LOFAR Imaging tutorial using WSClean

André Offringa

- We start from calibrated data (see previous tutorial)

  Everyone should have several calibrated sub-bands.
  NB: You should work on your own files: do not “share” measurement sets during imaging – the imager might write to it.

- Topics:
  - Making a quick dirty image
  - Cleaning
  - Multi-scale
  - LOFAR primary beam
  - Weighting, tapering
  - Wide bandwidth imaging

- Challenge:

  Make an image that looks better or is scientifically more valuable than mine.

We are going to use WSClean. A lot of help on WSClean is available at the WSClean wiki:

https://sourceforge.net/p/wsclean/wiki/Home/

WSClean is installed on the CEP clusters. Make it available with the following command:

`$ use Wsclean` (note the capital)

Check which version you are running:

```
$ wsclean --version
```

Get the WSClean command line help by running wsclean without parameters: `$ wsclean |less`

Whenever you run WSClean in this tutorial, be sure to inspect the output.

A quick look at the data

A quick look is useful...

- ...to check if calibration went well
- ...to determine a reasonable size and scale for the image

A quick look at the data

Pick a random sb and run wsclean as follows:

```
$ wsclean -size <width> <height> -scale <val>asec  \
     -name quick L456106_SB010_uv.dppp.MS.flg.ph
```

(Change the sb number to your random sb number)

Replace width and height by a number of pixels.

val is the image resolution, here specified in asec.

Determine good values for these for imaging this LOFAR set. You want to go a bit beyond the first beam null. Note that angularwidth ≈ width x scale

- Note that <val> and "asec" have no space between them, e.g.: "-scale 2.5asec"
- Other units can be specified, e.g.: "6amin", "50masec", "0.1deg"
- In order to keep processing fast for the tutorial, don't make images > 4k or wider than 20 deg. This quick imaging should not take more than ~3 min.
- WSClean will always automatically perform appropriate w-correction (i.e., corrections necessary for wide-field imaging)

A quick look at the data

Example command:

```
$ wsclean -size 1400 1400 -scale 50sec  \
     -name quick L456106_SB010_uv.dppp.MS.flg.ph
```

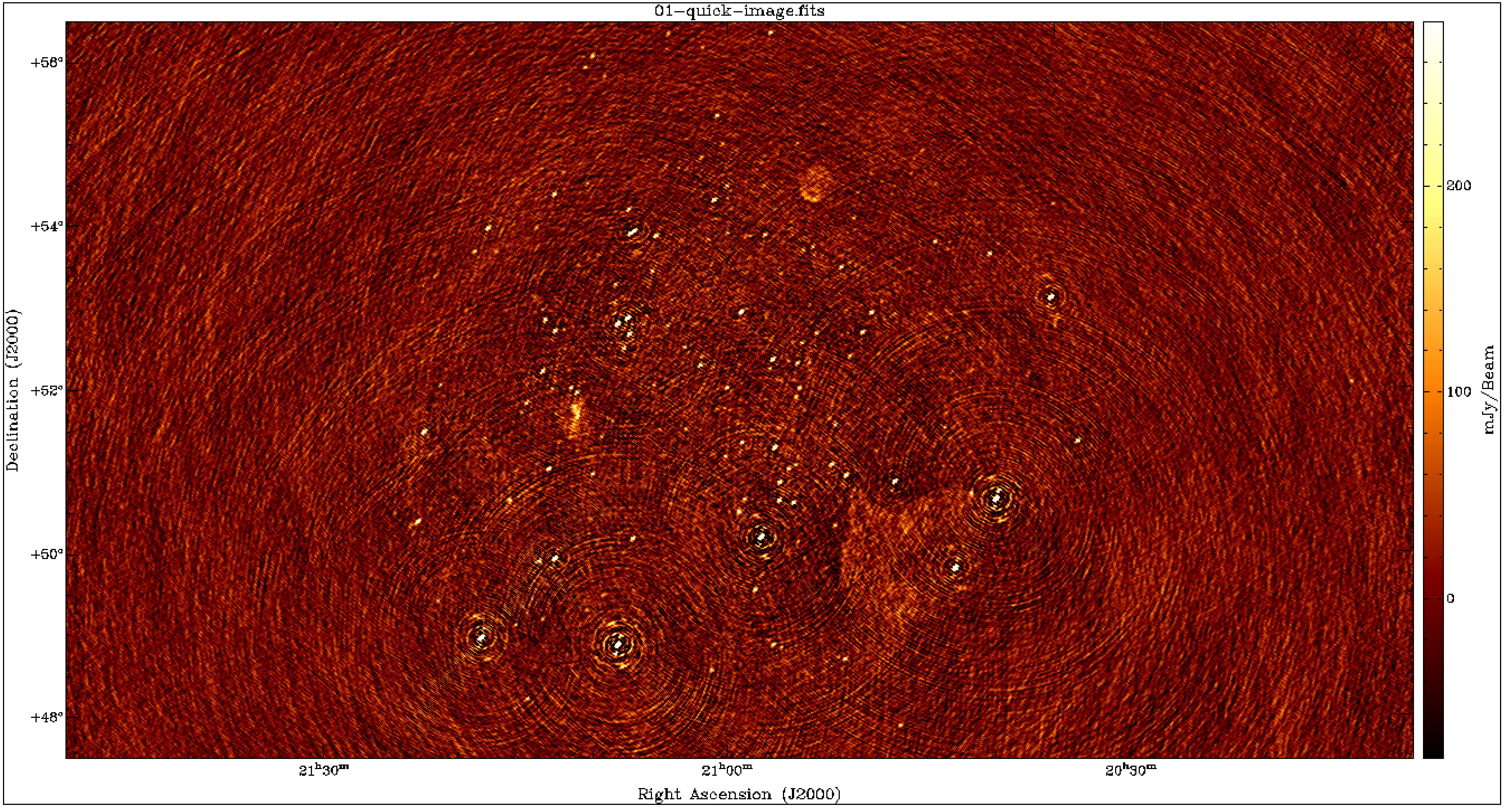
This will output “quick-dirty.fits” and “quick-image.fits”.

Inspect these with your favourite fitsviewer
(e.g., kvis, ds9, casaviewer).

For kvis:

```
$ use Karma
$ kvis quick-*.fits
```

Quick imaging result



Cleaning

The main parameters for cleaning are:

-niter <count> Turns cleaning on and sets max iterations. Normally, cleaning should end at the the threshold, not at the max iterations.

-mgain <gain> How much flux of the peak is subtracted before a major iteration is restarted. Depends on how good your beam is. 0.8 is safe, 0.9 almost always works and is faster.

-threshold <flux> Set the apparent flux (in Jy) at which to stop. Should typically be 3 x sigma.

Run the following command: (still on a single subband)

```
$ wsclean -size <width> <height> -scale <val>asec  \
    -niter <niter> -mgain 0.8 -threshold <flux>    \
    -name clean L456106_SB010_uv.dppp.MS.flg.ph
```

Cleaning

Example command:

```
$ wsclean -size 1400 1400 -scale 50asec        \
    -niter 50000 -mgain 0.8 -threshold 0.1      \
    -name clean L456106_SB010_uv.dppp.MS.flg.ph
```

- It is convenient to store the above command in a shell script.
- Inspect all output .fits images – can you explain what is what?
- Notice in the output the cleaning process:

Peak flux →

Reached Threshold in ~2000 iters →

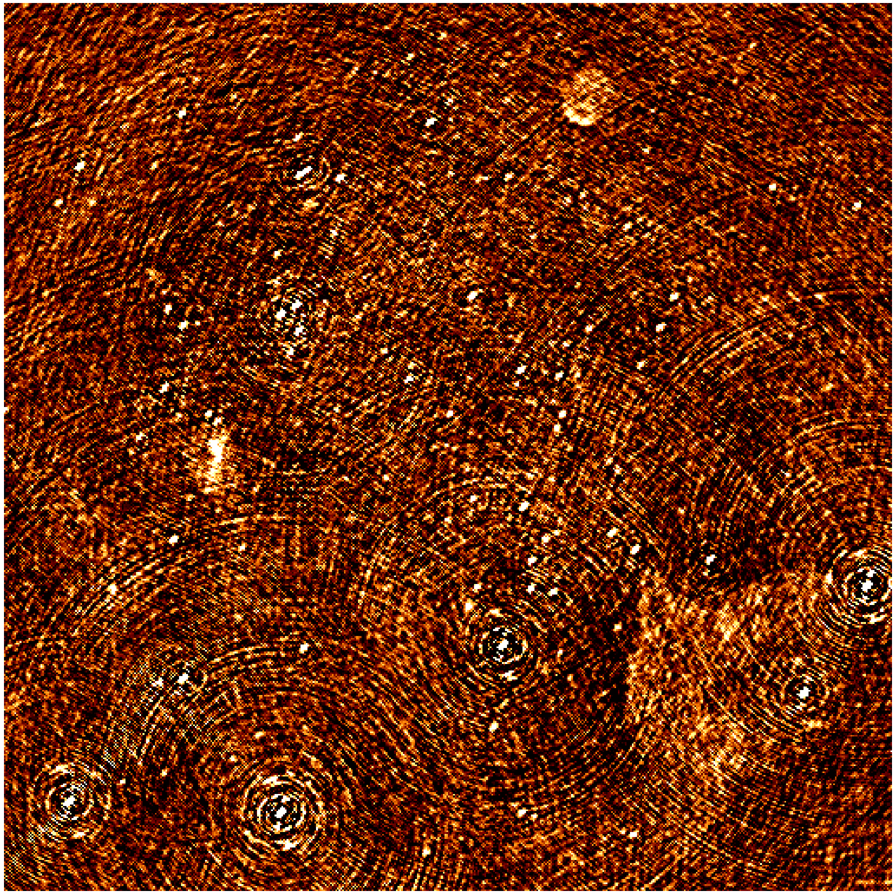
```
 == Cleaning (1) ==
Freed 222 image buffer(s).
Initial peak: 3.2568
Next major iteration at: 0.651359
Iteration 0: (602,465), 3.2568 Jy
[..]
Iteration 100: (731,561), 0.789584 Jy
Stopped on peak 0.646578
[..]
 == Cleaning (2) ==
[..]
Stopped on peak 0.130435
[..]
 == Cleaning (3) ==
Major iteration threshold reached global threshold of 0.1: final major iteration.
Iteration 2000: (545,542), 0.12621 Jy
Stopped on peak -0.0999906
```

Cleaning

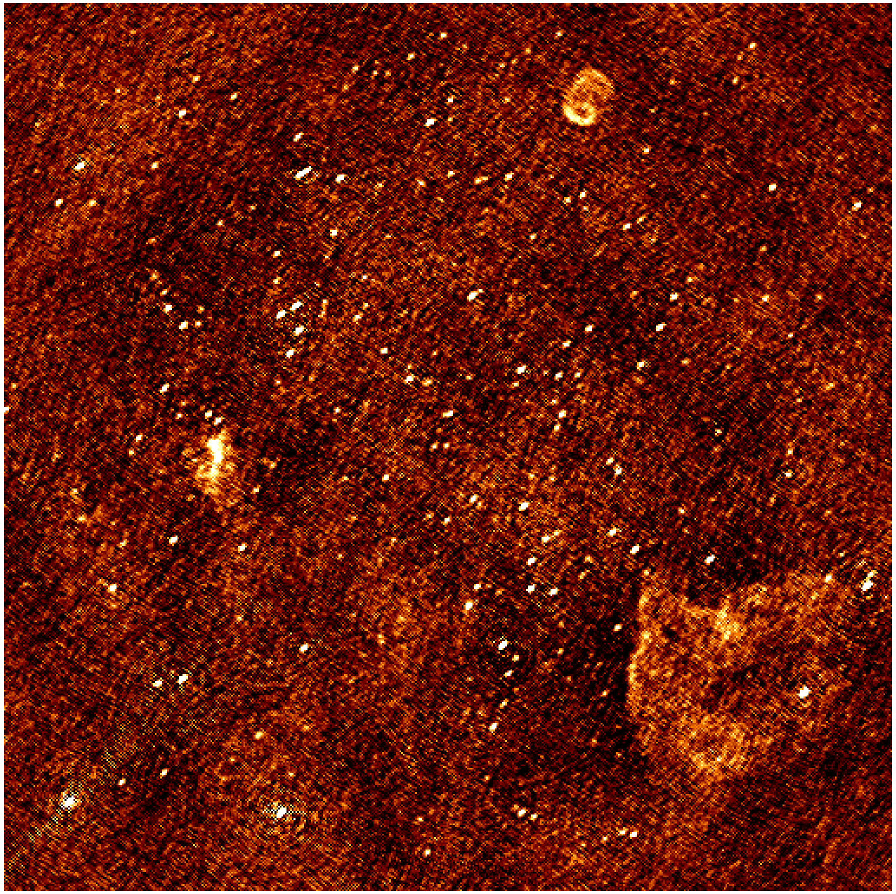
Example command:

```
$ wsclean -size 1400 1400 -scale 50asec          \
    -niter 50000 -mgain 0.8 -threshold 0.1        \
    -name clean L456106_SB010_uv.dppp.MS.flg.ph
```

clean-dirty.fits



clean-image.fits



Apply LOFAR beam

The LOFAR beam is applied by adding
`-apply-primary-beam`

Note that the beam was already applied on the phase centre during calibration (the “applybeam” step in NDPPP). WSClean needs to know this, otherwise **it will use the wrong beam**.

This is specified by also adding
`-use-differential-lofar-beam`

Repeat the previous imaging with the beam, similar to:

```
$ wsclean -size <width> <height> -scale <val>asec     \
    -apply-primary-beam -use-differential-lofar-beam  \
    -niter <niter> -mgain 0.8 -threshold <flux>       \
    -name lofarbeam L456106_SB010_uv.dppp.MS.flg.ph
```

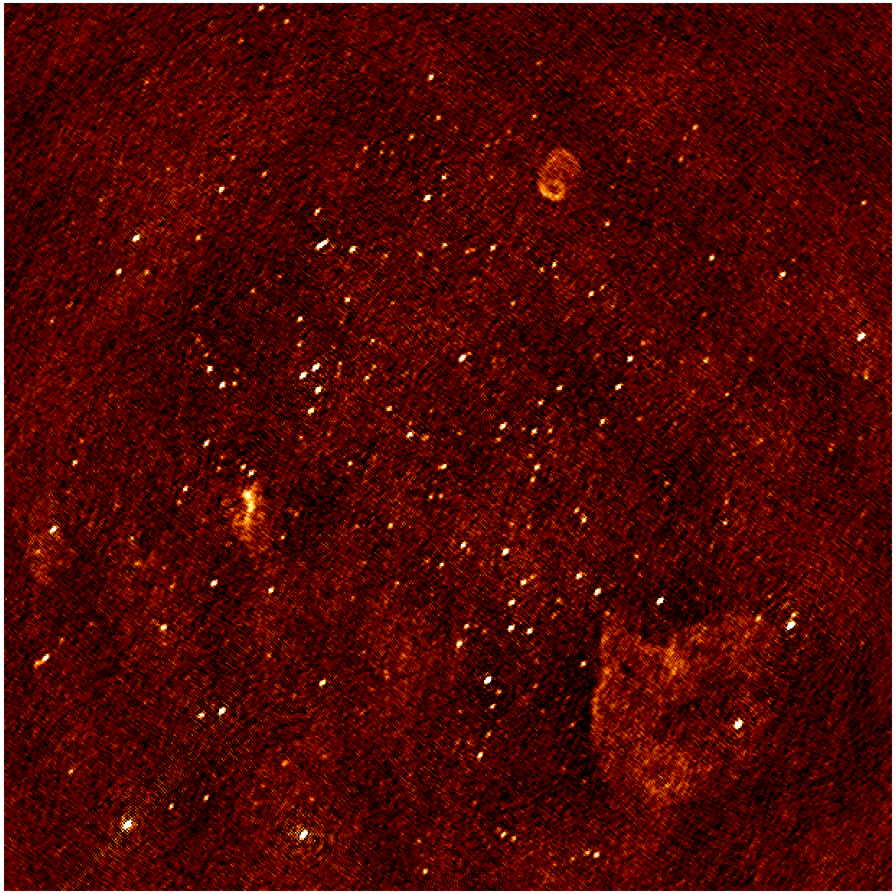
Inspect all the output images.

LOFAR primary beam correction

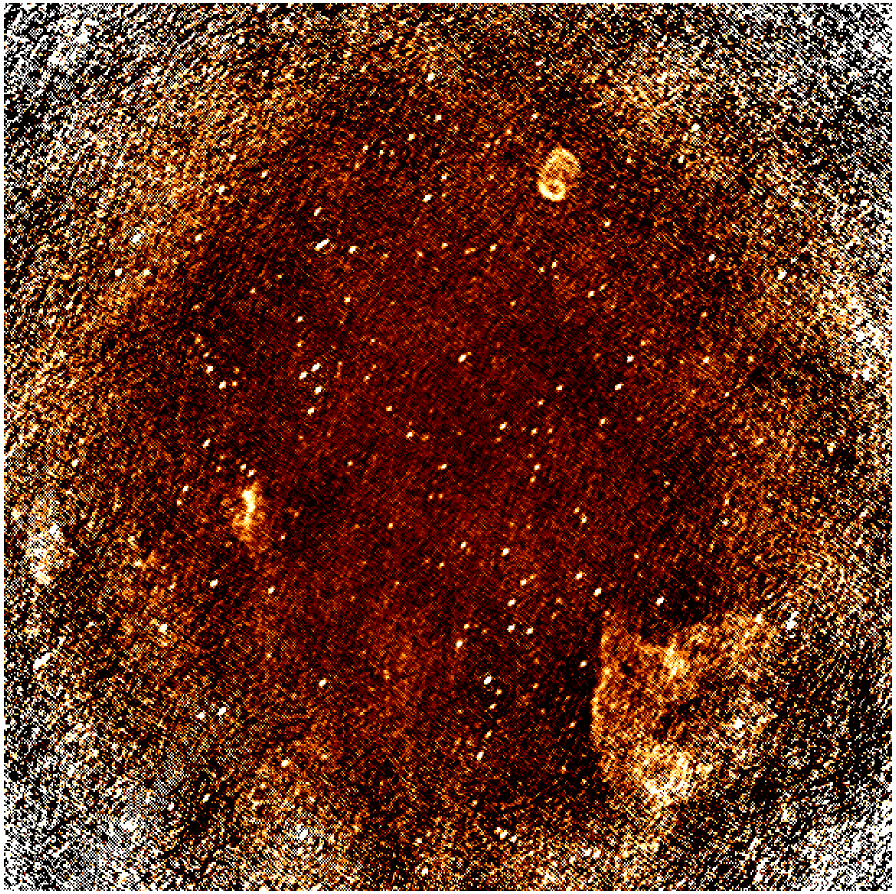
Example command:

```
$ wsclean -size 1400 1400 -scale 50asec                      \
    -apply-primary-beam -use-differential-lofar-beam \
    -niter 50000 -mgain 0.8 -threshold 0.1                  \
    -name clean L456106_SB010_uv.dppp.MS.flg.ph
```

No beam applied:



Differential beam applied:



Weighting and tapers

Read the documentation for `-weight`, `-taper-gaussian` and `-trim`, and optionally other weighting/tapering methods.

Repeat the previous imaging, but with settings for these parameters that are useful to:
- accentuate the diffuse emission; and
- to make the beam Gaussian like, to measure the flux of the emission more easily.

Correct for the primary beam as before.

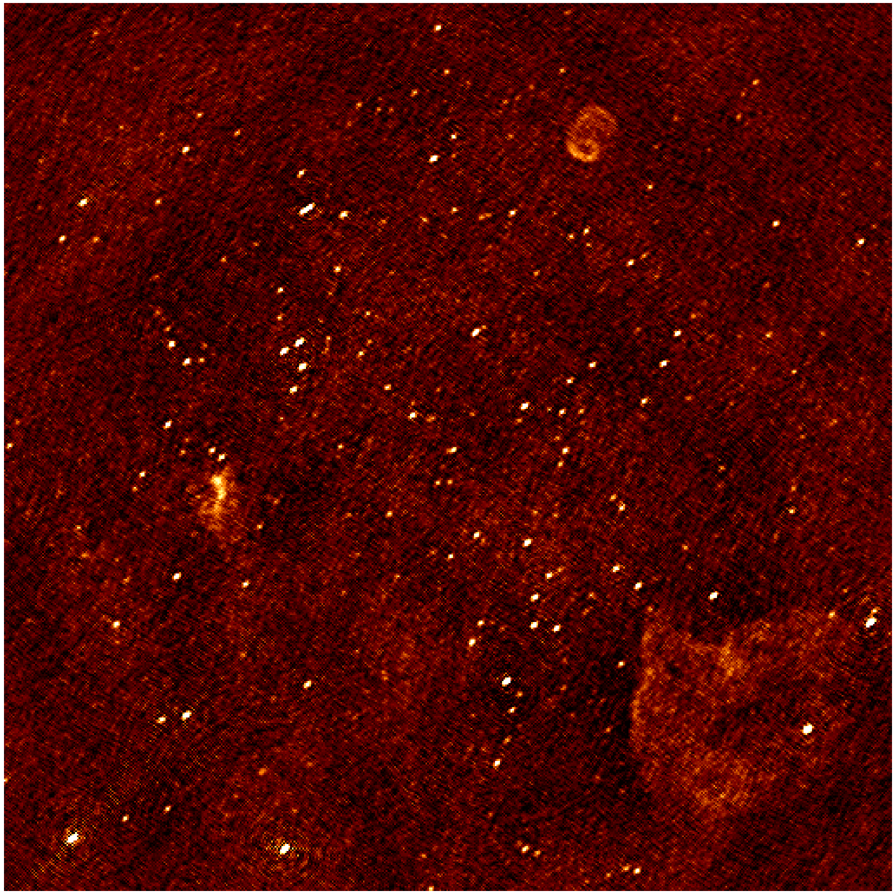
```
$ wsclean -size <width> <height> -scale <val>asec      \
    -trim <trimwidth> <trimheight>                      \
    -apply-primary-beam -use-differential-lofar-beam  \
    -niter <niter> -mgain 0.8 -threshold <flux>         \
    -weight [briggs <robustness> or natural]            \
    -taper-gaussian <val>amin                           \
    -name clean L456106_SB010_uv.dppp.MS.flg.ph
```

Weighting & tapers

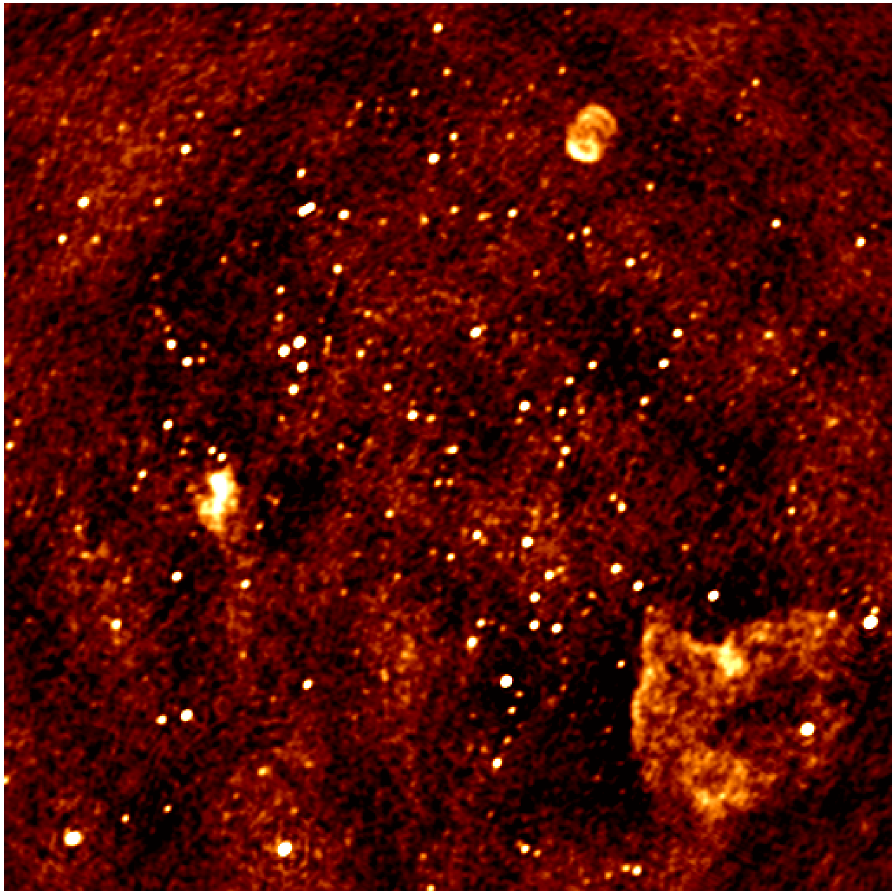
Example command:

```
$ wsclean -size 1800 1800 -scale 50asec                 \
        -trim 1400 1400 -weight briggs 0                \
        -niter 50000 -mgain 0.8 -threshold 0.1          \
        -name weighting L456106_SB010_uv.dppp.MS.flg.ph
```

With -weight briggs 0

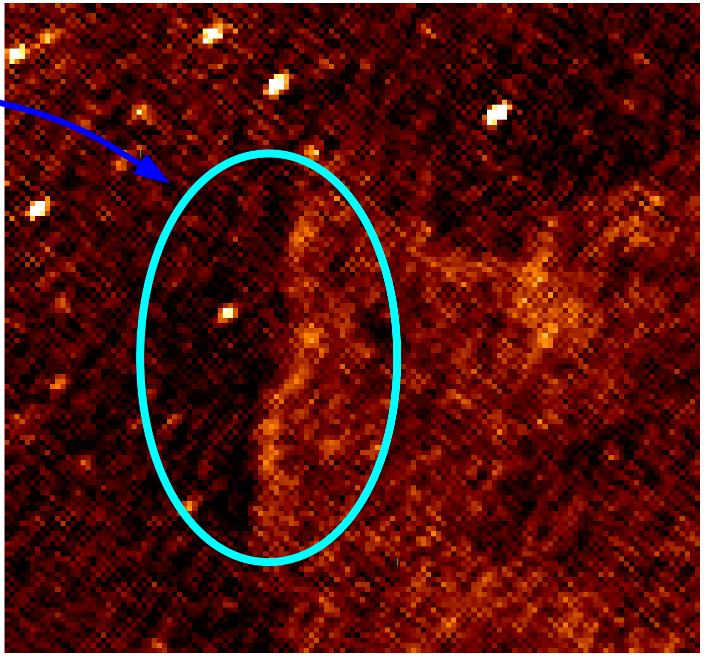


With -weight briggs 0 and -gaussian-taper 2amin



Multi-scale clean

Note the negative areas around the Diffuse sources.
Inspect the “model” image – how did WSClean model the diffuse emission & SNRs?



Repeat the previous imaging, but use multiscale. If you feel adventurous, you can play with `-multiscale-scales` and `-multiscale-scale-bias`. However, for LOFAR this is hardly ever necessary.

```
$ wsclean -size <width> <height> -scale <val>asec     \
    -trim <trimwidth> <trimheight>                    \
    -apply-primary-beam -use-differential-lofar-beam  \
    -niter <niter> -mgain 0.8 -threshold <flux>       \
    -weight [your weighting choice]                   \
    -taper-gaussian <val>amin                         \
    -multiscale                                       \
    -name multiscale L456106_SB010_uv.dppp.MS.flg.ph
```

Baseline-dependent averaging

Note: WSClean version >=1.12a is required for baseline-dependent averaging, not 1.12 or earlier. It might not be available; check your version with `wsclean --version` (maybe we can make it available for this tutorial).

Baseline-dependent averaging lowers the number of visibilities that need to be gridded, which therefore speeds up the imaging.

To enable b.d. averaging, one adds “-baseline-averaging” to the command line with the number of wavelengths (λs) that can be averaged over. Use this rule:

λs = max baseline in λs * 2pi * int. time in s / (24*60*60)

(see https://sourceforge.net/p/wsclean/wiki/BaselineDependentAveraging/ for info)

Rerun the previous imaging with b.d. averaging. Turn beam correction off. WSClean will initially fail with an error – solve the error.

```
$ wsclean -size <width> <height> -scale <val>asec     \
    [..]                                              \
    -baseline-averaging <λs>                          \
    -name bdaveraging L456106_SB010_uv.dppp.MS.flg.ph
```

Baseline-dependent averaging

Example command:

```
$ wsclean -size 1800 1800 -scale 50asec                              \
    -trim 1400 1400 -weight briggs 0                                 \
    -multiscale                                                      \
    -niter 100000 -mgain 0.8 -threshold 0.15                         \
    -baseline-averaging 2.0  -no-update-model-required               \
    -name bdaveraging L456106_SB010_uv.dppp.MS.flg.ph
```

Note in the output:

```
[..]
Averaging factor for longest baseline: 1 x . For the shortest: 775 x
Reordering ../L456106_SB010_uv.dppp.MS.flg.ph into 1 x 1 parts.
Reordering: 0%....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
Baseline averaging reduced the number of rows to 30.8%.
[..]
```

Try a second run with more averaging and inspect the difference between the images. How much averaging is acceptable?

Note: primary beam correction does not yet work with baseline averaging! Turn off primary beam correction.

Multiple output channels & joining

Several approaches for combining all bands (i.e. MSes) :

- Run WSClean on each band and combine images afterwards
  → Only limited cleaning possible.

- Image all MSes in one run with WSClean
  → Clean deep, but assumes flux is constant over frequency.

```
$ wsclean -size <width> <height> -scale <val>asec     \
    [..]                                              \
    -name fullbandwidth *.dppp.MS.flg.ph
```

This takes quite a lot of time. If you have time, you can run the command (but better commands will be presented in the next slides). You can also run it with only a few measurement sets. If you run clean on the full bandwidth, you can *decrease the threshold significantly*, because the system noise will go down by sqrt(29).

- ...

Multiple output channels & joining

Several approaches for combining all bands (i.e. MSes) :

- Run WSClean on each band and combine images afterwards
  → Only limited cleaning possible.

- Image all MSes in one run with WSClean
  → Clean deep, but assumes flux is constant over frequency.

- **Image all MSes and use multi-frequency deconvolution**
  → Cleans deep & incorporates frequency dependency.
  Relevant parameters: -channelsout <count> -joinchannels -fit-spectral-pol <terms> -deconvolution-channels <count>.

```
$ wsclean -size <width> <height> -scale <val>asec        \
     [..]                                                  \
     -channelsout <count> -joinchannels                    \
     -fit-spectral-pol <terms>                             \
     -deconvolution-channels <count>                       \
     -name mfclean *.dppp.MS.flg.ph
```

Decrease the threshold to an appropriate level.

Multiple output channels & joining

Example command using multi-frequency deconvolution:

```
$ wsclean -size 1800 1800 -scale 50asec                         \
    -apply-primary-beam -use-differential-lofar-beam            \
    -trim 1400 1400 -weight briggs 0                            \
    -multiscale                                                 \
    -niter 100000 -mgain 0.8 -threshold 0.15                    \
    -channelsout 14 -joinchannels -fit-spectral-pol 2           \
    -deconvolution-channels 4                                   \
    -name mfclean *.dppp.MS.flg.ph
```

This takes ~two hours (or 1h with baseline averaging).

Analyse the individual output images and the MFS images.

Run source detection

The PyBDSM source detector can be used for self-calibration or cataloguing:

```
 $ use LofIm
 $ pybdsm
PyBDSM version 1.8.7 (LOFAR revision 34639)
==============================================================
PyBDSM commands
  inp task ............ : Set current task and list parameters
  [....]

BDSM [1]:
```

Detect sources in your best output image:

```
BDSM [1]: inp process_image
...
BDSM [2]: filename="mfclean-MFS-image.fits"
BDSM [3]: interactive=True
BDSM [4]: output_opts=True
BDSM [5]: inp
...
BDSM [6]: go
```