LOFAR Imaging tutorial using WSClean

André Offringa

- We start from calibrated data (see previous tutorial)

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

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

- Challenge:

  Make the best looking (/ scientifically useful) image!

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

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

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

```
$ module load wsclean
```

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 observation.ms
```

(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:

- Either image ~up the beam FWHM to capture the entire field **OR** zoom in on the source
- Note that angularwidth ≈ **width** x **scale**.
- Our source is about 10’ in size.

More notes:

- 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 1200 1200 -scale 20asec  \
    -name quick observation.ms
```

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

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

For kvis:

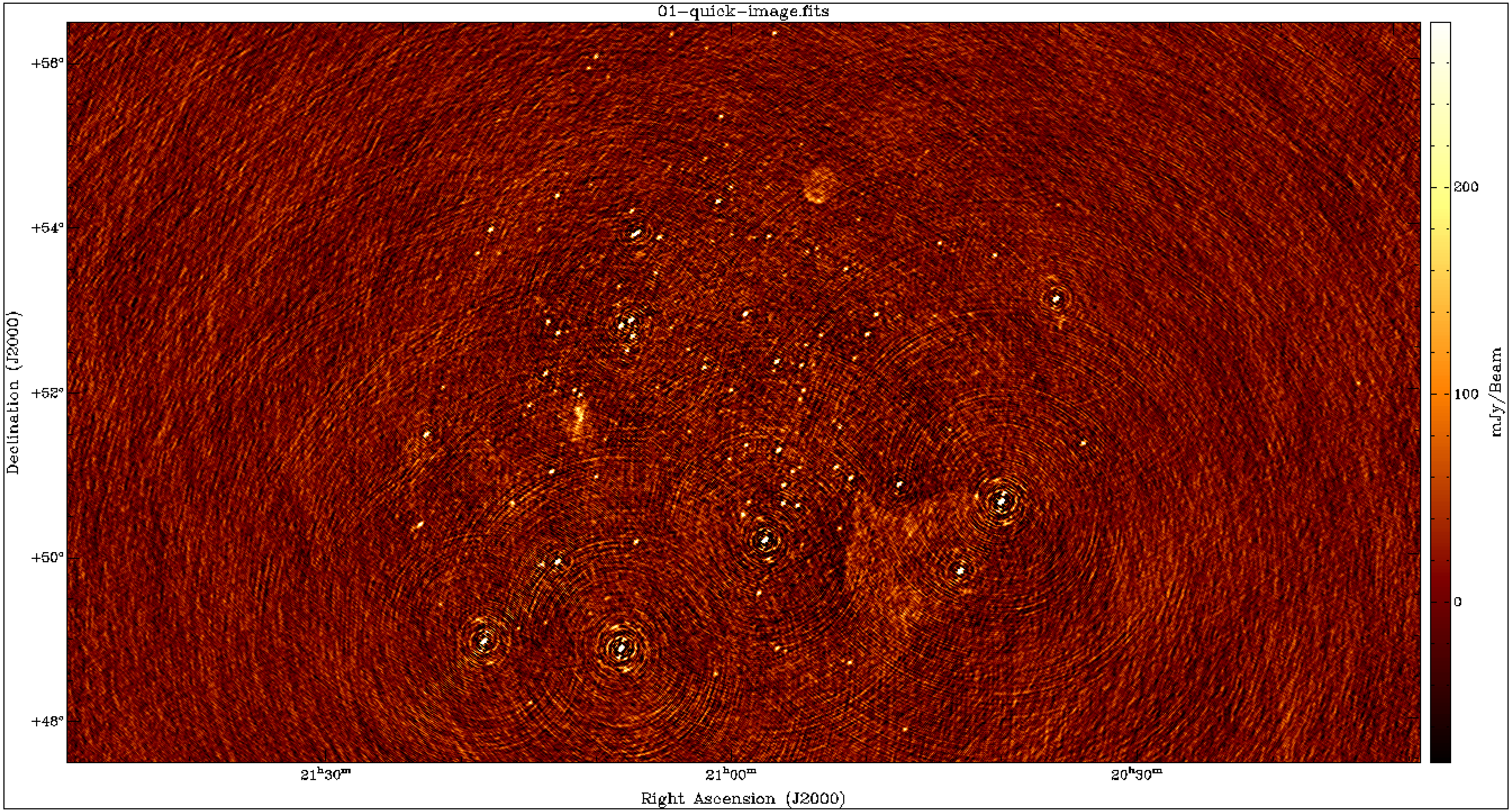
```
$ module load karma
$ kvis quick-*.fits
```

For ds9:

```
$ module load ds9
$ ds9 quick-*.fits
```

If you’re early, also try decreasing the scale (to “zoom in”)

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 the PSF is. 0.8 is safe, 0.9 almost always works and is faster.
-auto-threshold <sigma>	Set the apparent flux at which to stop. This is relative to the computed noise. Should typically be 3-5.

Run the following command:

```
$ wsclean -size <width> <height> -scale <val>asec      \
    -niter <niter> -mgain 0.8 -auto-threshold <level> \
    -name clean observation.ms
```

Cleaning

Example command:

```
 $ wsclean -size 1200 1200 -scale 20asec          \
     -niter 50000 -mgain 0.8 -auto-threshold 5    \
     -name clean observation.ms
```

- 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

Image noise

Reached Threshold in ~9000 iters

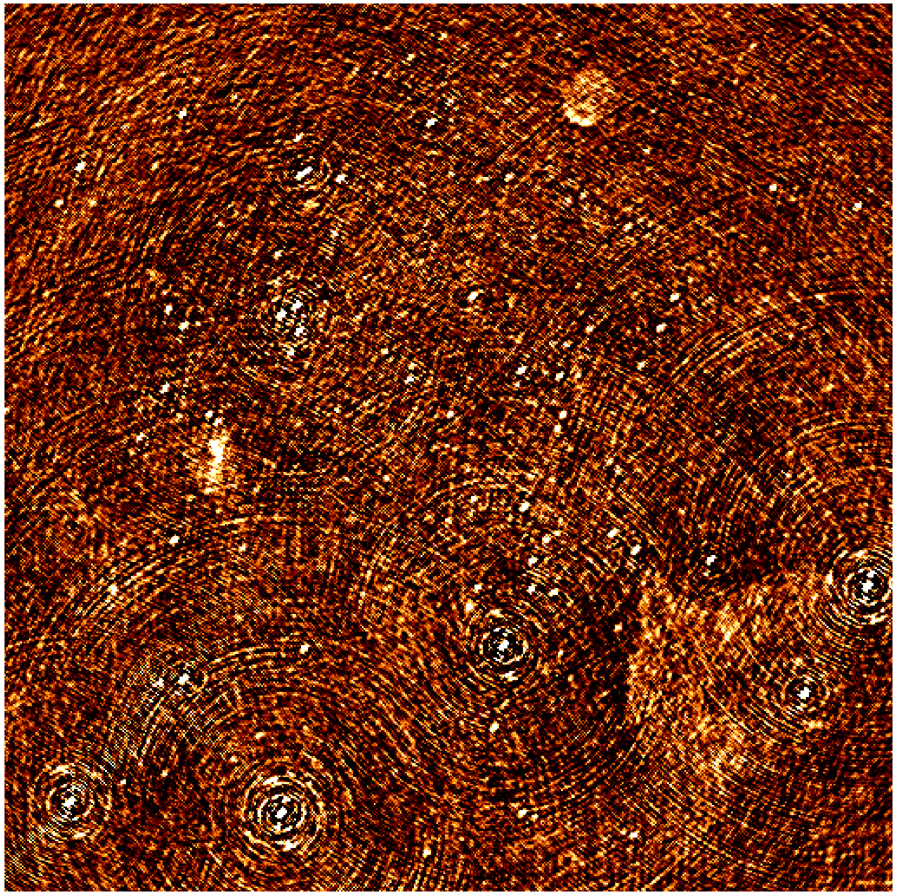
```
 == Cleaning (1) ==
Estimated standard deviation of background noise: 15.64 mJy
Initial peak: 2.39 Jy at 777,717
Next major iteration at: 477.06 mJy
Performed 398 iterations in total, 398 in this major iteration with Clark opt...
Stopped on peak 476.43 mJy, because the minor-loop threshold was reached...
[..]
 == Cleaning (2) ==
Estimated standard deviation of background noise: 12 mJy
Initial peak: -549.48 mJy at 392,632
Next major iteration at: 109.9 mJy
[..]
 == Cleaning (3) ==
Estimated standard deviation of background noise: 9.06 mJy
Initial peak: -164.94 mJy at 584,613
Major iteration threshold reached global threshold of 45.31 mJy: final major it...
Performed 9036 iterations in total, 5868 in this major iteration with Clark op...
Stopped on peak 45.31 mJy, because the threshold was reached.
```

Cleaning

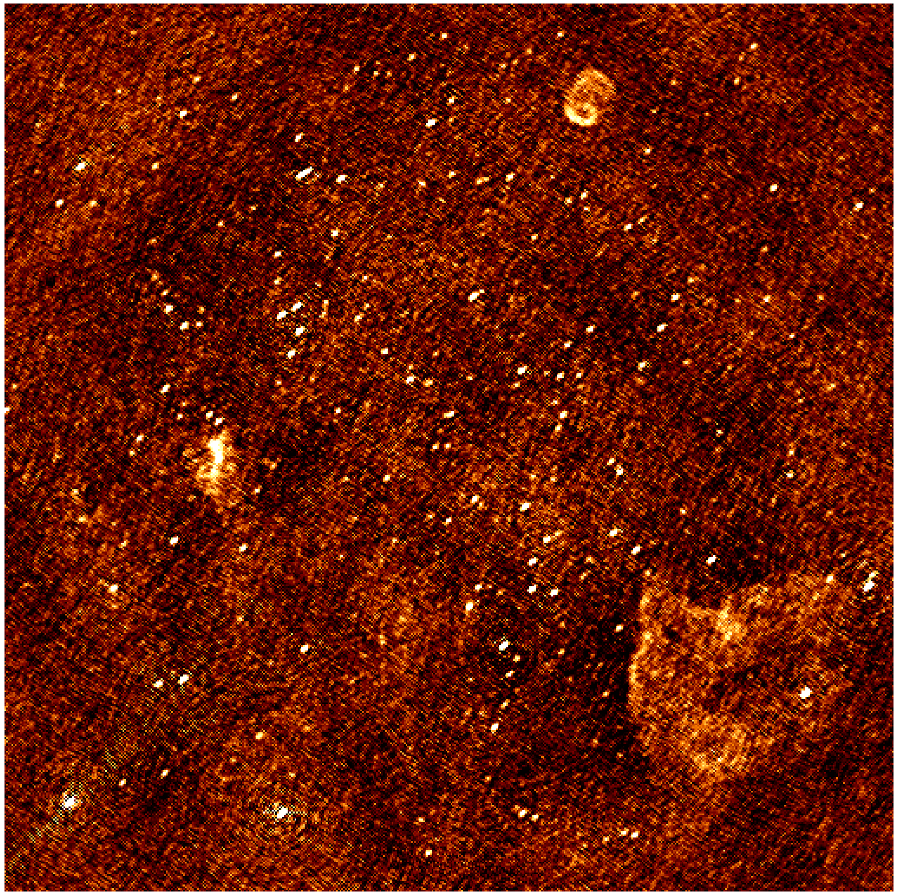
Example command:

```
 $ wsclean -size 1200 1200 -scale 20asec                \
     -niter 50000 -mgain 0.8 -auto-threshold 5   \
     -name clean observation.ms
```

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 -auto-threshold <level> \
    -name lofarbeam observation.ms
```

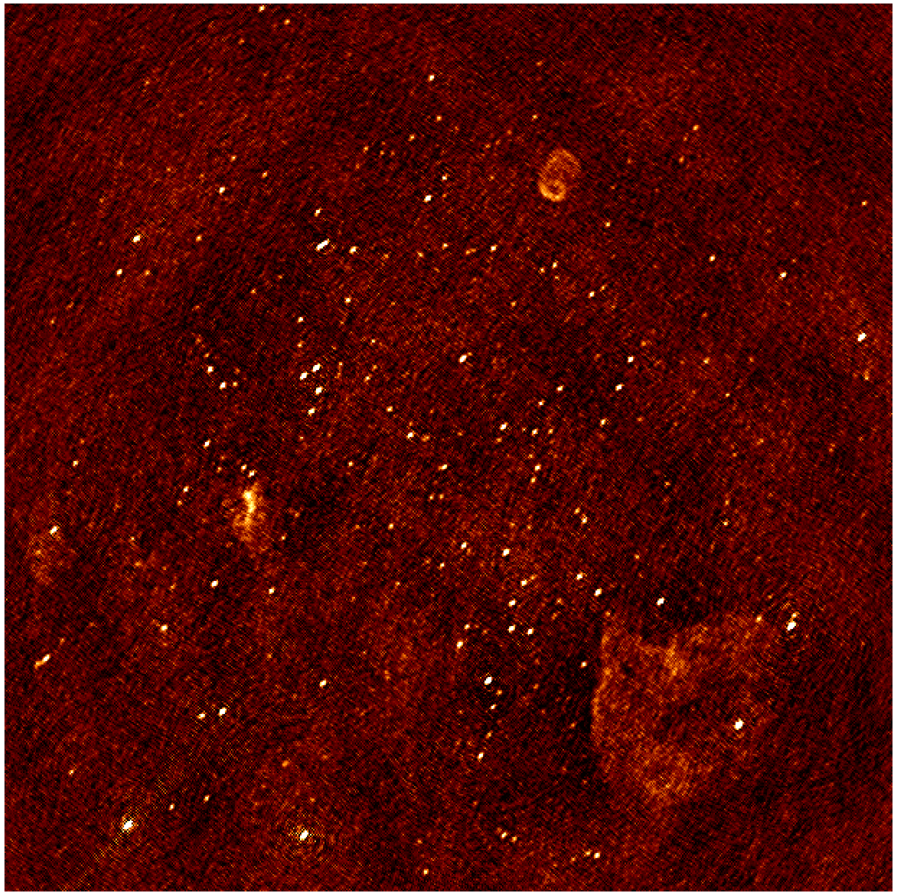
Inspect all the output images.

LOFAR primary beam correction

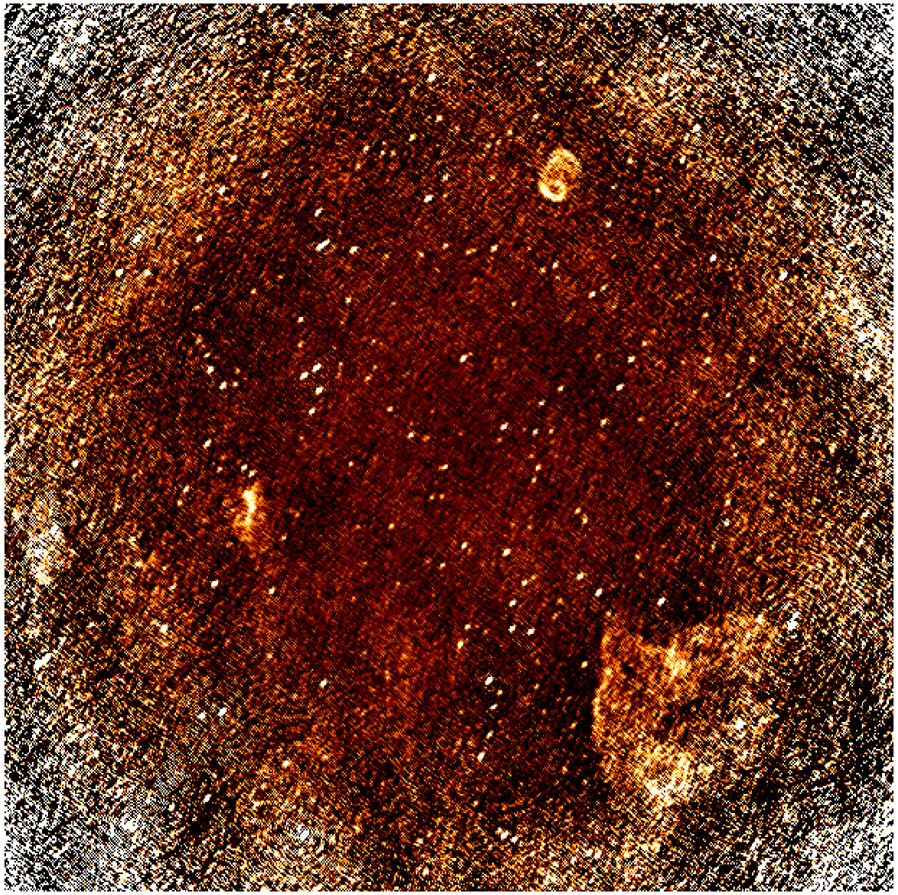
Example command:

```
$ wsclean -size 1200 1200 -scale 20asec                       \
    -apply-primary-beam -use-differential-lofar-beam          \
    -niter 50000 -mgain 0.8 -threshold 0.1                    \
    -name clean observation.ms
```

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
(hint: use '`-reuse-primary-beam`')

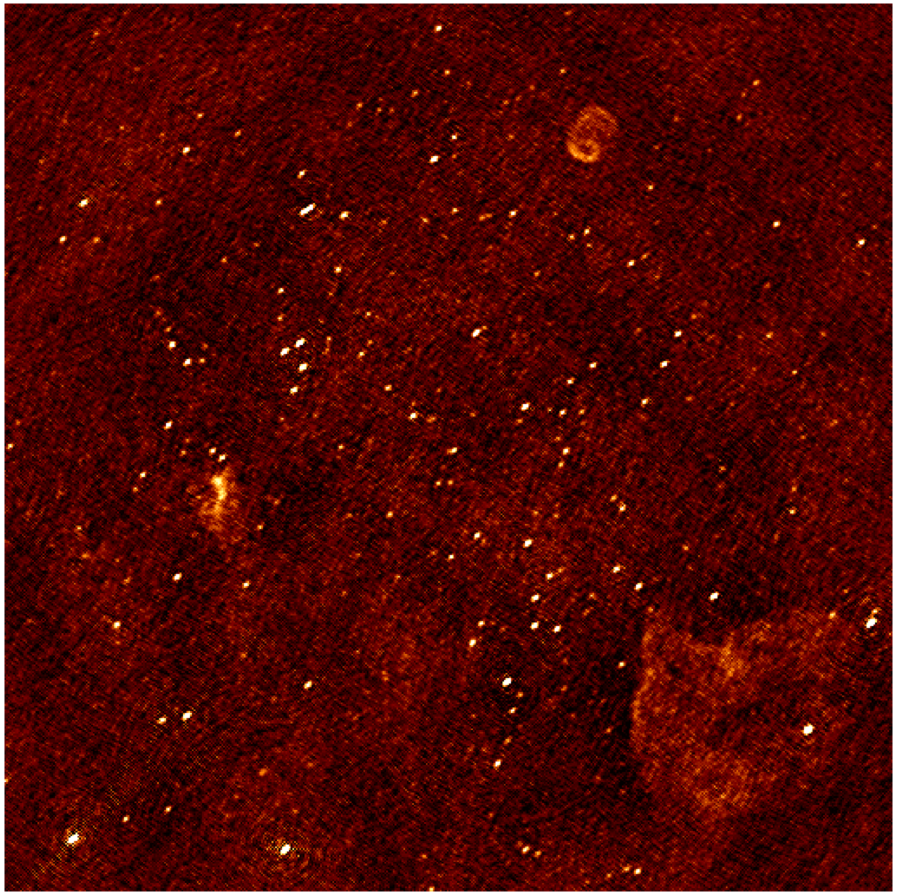
```
$ wsclean -size <width> <height> -scale <val>asec      \
    -reuse-primary-beam                                \
    -apply-primary-beam -use-differential-lofar-beam   \
    -niter <niter> -mgain 0.8 -auto-threshold <flux>   \
    -weight [briggs <robustness> or natural]           \
    -taper-gaussian <val>asec                          \
    -name clean observation.ms
```

Weighting & tapers

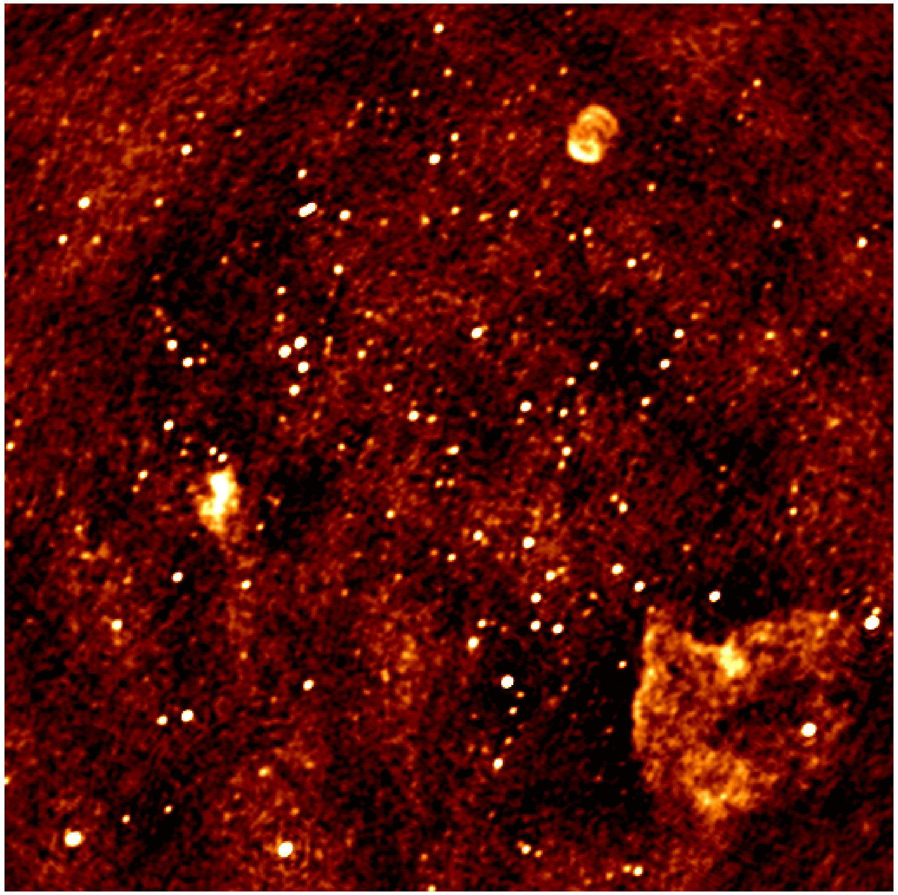
Example command:

```
 $ wsclean -size 1800 1800 -scale 20asec              \
        -trim 1200 1200 -weight briggs 0              \
        -niter 50000 -mgain 0.8 -auto-threshold 0.1 \
        -name weighting observation.ms
```

With -weight briggs 0



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

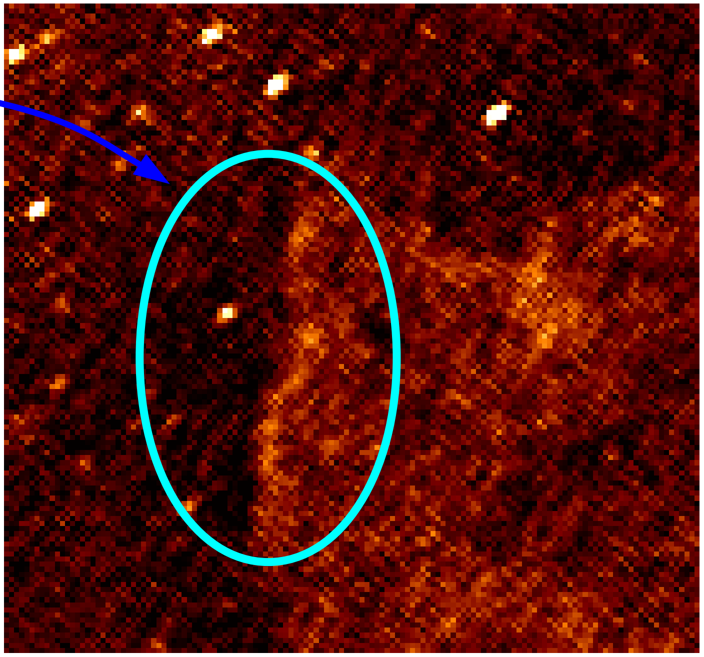


Multi-scale clean

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 cleaning.

Compare these images with the previous run.



```
$ wsclean -size <width> <height> -scale <val>asec          \
    -apply-primary-beam -use-differential-lofar-beam       \
    -niter <niter> -mgain 0.8                              \
    -weight [your weighting choice]                        \
    -taper-gaussian <val>asec                              \
    -multiscale -auto-mask 5 -auto-threshold 1             \
    -name multiscale observation.ms
```

Baseline-dependent averaging

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. WSClean will initially fail with an error – solve the error.

```
$ wsclean -size <width> <height> -scale <val>asec     \
    [..]                                              \
    -baseline-averaging <λs>                          \
    -name bdaveraging observation.ms
```

Baseline-dependent averaging

Example command:

```
$ wsclean -size 1200 1200 -scale 20asec                         \
    -trim 1200 1200 -weight briggs 0                            \
    -multiscale -auto-mask 5                                    \
    -niter 100000 -mgain 0.8 -auto-threshold 1                  \
    -baseline-averaging 2.0  -no-update-model-required          \
    -name bdaveraging observation.ms
```

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?

Multiple output channels & joining

Several approaches for combining multiple subbands (MSes) :

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

- 2. 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 *.ms
```

- ...

Multiple output channels & joining

Several approaches for combining multiple subbands (MSes) :

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

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

- **3. Image all MSes and use multi-frequency deconvolution**
  → Cleans deep & incorporates frequency dependency.
  Relevant parameters: -channels-out <count> -join-channels -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 *.ms
```

Multiple output channels & joining

Example command using multi-frequency deconvolution:

```
$ wsclean -size 1200 1200 -scale 20asec                              \
    -apply-primary-beam -use-differential-lofar-beam                 \
    -weight briggs 0                                                 \
    -multiscale -auto-mask 5                                         \
    -niter 100000 -mgain 0.8 -auto-threshold 1                       \
    -channels-out 4 -join-channels -fit-spectral-pol 2               \
    -name mfclean *.ms
```

In this tutorial, you have only one measurement set with a limited number of channels, but we can still perform MF imaging on the 4 channels in the measurement set.

After starting a MF run, get some coffee :)
(it takes significantly longer...)

Analyse the individual output images and the MFS images.

Run source detection

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

```
 $ module load pybdsf
 $ pybdsf
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
```

Do lots of tweaking of WSClean parameters and see what they do! You can also try imaging your calibrated set to see if you’ve imaged it properly.

In the near future...

We are implementing something called “*Image Domain Gridding*” (IDG). This will make the imaging *much* faster.’

It can also correct for ionospheric corrections and other direction-dependent effects.

Because it is still experimental, we have left it out of this tutorial, but you can do all the things you did in this tutorial with IDG by adding `` `-use-idg` `` to the wsclean command line.