVING =', SPACESAVING
    if (type(SPACESAVING)!=int or SPACESAVING<0):
        sys.exit('ERROR: SPACESAVING value \"'+str(SPACESAVING)+'\" not permitted, must be int>0.\n'
                 + 'Valid values: 0 = no saving,\n'
                 + '              1 = delete *.ms.split,\n'
                 + '              2 = delete *.ms and *.ms.split,\n'
                 + '            >=3 = delete *.ms, *.ms.split, and if possible *.ms.split.cal')

    savingslevel = SPACESAVING

if (os.path.basename(os.getcwd()) != 'script'):
    sys.exit('ERROR: Please start this script in directory \"script\".')

scriptnames = glob.glob('uid*.ms.scriptForCalibration.py')

pscriptnames = glob.glob('casa_piperestorescript.py')

p2scriptnames = glob.glob('*casa_pipescript.py')
pipererun = False
istppipe = False

pprnames = glob.glob('PPR*')

if ((len(scriptnames) + len(pscriptnames))  == 0):
    if len(p2scriptnames)>0:
        print 'Pipeline calibration by pipeline rerun'
        pipererun = True
        if os.path.exists('../calibration/jyperk.csv'):
            istppipe = True
    else:
        sys.exit('ERROR: No calibration script found.')

pprasdms = []
if (len(pprnames)>0):
    for line in open(pprnames[0]):
        if "<AsdmDiskName>" in line:
            pprasdms.append(line[line.index('uid'):line.index('</')])

try:
    os.chdir('../raw')
except:
    sys.exit('ERROR: directory \"raw\" not present.\n'
             '       Please download your raw data and unpack it to create and fill directory \"raw\".')

# check available disk space
tmppipe = os.popen("df -P -m $PWD | awk '/[0-9]%/{print $(NF-2)}'")
avspace = int((tmppipe.readline()).rstrip('\n'))
tmppipe.close()
tmppipe = os.popen("du -smc ../* | grep total | tail -n 1 | cut -f1")
packspace = int((tmppipe.readline()).rstrip('\n'))
tmppipe.close()

spacefactor = 0.

fcalpresent = False
if os.path.exists('../script/scriptForFluxCalibration.py'):
    fcalpresent = True
```

```
if os.path.exists('../script/scriptForFluxCalibration.py'):
    fcalpresent = True
    spacefactor = 1.

impreppresent = False
if os.path.exists('../script/scriptForImagingPrep.py'):
    impreppresent = True
    spacefactor = 1.

polcalpresent = False
if os.path.exists('../script/scriptForPolCalibration.py'):
    polcalpresent = True
    spacefactor = 1.

spaceneed = packspace*(11.+spacefactor*3.)

if (savingslevel==1):
    print 'Will delete intermediate MSs named *.ms.split to save disk space.'
    spaceneed = packspace*(7.+spacefactor*3.)
elif (savingslevel==2):
    print 'Will delete intermediate MSs named *.ms and *.ms.split to save disk space.'
    spaceneed = packspace*(3.+spacefactor*3.)
elif (savingslevel>=3):
    print 'Will delete all intermediate MSs to save disk space.'
    spaceneed = packspace*(3.+spacefactor*3.)


print 'Found ',avspace,' MB of available free disk space.'
print 'Expect to need up to ',spaceneed,' MB of free disk space.'
if(spaceneed>avspace):
    sys.exit('ERROR: not enough free disk space. Need at least '+str(spaceneed)+' MB.')

asdmnames = glob.glob('uid*.asdm.sdm')


if len(asdmnames) == 0:
    sys.exit('ERROR: No ASDM found in directory \"raw\".')

print 'Found the following ASDMs:', asdmnames

for i in range(len(asdmnames)):
    asdmnames[i] = asdmnames[i].replace('.asdm.sdm', '')


scriptasdms = []
for i in range(len(scriptnames)):
    scriptasdms.append(scriptnames[i].replace('.ms.scriptForCalibration.py', ''))

allasdms = []
allasdms.extend(scriptasdms)
allasdms.extend(pprasdms)

missing = []

if sorted(asdmnames) != sorted(allasdms):
    print "WARNING: Inconsistency between ASDMs and calibration scripts"
    print "         Calibration info available for: ", sorted(allasdms)
    print "         ASDMs available in directory raw: ", sorted(asdmnames)
    for myname in allasdms:
        if not (myname in asdmnames):
            missing.append(myname)
    if len(missing)==0:
        print "       The ASDMs without calibration info are probably \"QA semipass\" data which were"
        print "       not used to create the science products and are not needed to achieve the science goal."
        print "       Only the ASDMs for which there is calibration information will be calibrated."
    else:
        print "ERROR: the following ASDMs have calibration information but are absent from directory \"raw\":"
        print missing
        print "Will try to proceed with the rest ..."
        for myname in missing:
            if myname in scriptasdms:
                scriptasdms.remove(myname)
```

```
            if myname in scriptasdms:
                scriptasdms.remove(myname)
            if myname in pprasdms:
                pprasdms.remove(myname)
            if myname in allasdms:
                allasdms.remove(myname)
        if(len(allasdms)==0):
            sys.exit('ERROR: Nothing to process.')

ephnames = glob.glob('../calibration/*.eph')

if len(ephnames)>0:
    print "Note: this dataset uses external ephemerides."
    print "      You can find them in directory \"calibration\"."

if os.path.exists('../calibrated') and not globals().has_key("USEMS"):
    os.chdir('../calibrated')
    sys.exit('WARNING: will stop here since directory '+os.path.abspath(os.path.curdir)
             +' already exists.\nPlease delete it first and then try again.')

if not globals().has_key("USEMS"):
    print 'Creating destination directory for calibrated data.'
    os.mkdir('../calibrated')
else:
    print 'You have set USEMS. Will use your pre-imported MSs rather than importing them from the ASDMs.'
    for asdmname in scriptasdms:
        if not os.path.exists('../calibrated/'+asdmname+'.calibration/'+asdmname+'.ms'):
            print 'When USEMS is set, you must have created the directory \"calibrated\" and'
            print 'put the imported raw MSs \"uid*.ms\" in individual working directories'
            print 'named \"uid*.calibration\" inside \"calibrated\".'
            sys.exit('ERROR: cannot find calibrated/'+asdmname+'.calibration/'+asdmname+'.ms')

os.chdir('../calibrated')


for asdmname in scriptasdms:

    print 'Processing ASDM '+asdmname

    if not globals().has_key("USEMS"):
        os.mkdir(asdmname+'.calibration')

    os.chdir(asdmname+'.calibration')

    if not os.path.exists('../../raw/'+asdmname+'.asdm.sdm'):
        sys.exit('ERROR: cannot find raw/'+asdmname+'.asdm.sdm')

    os.system('ln -sf ../../raw/'+asdmname+'.asdm.sdm '+asdmname)

    for ephname in ephnames:
        os.system('ln -sf ../'+ephname)

    execfile('../../script/'+asdmname+'.ms.scriptForCalibration.py')

    if not os.path.exists(asdmname+'.ms.split.cal'):
        print 'ERROR: '+asdmname+'.ms.split.cal was not created.'
    else:
        print asdmname+'.ms.split.cal was produced successfully, moving it to \"calibrated\" directory.'
        os.system('mv '+asdmname+'.ms.split.cal ..')
        if (savingslevel>=2):
            print 'Deleting intermediate MS ', asdmname+'.ms'
            os.system('rm -rf '+asdmname+'.ms')
        if (savingslevel>=1):
            print 'Deleting intermediate MS ', asdmname+'.ms.split'
            os.system('rm -rf '+asdmname+'.ms.split')

    os.chdir('..')

if (len(pprasdms)>0):

    if pipererun:
        print 'Processing the ASDMs ', pprasdms, ' in pipeline rerun.'
```

```
if pipererun:
    print 'Processing the ASDMs ', pprasdms, ' in pipeline rerun.'
else:
    print 'Processing the ASDMs ', pprasdms, ' using pipeline restore.'

    os.mkdir('rawdata')
    os.chdir('rawdata')
    for asdmname in pprasdms:
        if not os.path.exists('../../raw/'+asdmname+'.asdm.sdm'):
            sys.exit('ERROR: cannot find raw/'+asdmname+'.asdm.sdm')

        os.system('ln -sf ../../raw/'+asdmname+'.asdm.sdm '+asdmname)

    os.chdir('..')

    os.system('ln -sf ../calibration products')

os.mkdir('working')
os.chdir('working')

if pipererun:
    for asdmname in pprasdms:
        if not os.path.exists('../../raw/'+asdmname+'.asdm.sdm'):
            sys.exit('ERROR: cannot find raw/'+asdmname+'.asdm.sdm')

        os.system('ln -sf ../../raw/'+asdmname+'.asdm.sdm '+asdmname)

    if istppipe:
        os.system('cp ../../calibration/jyperk.csv .')

    os.system('cp ../../calibration/*flagtemplate.txt .')

    fluxfiles = glob.glob("../../calibration/*flux.csv")
    if len(fluxfiles)>0:
        if len(fluxfiles)==1:
            os.system('cp ../../calibration/*flux.csv ./flux.csv')
        else:
            print fluxfiles
            sys.exit('ERROR: found more than one *flux.csv file in directory "calibration"')

    antennaposfiles = glob.glob("../../calibration/*antennapos.csv")
    if len(antennaposfiles)>0:
        if len(antennaposfiles)==1:
            os.system('cp ../../calibration/*antennapos.csv ./antennapos.csv')
        else:
            print antennaposfiles
            sys.exit('ERROR: found more than one *antennapos.csv file in directory "calibration"')

    contfiles = glob.glob("../../calibration/*cont.dat")
    if len(contfiles)>0:
        if len(contfiles)==1:
            os.system('cp ../../calibration/*cont.dat ./cont.dat')
        else:
            print contfiles
            sys.exit('ERROR: found more than one *cont.dat file in directory "calibration"')

    print "Directory \"working\" set up for pipeline re-run:"
    os.system('ls -l')

    execfile('../../script/'+p2scriptnames[0])

else:
    print "now running ", pscriptnames[0]
    execfile('../../script/'+pscriptnames[0])

for asdmname in pprasdms:
    if not os.path.exists(asdmname+'.ms'):
        print 'ERROR: '+asdmname+'.ms was not created.'
    elif pipererun and istppipe:
        tpmsnames = glob.glob(asdmname+'.PM*.ms')
        if len(tpmsnames)==0:
            print 'ERROR: '+asdmname+'.PM*.ms was not created.'
```

```
        if len(tpmsnames)==0:
            print 'ERROR: '+asdmname+'.PM*.ms was not created.'
        else:
            os.system('mv '+asdmname+'.PM*.ms ..')

        if (savingslevel>=2):
            print 'Deleting intermediate MS ', asdmname+'.ms'
            os.system('rm -rf '+asdmname+'.ms')
    else:
        msmd.open(asdmname+'.ms')
        targetspws = msmd.spwsforintent('OBSERVE_TARGET*')
        sciencespws = ''
        outputspws = ''
        i = 0
        for myspw in targetspws:
            if msmd.nchan(myspw)>4:
                        sciencespws += str(myspw)+','
                        outputspws += str(i)+','
                        i += 1
        sciencespws = sciencespws.rstrip(',')
        outputspws = outputspws.rstrip(',')
        msmd.close()
        print 'Splitting out science SPWs for '+asdmname+': '+sciencespws+' -> '+outputspws
        split(vis=asdmname+'.ms', outputvis=asdmname+'.ms.split.cal', spw = sciencespws)
        if not os.path.exists(asdmname+'.ms.split.cal'):
            print 'ERROR: '+asdmname+'.ms.split.cal was not created.'
        else:
            if (savingslevel>=2):
                print 'Deleting intermediate MS ', asdmname+'.ms'
                os.system('rm -rf '+asdmname+'.ms')
            os.system('mv '+asdmname+'.ms.split.cal ..')

os.chdir('..')

if polcalpresent:
    print 'Executing scriptForPolCalibration.py ...'
    execfile('../script/scriptForPolCalibration.py')

if fcalpresent:
    print 'Executing scriptForFluxCalibration.py ...'
    execfile('../script/scriptForFluxCalibration.py')

if impreppresent:
    print 'Executing scriptForImagingPrep.py ...'
    execfile('../script/scriptForImagingPrep.py')

if (savingslevel>=3) and os.path.exists('calibrated.ms'):
    for asdmname in allasdms:
        print 'Deleting intermediate MS ', asdmname+'.ms.split.cal'
        os.system('rm -rf '+asdmname+'.ms.split.cal')

print 'Done. Please find results in current directory.'
```

Run

① **READ the README!**

② Go to '`scripts`' folder

③ Start the corresponding CASA version:
- `casapy-XX`
- `casapy-XX --pipeline`

④ Optional to delete intermediate files:
- `SPACESAVING=N`

⑤ `execfile('scriptForPI.py')`

WARNINGS/ERRORS

If you have additional ASDMs for which there is no calibration info available:

```
WARNING: Inconsistency between ASDMs and calibration scripts
  Calibration info available for: uid...
  ASDMs available in directory raw: uid...
  Only the ASDMs for which there is calibration information
    will be calibrated
```

If you have not downloaded and unpacked all ASDMs for which there is calibration info:

```
ERROR: the following ASDMs have calibration information but are absent
  from directory "raw": uid...
  Will try to proceed with the rest ...
```


Credit: D. Petry (ESO)

scriptForPI.py

uid*.scriptForCalibration.py	**Luke Maud**
scriptForImaging.py	**Yanett Contreras**