A Crash Course in CASA

With a focus on calibration

Amy Kimball

North American ALMA Science Center

Atacama Large Millimeter/submillimeter Array

Expanded Very Large Array

Robert C. Byrd Green Bank Telescope

Very Long Baseline Array

NRAO

CASA

Common Astronomy Software Applications

- The offline data reduction package for ALMA and EVLA
  HANDLES BOTH INTERFEROMETRIC AND SINGLE-DISH ALMA DATA
- Current version: 3.4.0
  NEW RELEASES ABOUT EVERY 6 MONTHS
- CASA home: http://casa.nrao.edu
  DOWNLOAD, COOKBOOK, REFERENCE, EXAMPLE SCRIPTS, MAILING LISTS
- Training material on "CASAguides" wiki: http://casaguides.nrao.edu
- NRAO helpdesk: http://help.nrao.edu
- NRAO user's forum: https://science.nrao.edu/forums/

Outline

- **CASA interface: Python, tools, and tasks**
- Structure of CASA data
- Basic calibration flow in CASA
- Example calibration task: focus on *gaincal*
- ALMA online calibration

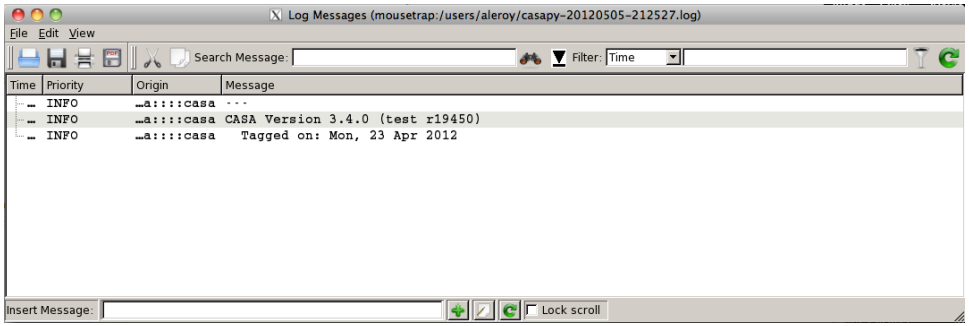
casapy Shell

- Start CASA by typing `casapy`

  VERSION NUMBER AND LOGGER WILL APPEAR, YOU GET AN IPYTHON PROMPT

```
CASA Version 3.4.0 (r19450)
  Compiled on: Tue 2012/04/24 03:44:22 UTC
This is my initialization file in ~/.casa/init.py
... appending my script directory to PYTHONPATH.
--------------------------------------------------------------------------------
    For help use the following commands:
    tasklist               - Task list organized by category
    taskhelp               - One line summary of available tasks
    help taskname          - Full help for task
    toolhelp               - One line summary of available tools
    help par.parametername - Full help for parameter name
    Single Dish sd* tasks are available after asap_init() is run
--------------------------------------------------------------------------------
Activating auto-logging. Current session state plus future input saved.
Filename       : ipython-20120505-212545.log
Mode           : backup
Output logging : False
Raw input log  : False
Timestamping   : False
State          : active
*** Loading ATNF ASAP Package...
*** ... ASAP (trunk rev#19449) import complete ***

CASA <2>:
```



- Python tips (tutorials at http://python.org/doc):
  - Indentation matters! So careful with cut/paste (a few lines at a time)
  - or use `cpaste` (type `cpaste`, paste code, end with a line of "`--`")
  - Run shell commands with leading "`!`", e.g., "`!du –hc`"
  - To run a script: `execfile('scriptname.py')`

"Tasks"

- Tasks – high-level functionality
  - Python wrapper around the toolkit and pythoncode
  - Accessed via python function call or parameter setting interface
  - List CASA tasks with command `tasklist` or `taskhelp`
  - Most data reduction and tutorials and CASA guides focus on tasks

Import/export	Information	Editing	Manipulation
exportasdm	imhead	fixplanets	concat
exportfits	imstat	fixvis	conjugatevis
exportuvfits	imval	flagautocorr	cvel
importasdm	listcal	flagcmd	fixvis
importfits	listfits	flagdata	hanningsmooth
importfitsidi	listhistory	flagmanager	imhead
importuvfits	listobs	msview	msmoments
importvla	listvis	plotms	plotms
(importevla)	plotms	plotxy	plotxy
(importgmrt)	plotuv	(flagdata2)	split
	plotxy	(testautoflag)	testconcat
	vishead	(tflagcmd)	uvcontsub
	visstat	(tflagdata)	vishead
	(asdmsummary)		(statwt)
	(listsdm)		(uvcontsub3)
			{uvcontsub2}

"Tools"

- Tools – low level, complete functionality
  - Interface to underlying C++ code
  - Intended for power users, less user-friendly, less well-documented
  - Objects: call with `<tool>.<method>`
  - List available tools with command `toolhelp`

```
Available tools:

at : Juan Pardo ATM library
cb : Calibration utilities
cl : Component list utilities
cp : Cal solution plotting utilities
cs : Coordinate system utilities
cu : Class utilities
dc : Deconvolver utilities
fg : Flagging/Flag management utilities
ia : Image analysis utilities
im : Imaging utilities
me : Measures utilities
ms : MeasurementSet (MS) utilties
mp : MS plotting (data (amp/phase) versus other quantities)
pm : PlotMS utilities
po : Imagepol utilities
rg : Region manipulation utilities
tb : Table utilities (selection, extraction, etc)
tp : Table plotting utilities
qa : Quanta utilities
sl : Spectral line import and search
tf : Test flagger utilities
sm : Simulation utilities
vp : Voltage pattern/primary beam utilties
---
pl : pylab functions (e.g., pl.title, etc)
sd : Single dish utilities
---
```

Task Syntax

- get detailed help with `help(<taskname>)`
- Two ways to call tasks:
  - As a function with arguments:
    UNSPECIFIED PARAMETERS USE DEFAULT VALUES

```
gaincal(vis='mydata.ms', caltable='caltable.cal', field='2')
```

  - Standard, interactive task mode:
    set global input variables ahead of time, then <taskname> OR go
    OMITTING "TASKNAME" OPERATES ON CURRENT TASK

default(<taskname>)	sets task's parameters to default values
inp(<taskname>)	see task's parameter settings (input values)
saveinputs(<taskname>)	saves parameters to <taskname>.saved
tget(<taskname>)	retrieves parameters (<taskname>.last)

Standard Task Interface

Examine task parameters (inputs) with *inp* :

```
CASA <49>: inp gaincal
---------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 = 'mydata.ms'       #  Name of input visibility file
caltable            = 'mytable.bandpass.bpcal' #  Name of output gain calibration table
field               =         ''        #  Select field using field id(s) or field name(s)
spw                 =         ''        #  Select spectral window/channels
intent              =         ''        #  Select observing intent
selectdata          =       True        #  Other data selection parameters
     timerange      =         ''        #  Select data based on time range
     uvrange        =         ''        #  Select data within uvrange (default units meters)
     antenna        =         ''        #  Select data based on antenna/baseline
     scan           =         ''        #  Scan number range
     observation    =         ''        #  Select by observation ID(s)
     msselect       =         ''        #  Optional complex data selection (ignore for now)

solint              =      'inf'        #  Solution interval: egs. 'inf', '60s' (see help)
combine             =         ''        #  Data axes which to combine for solve (scan, spw, and/or
                                        #   field)
preavg              = 'hogwarts'        #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''        #  Reference antenna name(s)
minblperant         =          4        #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0        #  Reject solutions below this SNR
solnorm             =      False        #  Normalize average solution amplitudes to 1.0 (G, T only)
```

Standard Task Interface

Default values in BLACK

```
CASA <49>: inp gaincal
---------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 = 'mydata.ms'        #  Name of input visibility file
caltable            = 'mytable.bandpass.bpcal' #  Name of output gain calibration table
field               =         ''         #  Select field using field id(s) or field name(s)
spw                 =         ''         #  Select spectral window/channels
intent              =         ''         #  Select observing intent
selectdata          =       True         #  Other data selection parameters
     timerange      =         ''         #  Select data based on time range
     uvrange        =         ''         #  Select data within uvrange (default units meters)
     antenna        =         ''         #  Select data based on antenna/baseline
     scan           =         ''         #  Scan number range
     observation    =         ''         #  Select by observation ID(s)
     msselect       =         ''         #  Optional complex data selection (ignore for now)

solint              =      'inf'         #  Solution interval: egs. 'inf', '60s' (see help)
combine             =         ''         #  Data axes which to combine for solve (scan, spw, and/or
                                         #   field)
preavg              = 'hogwarts'         #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''         #  Reference antenna name(s)
minblperant         =          4         #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0         #  Reject solutions below this SNR
solnorm             =      False         #  Normalize average solution amplitudes to 1.0 (G, T only)
```

Standard Task Interface

Expandable parameters are highlighted

Sub-parameters indented

```
CASA <49>: inp gaincal
---------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =  'mydata.ms'        #  Name of input visibility file
caltable            =  'mytable.bandpass.bpcal' #  Name of output gain calibration table
field               =         ''          #  Select field using field id(s) or field name(s)
spw                 =         ''          #  Select spectral window/channels
intent              =         ''          #  Select observing intent
selectdata          =       True          #  Other data selection parameters
     timerange      =         ''          #  Select data based on time range
     uvrange        =         ''          #  Select data within uvrange (default units meters)
     antenna        =         ''          #  Select data based on antenna/baseline
     scan           =         ''          #  Scan number range
     observation    =         ''          #  Select by observation ID(s)
     msselect       =         ''          #  Optional complex data selection (ignore for now)

solint              =      'inf'          #  Solution interval: egs. 'inf', '60s' (see help)
combine             =         ''          #  Data axes which to combine for solve (scan, spw, and/or
                                          #   field)
preavg              = 'hogwarts'          #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''          #  Reference antenna name(s)
minblperant         =          4          #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0          #  Reject solutions below this SNR
solnorm             =      False          #  Normalize average solution amplitudes to 1.0 (G, T only)
```

Standard Task Interface

User set values in BLUE

Erroneous values in RED

```
CASA <49>: inp gaincal
---------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 = 'mydata.ms'        #  Name of input visibility file
caltable            = 'mytable.bandpass.bpcal' #  Name of output gain calibration table
field               =         ''         #  Select field using field id(s) or field name(s)
spw                 =         ''         #  Select spectral window/channels
intent              =         ''         #  Select observing intent
selectdata          =       True         #  Other data selection parameters
     timerange      =         ''         #  Select data based on time range
     uvrange        =         ''         #  Select data within uvrange (default units meters)
     antenna        =         ''         #  Select data based on antenna/baseline
     scan           =         ''         #  Scan number range
     observation    =         ''         #  Select by observation ID(s)
     msselect       =         ''         #  Optional complex data selection (ignore for now)

solint              =      'inf'         #  Solution interval: egs. 'inf', '60s' (see help)
combine             =         ''         #  Data axes which to combine for solve (scan, spw, and/or
                                         #   field)
preavg              = 'hogwarts'         #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''         #  Reference antenna name(s)
minblperant         =          4         #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0         #  Reject solutions below this SNR
solnorm             =      False         #  Normalize average solution amplitudes to 1.0 (G, T only)
```

Outline

- CASA interface: Python, tools, and tasks
- **Structure of CASA data**
- Basic calibration flow in CASA
- Example calibration task: focus on *gaincal*
- ALMA online calibration

NRAO

Measurement Set

- CASA stores u-v data in directories called "Measurement Sets"
  TO DELETE THEM USE `os.system("rm –rf my_data.ms")`

- These data sets store two copies of the data (called "columns"):

"Data" Column	"Corrected" Column
Contains the raw, unprocessed measurements.	Usually created by applying one or more calibration terms to the data.

- Additionally a "model" may be stored separately.
  THIS IS USED TO CALCULATE WHAT THE TELESCOPE SHOULD HAVE OBSERVED.

- Each data point may also be "flagged," i.e., marked bad.
  IN THIS CASE IT IS IGNORED (TREATED AS MISSING) BY CASA OPERATIONS.

`listobs`

- Measurement sets contain a mix of data:
  - One or more spectral windows
  - One or more fields (e.g., source, phase calibrator, flux calibrator)
  - Data from several antennas
  - Data organized into discrete scans

- Inspect the contents of your measurement set using `listobs`.
  - Can print output to a file or the logger (use the file option!)
  - Verbose (detailed time log) output possible
  - Summarizes fields, antennas, sources, spectral windows

Always run `listobs` first to get oriented!

Calibration Tables

- Calibration yields estimates of phase and amplitude corrections.
  E.G., AS A FUNCTION OF TELESCOPE, TIME, FREQUENCY, POLARIZATION.

- CASA stores these corrections in directories called "calibration tables."
  TO DELETE THEM USE `os.system("rm –rf my_table.cal")`

- These are created by calibration tasks:
  E.G., `gaincal, bandpass, gencal`

- Applied via "`applycal`" to the data column and saved as corrected.

Outline

- CASA interface: Python, tools, and tasks
- Structure of CASA data
- **Basic calibration flow in CASA**
- Example calibration task: focus on *gaincal*
- ALMA online calibration

NRAO

Key Tasks for Calibration

CASA
Common Astronomy Software Applications

Derive Calibration Tables

- `setjy`: set "model" (correct) visibilities using known model for a calibrator
- `bandpass`: calculate bandpass calibration table (amp/phase vs frequency)
- `gaincal`: calculate temporal gain calibration table (amp/phase vs time)
- `fluxscale`: apply absolute flux scaling to calibration table from known source

Manipulate Your Measurement Set

- `flagdata`/`flagcmd`/`flagmanager`: flag (remove) bad data
- `applycal`: apply calibration table(s) from previous steps
- `split`: split off calibrated data from your ms (for imaging!)

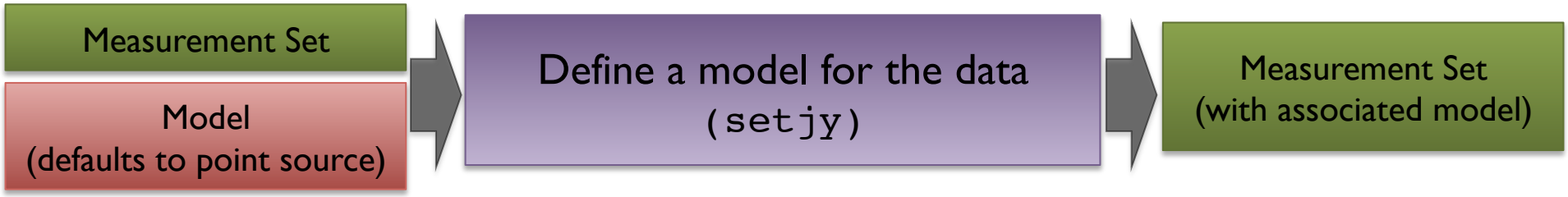
Inspect Your Data and Results

- `plotms`: inspect your data interactively
- `plotcal`: examine a calibration table

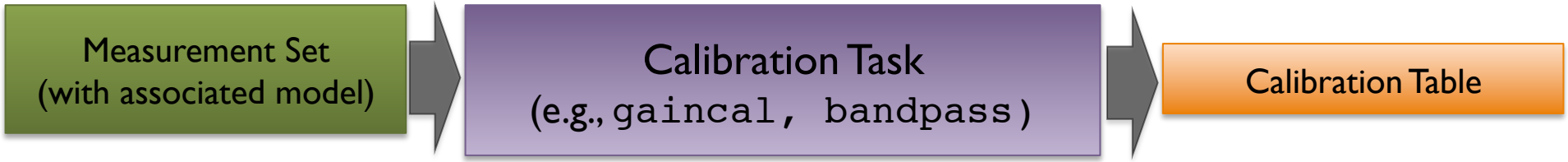
NRAO

Basic Calibration Flow

Define what the telescope SHOULD have seen.



Derive the corrections needed to make the data match the model.



Apply these corrections to derive the corrected (calibrated) data.

Schematic Calibration

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE TIME VARIABILITY

Calibrate the Amplitude and Phase vs. Time of Each Antenna
ASSUME TIME & FREQUENCY RESPONSE SEPARABLE, REMOVE FREQ. VARIABILITY

Set the Absolute Amplitude Scale With Reference to a Known Source
PLANET (MODELED), MONITORED QUASAR, ETC.

Apply all corrections to produce calibrated data

Schematic Calibration

Calibrate the Amplitude and Phase vs. Frequency of Each Antenna

`bandpass`

Bandpass Calibration Table

Calibrate the Amplitude and Phase vs. Time of Each Antenna

`gaincal`

Phase Calibration Table
Amplitude Calibration Table

Set the Absolute Amplitude Scale With Reference to a Known Source

`fluxscale`

Flux Calibration Table

Apply all corrections to produce calibrated data

`applycal`

Measurement Set

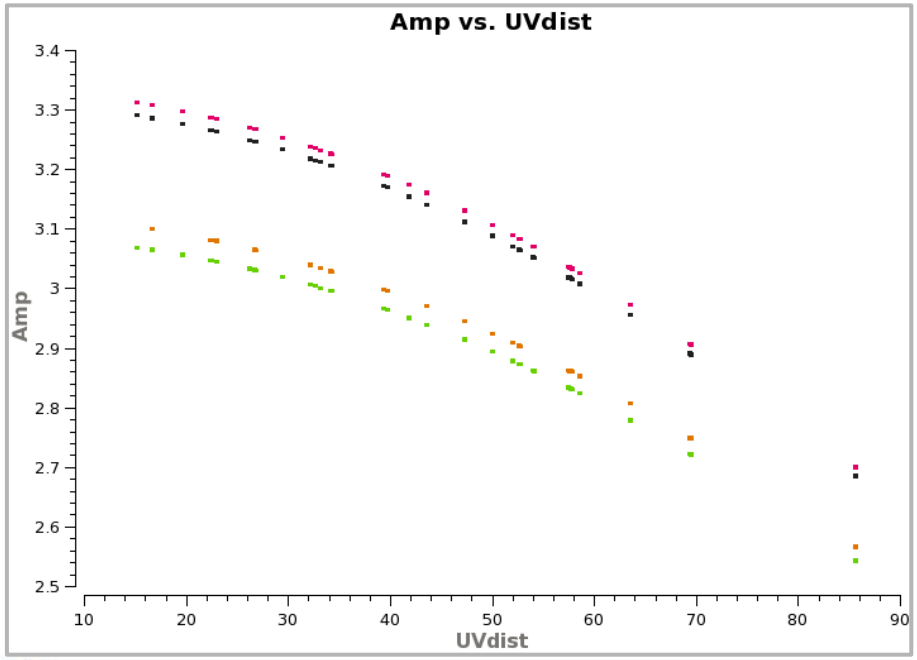
Corrected column now holds calibrated data.

Absolute flux calibration

Define a model for the flux calibrator source

`setjy`

Titan model (distributed with CASA)



For all other sources, default model is 1 Jy point source

Absolute flux calibration

Define a model for the flux calibrator source

`setjy`

Find calibration solutions to give best match of data to model

`gaincal`

Amplitude Calibration Table

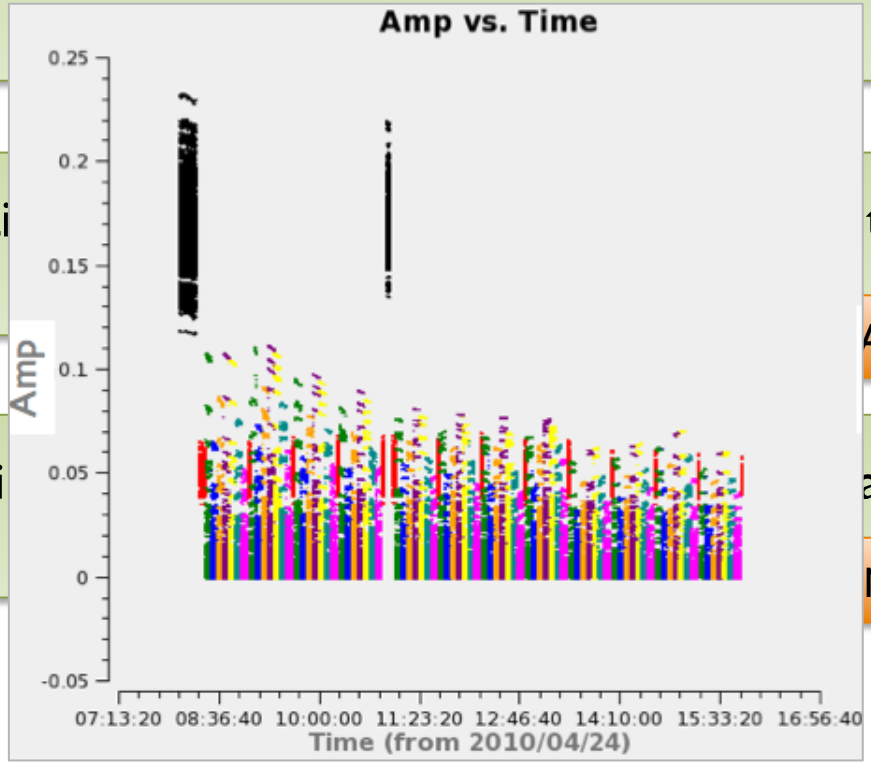
Transfer gain scaling from flux calibrator to phase calibrator

`fluxscale`

New Flux Calibration Table

Absolute flux calibration

Define a model for the flux calibrator source



Find calibrati … to model

Amplitude Calibration Table

Transfer gai … alibrator

New Flux Calibration Table

Fluxscale: applies constraint that field-dependent antenna gains are due solely to unknown flux densities of calibrators.

Outline

- CASA interface: Python, tools, and tasks
- Structure of CASA data
- Basic calibration flow in CASA
- **Example calibration task: focus on *gaincal***
- ALMA online calibration

NRAO

gaincal

Measurement Set

Data column holds observations.

(Optional)
One or More Calibration Tables
(applied on the fly before solution)

Associate a model (expected sky distribution) with the MS. (Default: assume point source)

`gaincal`
Solve for phase and amplitude response of each telescope as a function of time.
(Solutions derived to give best match of data to model once they are applied.)

- What time interval to solve over?
- Requirements for a good solution.
- Reference Antenna

Calibration Table
Later applied with `applycal`

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select field using field id(s) or field name(s)
spw                 =         ''            #  Select spectral window/channels
intent              =         ''            #  Select observing intent
selectdata          =      False            #  Other data selection parameters
solint              =      'inf'            #  Solution interval: egs. 'inf', '60s' (see help)
combine             =         ''            #  Data axes which to combine for solve (scan, spw,
                                            #   and/or field)
preavg              =       -1.0            #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''            #  Reference antenna name(s)
minblperant         =          4            #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select field using field id(s) or field name(s)
spw                 =         ''            #  Select spectral window/channels
intent              =         ''            #  Select observing intent
selectdata          =      False            #  Other data selection parameters
solint              =      'inf'            #  Solution interval: integration [default], ...
combine             =         ''            #  Data axes to combine for solving (scan, spw,
                                            #   and/or field)
preavg              =       -1.0            #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''            #  Reference antenna name(s)
minblperant         =          4            #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Input Measurement Set
(with model set, if needed)

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select field using field id(s) or field name(s)
spw                 =         ''            #  Select spectral window/channels
intent              =         ''            #  Select observing intent
selectdata          =      False            #  Other data selection parameters
solint              =      'inf'            #  Solution interval: integration [default], ...
combine             =         ''            #  Data axes to combine for solving (scan, spw,
                                            #   and/or field)
preavg              =       -1.0            #  Pre-averaging interval (sec) (rarely needed)
refant              =         ''            #  Reference antenna name(s)
minblperant         =          4            #  Minimum baselines _per antenna_ required for solve
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Output Calibration Table

(apply later with `applycal`)

CASA
Common Astronomy Software Applications

NRAO

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #
combine             =         ''            #  Data
                                            #   and/or
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referen
minblperant         =          4            #  Minimum
minsnr              =        3.0            #  Reject
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Options to select which data to consider:

e.g., select calibrator fields

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data a
                                            #   and/or
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimu
minsnr              =        3.0            #  Reject
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Time interval over which to solve.

(Only cross scan or spw boundaries with "combine")

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data
                                            #  and/or
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimur
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Reference Antenna

(pick a central one with little or no flagging)

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data
                                            #
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimur
minsnr              =        3.0            #  Reject
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Requirements for a solution in terms of S/N and # of baselines contributing

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data
                                            #   and/or
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimu
minsnr              =        3.0            #  Reject
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Normalize solutions?

gaincal

```
# gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data a
                                            #   and/or
preavg              =       -1.0            #  Preave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimum
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

What to solve for?

‘a’mplitude

‘p’hase

‘ap’ - both

gaincal

```
--------> inp(gaincal)
#  gaincal :: Determine temporal gains from calibrator observations
vis                 =         ''            #  Name of input visibility file
caltable            =         ''            #  Name of output gain calibration table
field               =         ''            #  Select
spw                 =         ''            #  Select
intent              =         ''            #  Select
selectdata          =      False            #  Other
solint              =      'inf'            #  Solutic
combine             =         ''            #  Data a
                                            #   and/
preavg              =       -1.0            #  Pre-ave
refant              =         ''            #  Referer
minblperant         =          4            #  Minimum
minsnr              =        3.0            #  Reject solutions below this SNR
solnorm             =      False            #  Normalize average solution amplitudes to 1.0 (G, T
                                            #   only)
gaintype            =        'G'            #  Type of gain solution (G,T,GSPLINE,K,KCROSS)
smodel              =         []            #  Point source Stokes parameters for source model.
calmode             =       'ap'            #  Type of solution: ('ap', 'p', 'a')
append              =      False            #  Append solutions to the (existing) table
gaintable           =       ['']            #  Gain calibration table(s) to apply on the fly
gainfield           =       ['']            #  Select a subset of calibrators from gaintable(s)
interp              =       ['']            #  Temporal interpolation for each gaintable (=linear)
spwmap              =         []            #  Spectral windows combinations to form for
                                            #   gaintables(s)
gaincurve           =      False            #  Apply internal VLA antenna gain curve correction
opacity             =         []            #  Opacity correction to apply (nepers), per spw
parang              =      False            #  Apply parallactic angle correction on the fly
async               =      False            #  If true the taskname must be started using
                                            #   gaincal(...)
```

Calibration tables to apply before solution:

e.g., apply bandpass calibration before `gaincal`

Outline

- CASA interface: Python, tools, and tasks
- Structure of CASA data
- Basic calibration flow in CASA
- Example calibration task: focus on *gaincal*
- **ALMA online calibration**

NRAO

ALMA Online Calibration

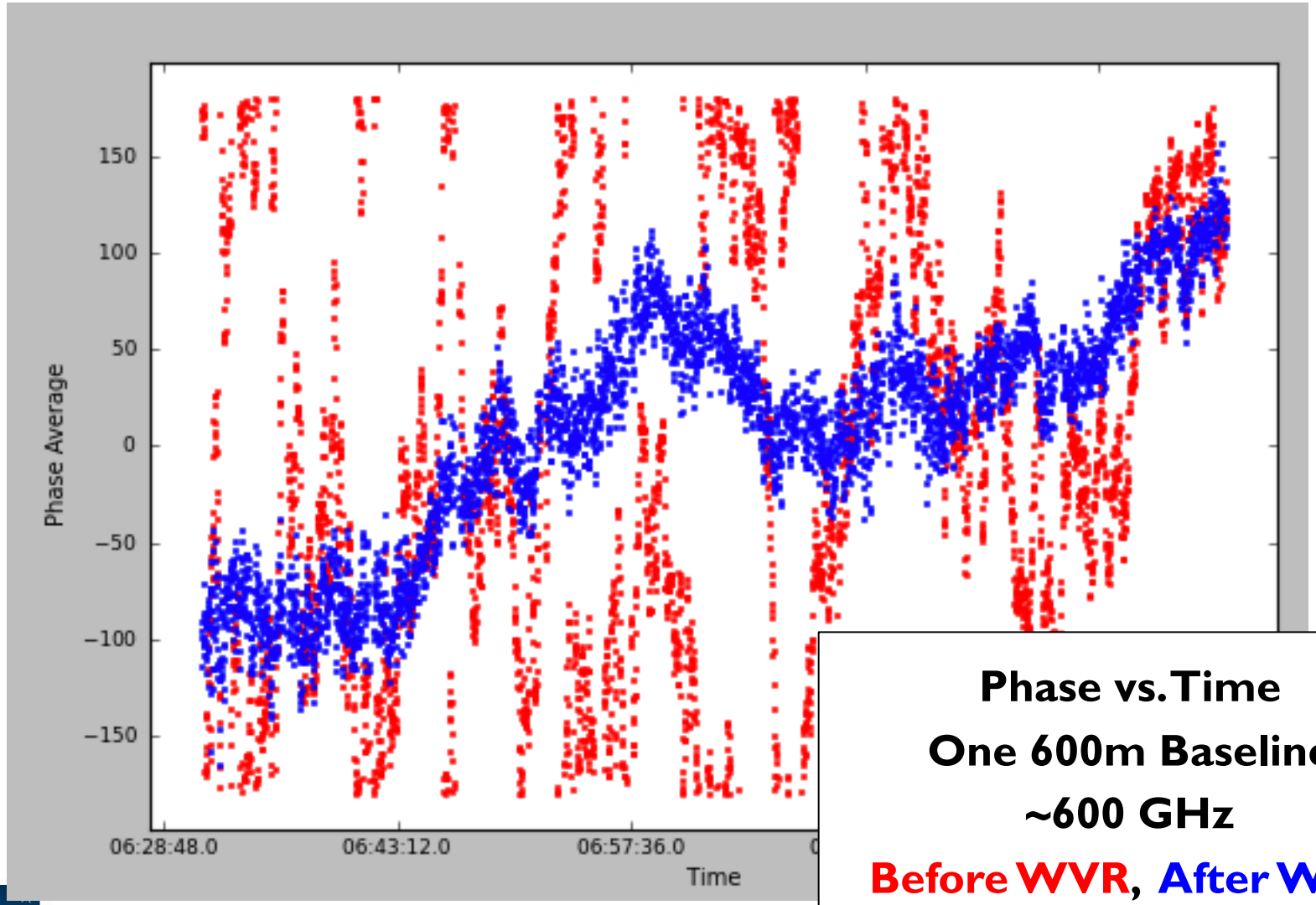
- System Temperature (Tsys) – atmospheric emission/opacity
  - Key to gain transfer across elevation
  - Amplitude calibration, variable with frequency (observed in "TDM")

- Water Vapor Radiometer (WVR) – phase delay due to atmosphere
  - Key to correct short-timescale phase variations
  - Phase calibration, variable with time

These are provided by the observatory (eventually applied online).

  - Apply them as first step (or start with provided pre-applied versions)
  - In either case, inspect these tables to learn about data quality

ALMA Online Calibration



Phase vs. Time

One 600m Baseline

~600 GHz

Before WVR, After WVR

Your Turn

- Point your web browser at the Synthesis Imaging School CASA guide.

  `http://casaguides.nrao.edu/index.php?title=TWHydraBand7_SS12`

- Decide whether to start with WVR and Tsys applied.

- Work end-to-end through the calibration of a single measurement set.
  THE FULL ONLINE GUIDES STEP THROUGH CALIBRATION FOR SEVERAL MSS.

- (Optional) Try writing a python script as you go.
  THIS IS VERY GOOD PRACTICE FOR ACTUAL REDUCTION.

- After lunch, we will image the results.
  DON'T WORRY, WE HAVE PROVIDED CALIBRATED DATA FOR THE AFTERNOON!

ASK IF YOU NEED HELP!