

A Crash Course in CASA

With a focus on calibration



Whoever

North American ALMA Science Center

Atacama Large Millimeter/submillimeter Array
Expanded Very Large Array
Robert C. Byrd Green Bank Telescope
Very Long Baseline Array



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 (released XX May 2012)
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>



Outline

- **CASA interface: Python, tools, and tasks**
- Structure of CASA data
- Key CASA tasks for data reduction/calibration
- CASA tasks for examining your data



casapy Shell

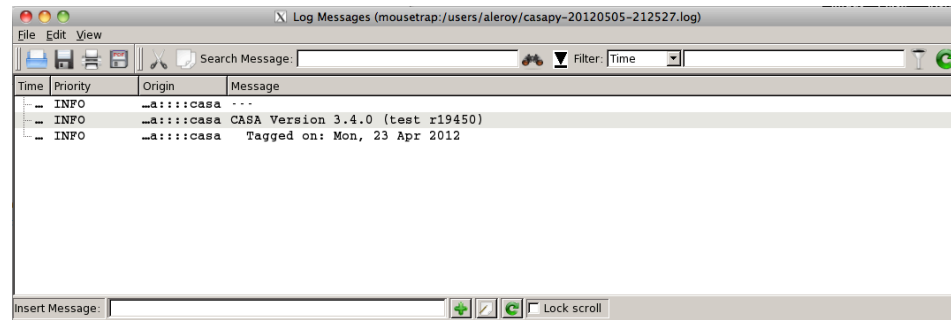
- Start CASA by typing `casapy`
VERSION NUMBER AND LOGGER WILL APPEAR, YOU GET AN IPYTHON PROMPT

```
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 <?: █
```



- 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
(importgmt)	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) utilities
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 utilities
---
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:
OMITTING “TASKNAME” OPERATES ON CURRENT TASK

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

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 @ (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 @ (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 (C, 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 (C, T only)
```

Outline

- CASA interface: Python, tools, and tasks
- **Structure of CASA data**
- Key CASA tasks for data reduction/calibration
- CASA tasks for examining your data

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”):

<p>“Data” Column</p> <p>Contains the raw, unprocessed measurements.</p>	<p>“Corrected” Column</p> <p>Usually created by applying one or more calibration terms to the data.</p>
---	---

- 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!

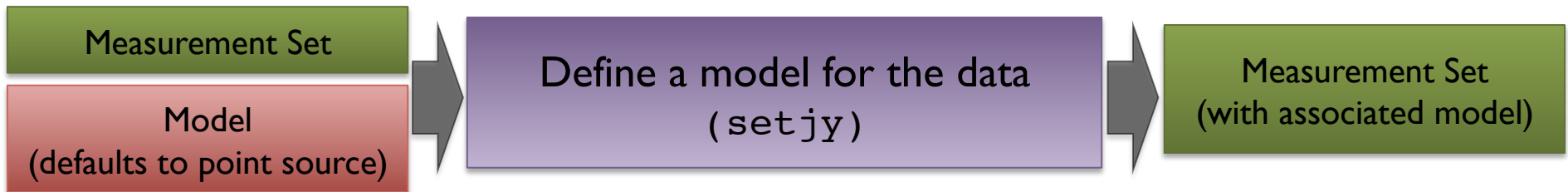
Calibration Tables

- 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_data.ms")`
- These are created by calibration tasks:
E.G., `gaincal`, `bandpass`, `gencal`
- Applied via “`applycal`” to the data column and saved as corrected.



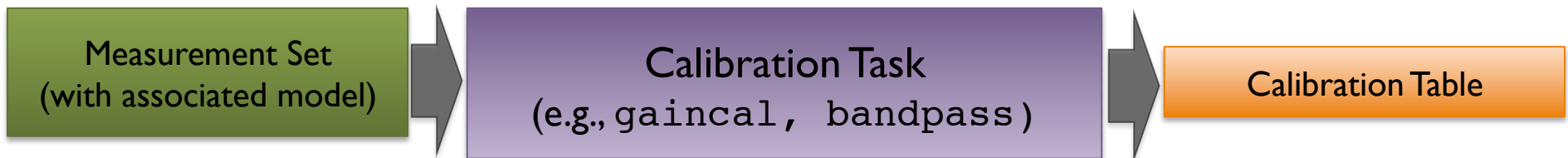
Basic Calibration Flow

Define what the telescope SHOULD have seen.



Basic Calibration Flow

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



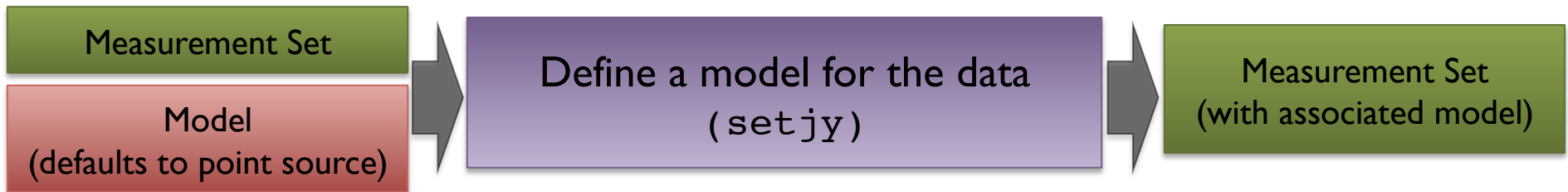
Basic Calibration Flow

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

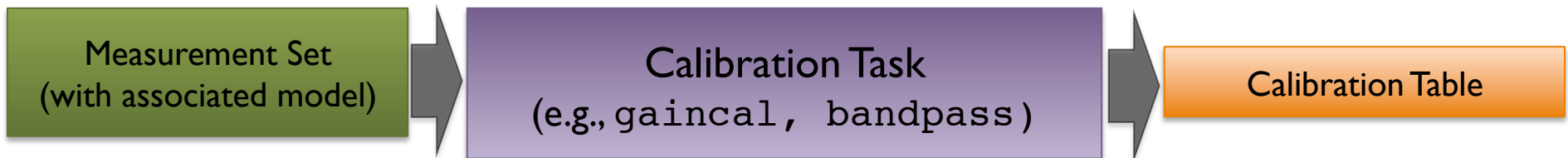


Basic Calibration Flow

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.



Outline

- CASA interface: Python, tools, and tasks
- Structure of CASA data
- **Key CASA tasks for data reduction/calibration**
- CASA tasks for examining your 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

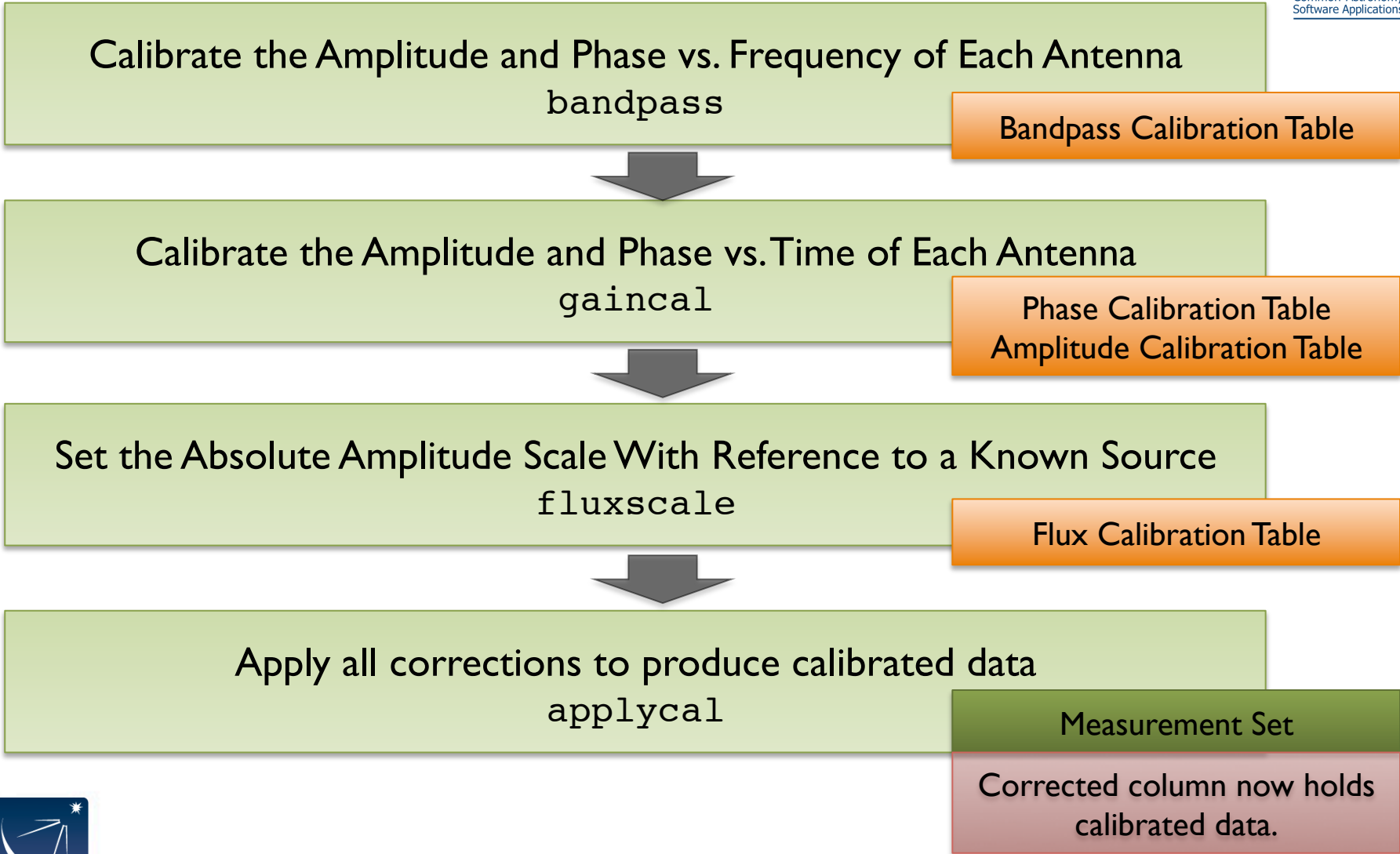


Set the Absolute Amplitude Scale With Reference to a Known Source
PLANET (MODELLED), MONITORED QUASAR, ETC.



Apply all corrections to produce calibrated data

Schematic Calibration



Key Tasks for Calibration

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

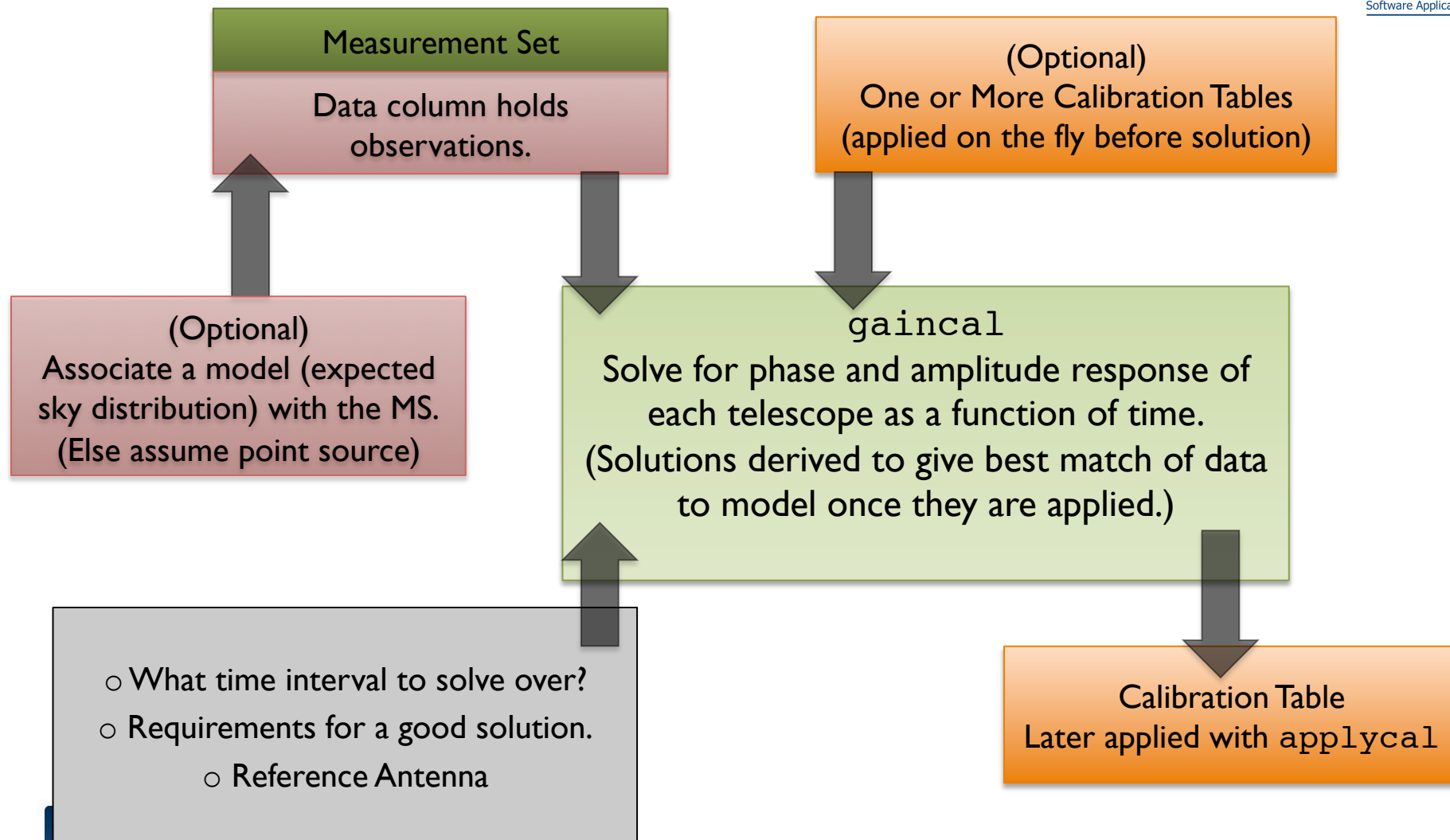
Manipulate Your Measurement Set

- `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

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; 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
asyncl            =      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 a subset of calibrators from gaintable(s)
spw = '' # Select a subset of spectral windows from spwmap(s)
intent = '' # Select a subset of observations from intent(s)
selectdata = False # Other options
solint = 'inf' # Solution interval (sec)
combine = '' # Data axis combination (antenna, frequency,
# 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 calibrators from visibility file
spw = '' # Select spectral windows
intent = '' # Select observation intents
selectdata = False # Other options
solint = 'inf' # Solution interval (sec)
combine = '' # Data axis combination
# 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)**

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 calibrators from gaincal(
spw                =      ''      # Select
intent            =      ''      # Select
selectdata       =      False    # Other o
solint             =      'inf'    # Solutio
combine           =      ''      # Data a
                    # and/or
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(...)

```

**Options to select which data to consider:
e.g., select calibrator fields**

gaincal

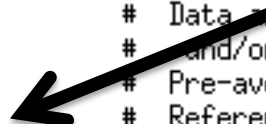
```
-----/ inp(gaincal)
# gaincal ;; Determine temporal gains from calibrator observations
vis = '' # Name of input visibility file
caltable = '' # Name of output gain calibration table
field = '' # Select a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from caltable(s)
intent = '' # Select a subset of observations from caltable(s)
selectdata = False # Other options
solint = 'inf' # Solution interval
combine = '' # Data combination
preavg = -1.0 # Pre-averaging
refant = '' # Reference antenna
minblperant = 4 # Minimum baseline per antenna
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(...)
```

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 a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from spwmap(s)
intent = '' # Select a subset of observations from intent(s)
selectdata = False # Other options
solint = 'inf' # Solution interval (seconds)
combine = '' # Data combination method
preavg = -1.0 # Pre-average flag
refant = '' # Reference antenna
minblperant = 4 # Minimum number of baselines per antenna
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 a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from spwmap(s)
intent = '' # Select a subset of observations from intent(s)
selectdata = False # Other options
solint = 'inf' # Solution interval
combine = '' # Data combination method
preavg = -1.0 # Pre-average flag
refant = '' # Reference antenna
minblperant = 4 # Minimum number of baselines per antenna
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(...)
```

**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 a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from caltable(s)
intent = '' # Select a subset of observations from caltable(s)
selectdata = False # Other options
solint = 'inf' # Solution interval
combine = '' # Data combination method
preavg = -1.0 # Pre-average solutions
refant = '' # Reference antenna
minblperant = 4 # Minimum number of baselines per antenna
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(...)
```

Normalize solutions?

gaincal

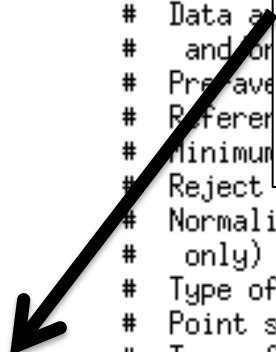
```
-----/ inp(gaincal)
# gaincal :: Determine temporal gains from calibrator observations
vis = '' # Name of input visibility file
caltable = '' # Name of output gain calibration table
field = '' # Select a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from caltable(s)
intent = '' # Select a subset of observations from caltable(s)
selectdata = False # Other options
solint = 'inf' # Solution interval
combine = '' # Data combination
preavg = -1.0 # Pre-average
refant = '' # Reference antenna
minblperant = 4 # Minimum baseline per antenna
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’plitude

‘p’hase

‘ap’ - both

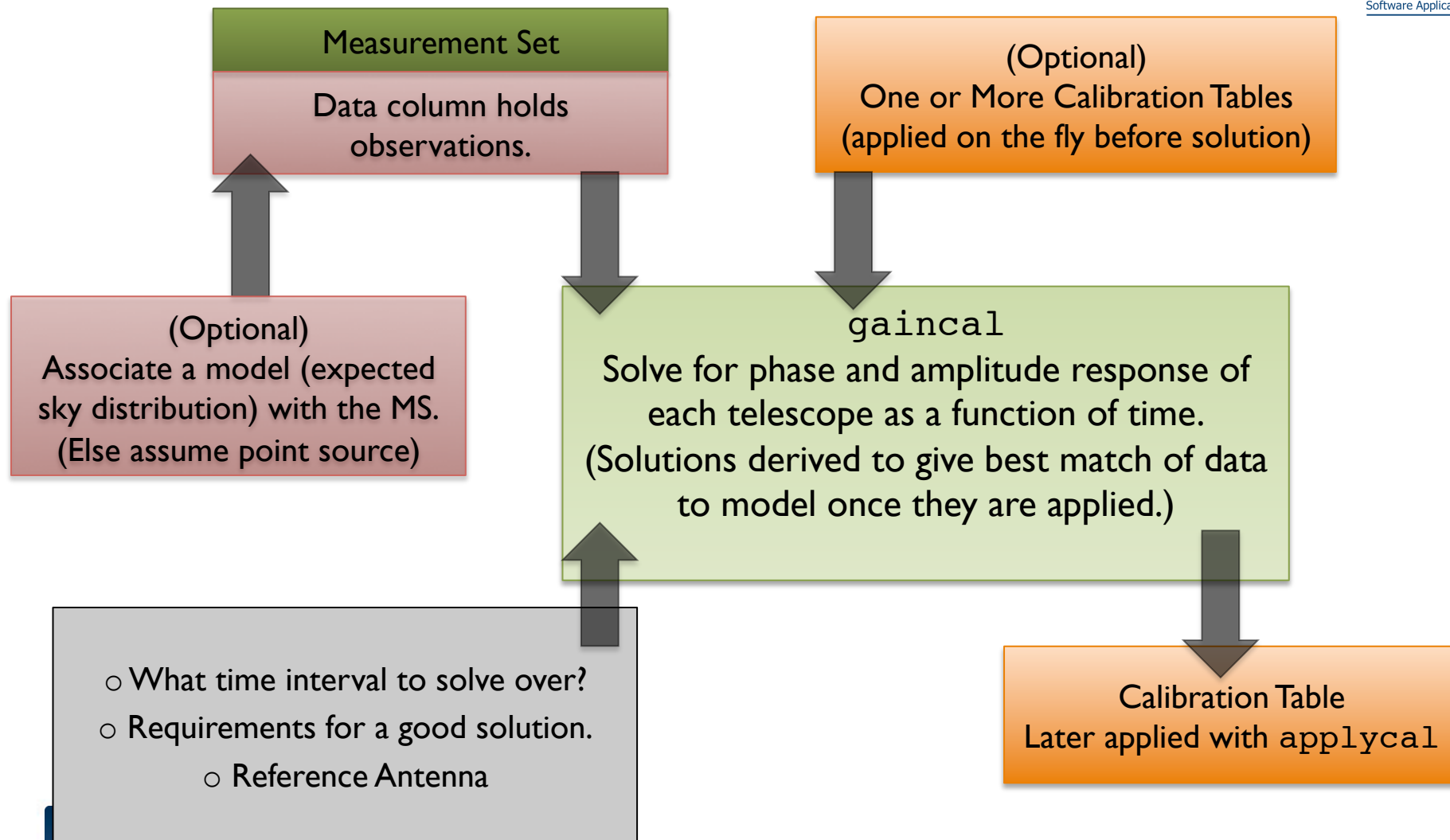


gaincal

```
-----/ inp(gaincal)
# gaincal ;; Determine temporal gains from calibrator observations
vis = '' # Name of input visibility file
caltable = '' # Name of output gain calibration table
field = '' # Select a subset of calibrators from caltable(s)
spw = '' # Select a subset of spectral windows from caltable(s)
intent = '' # Select a subset of observations from caltable(s)
selectdata = False # Other options
solint = 'inf' # Solution interval (seconds)
combine = '' # Data combination method
preavg = -1.0 # Pre-average gain solutions
refant = '' # Reference antenna
minblperant = 4 # Minimum number of baselines per antenna
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**

gaincal



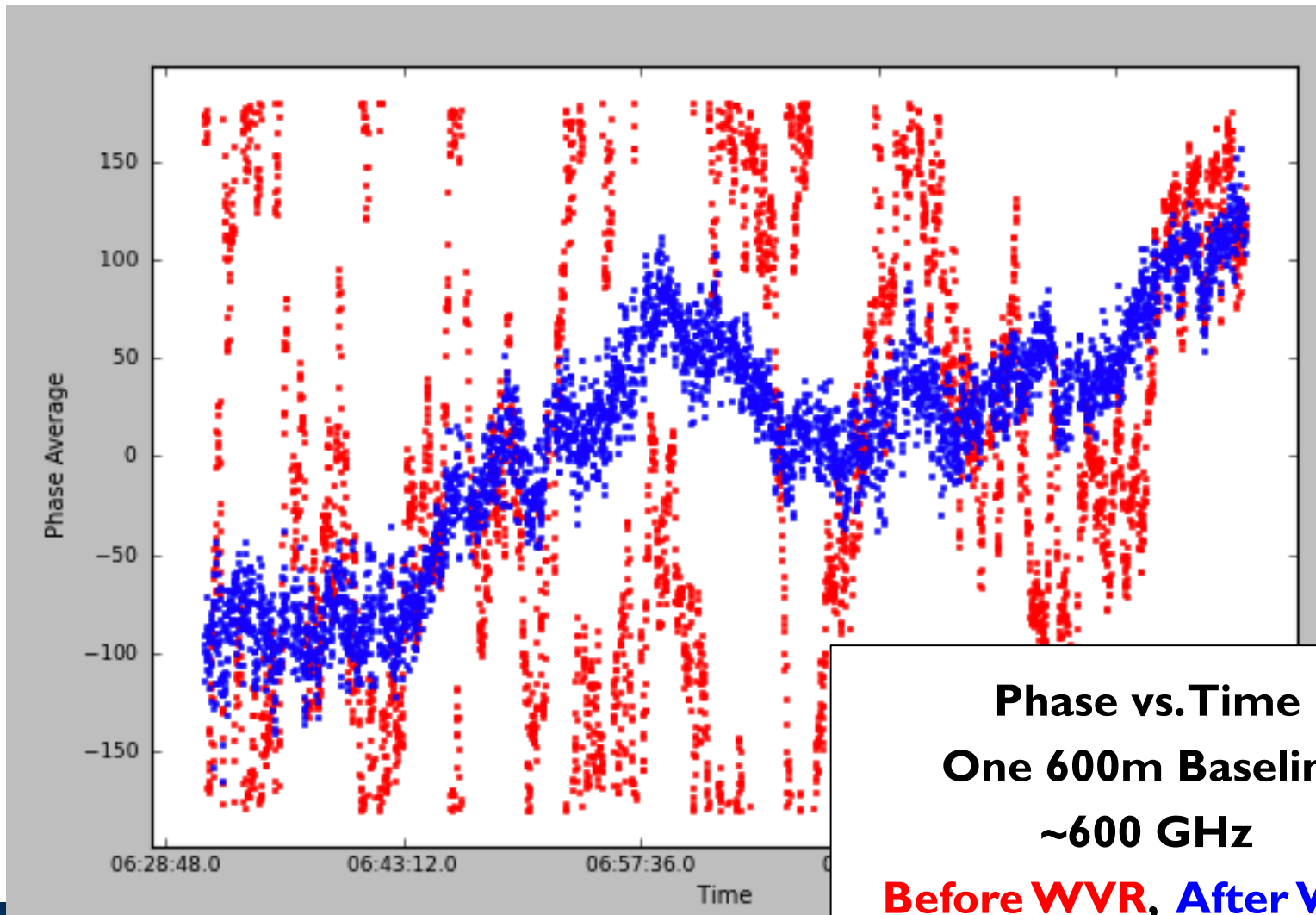
ALMA Online Calibration

- **System Temperature (T_{sys})** – 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!