SCRIPTS AVAILABLE ON: https://github.com/hampusosterbom/An-analysis-of-upgrade-feasibilities-for-ALMA-scripts.git

NOTE: Scripts are not perfect, if you run into any issues or have any questions feel free to send an email to `hampusosterbom1@gmail.com`

Configuration files can be found here: https://almascience.nrao.edu/tools/casa-simulator

---

kml_to_cfg.py

Script to read markers from a .kml file and add them to a configuration with correct elevation

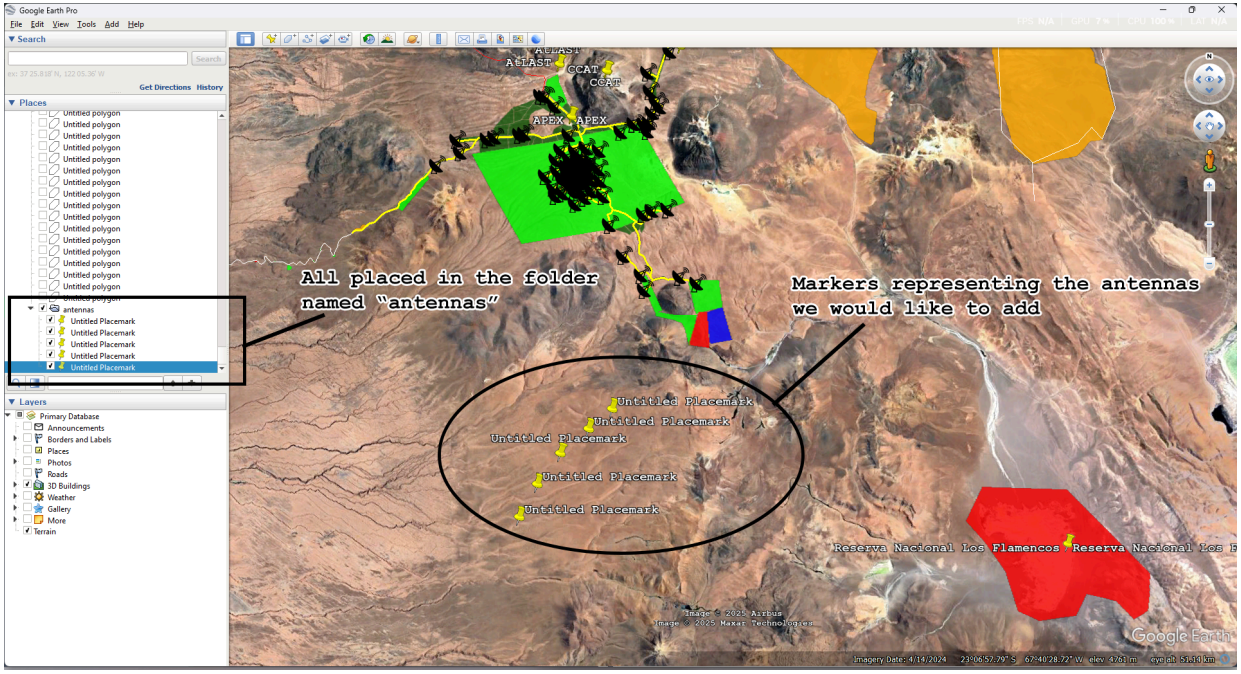
- The scripts uses "open-elevations" API to get correct elevation values given coordinates (which are read from the markers from the .kml-file).
- It first gets ALMAs reference point elevation ($\emph{lat}_0 = -23.02271113^\circ$, $\emph{lon}_0 = -67.75436287^\circ$, OSO monument, i.e. MASTER0) in order to convert the ABSOLUTE elevation, retrieved from the API, to the relative elevation compared to the reference point (as in standard ALMA-configurations). PS: we use the Chalmers master ant. (MASTER0) as the reference point, by using these coordinates I found the scripts and positions line up correctly / most accurately when going between cfg-files and kml-files, see https://legacy.nrao.edu/alma/site/Chajnantor/maps/coordinates.html
- It then adds these new XYZ coordinates as antennas to the base-cfg provided and writes out a combined configuration.

Example: we want to add 5 antennas to C43-10:

- Start by placing markers in Google earth where you would like to add antennas
- Make sure you place all the markers (antennas) in a folder
- Save the file as a .kml

Packages needed:

`pandas`, `geopandas`, `numpy`, `requests`, `pyproj`, `fiona`



Import generate_cfg_from_kml and setup:

In [37]:
```
# Be sure to save kml_to_cfg.py in a directory/folder then you can either make a script in that folder and do
# the following or run it with "python kml_to_cfg.py --base_cfg --kml_path --folder_name --out_cfg"
from kml_to_cfg import generate_cfg_from_kml

generate_cfg_from_kml(
    base_cfg_path="alma.cycle11.10.cfg",        # Path to the cfg you want to add pads to
    kml_path="5pads.kml",                       # Path to the kml file with the markers
    folder_name="antennas",                     # Name of the folder with the markers
    output_path="C43-10p5.cfg"                  # Output path of combined cfg
)
```

```
2025-08-10 20:28:12,956 - INFO - No existing NP pads; starting at NP001
2025-08-10 20:28:13,406 - INFO - Reference center elevation: 5032.0 m
2025-08-10 20:28:13,580 - INFO - Adding 5 new NP pads with relative z values
2025-08-10 20:28:13,585 - INFO - Wrote config to C:\Users\hampu\Guide\C43-10p5.cfg
```

Now the 5 antennas we wanted to add is part of the configuration, denoted as NP# (New Pad):

```
1588.616 -13517.848 -384.000 12.0 NP001
321.423 -14580.016 -491.000 12.0 NP002
-1223.183 -15896.421 -655.000 12.0 NP003
-2489.795 -17557.162 -773.000 12.0 NP004
-3485.325 -19329.107 -931.000 12.0 NP005
```

---

cfg_to_kml.py

Script to convert antenna positions from a cfg-file into markers in a kml-file:

- Like 'kml_to_cfg.py' but reversed, here we can take any cfg-file and convert it to a .kml file with the antennas as markers.

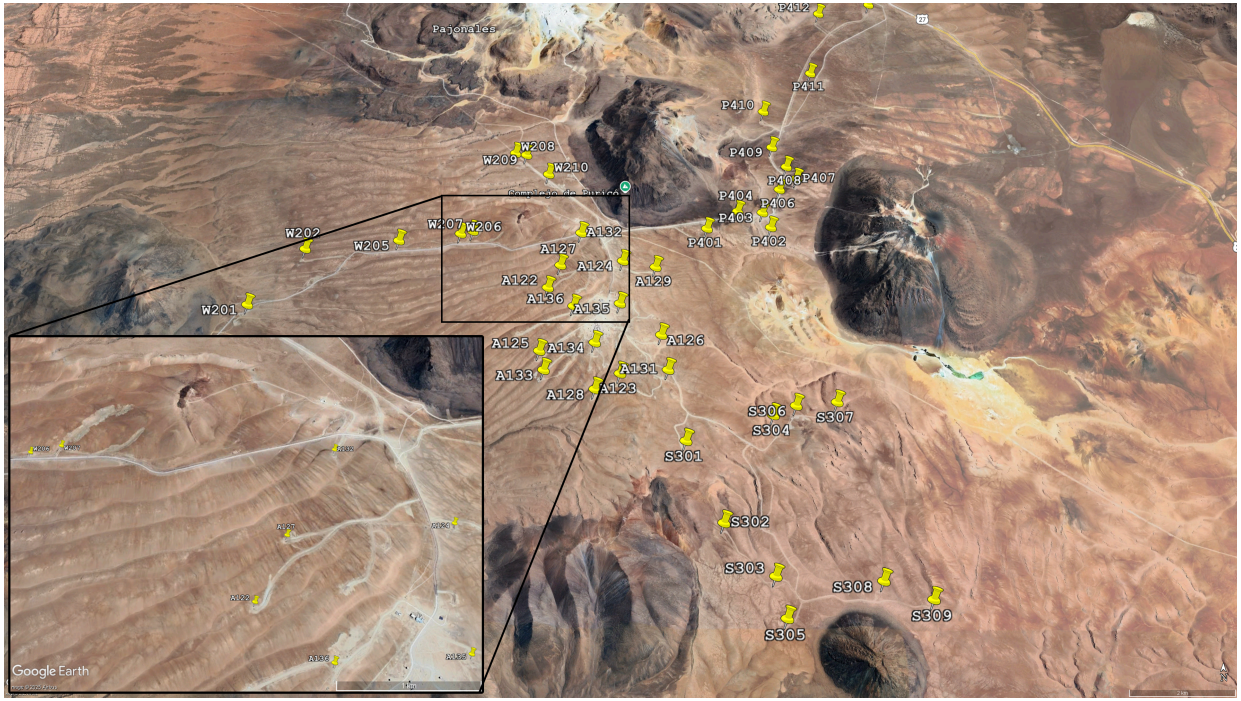
Packages needed:

`pandas`, `pyproj`, `simplekml`

In [39]:
```
from cfg_to_kml import generate_kml_from_cfg

generate_kml_from_cfg("alma.cycle11.10.cfg", "example.kml")
print("Example KML file generated: example.kml")
```

```
2025-08-10 20:28:21,032 - INFO - Saved KML to example.kml
Example KML file generated: example.kml
```

Now we succesfully have a kml-file with markers at all antenna positions for alma.cycle11.10:



pad_placer_alma.py

Script that tries to optimize pad placement in given allowed regions (focuses on BLD smoothness):

Packages needed: `pandas` , `numpy` , `shapely` , `scipy` , `pyproj` , `matplotlib` , `geopandas` , `scikit-learn` , `requests`

Key parameters:

- `cfg_path` – existing ALMA `.cfg` (LOC: x y z diam name)
- `kml_path` – KML with allowed regions
- `poly_layers` – layer names inside `kml_path` to use

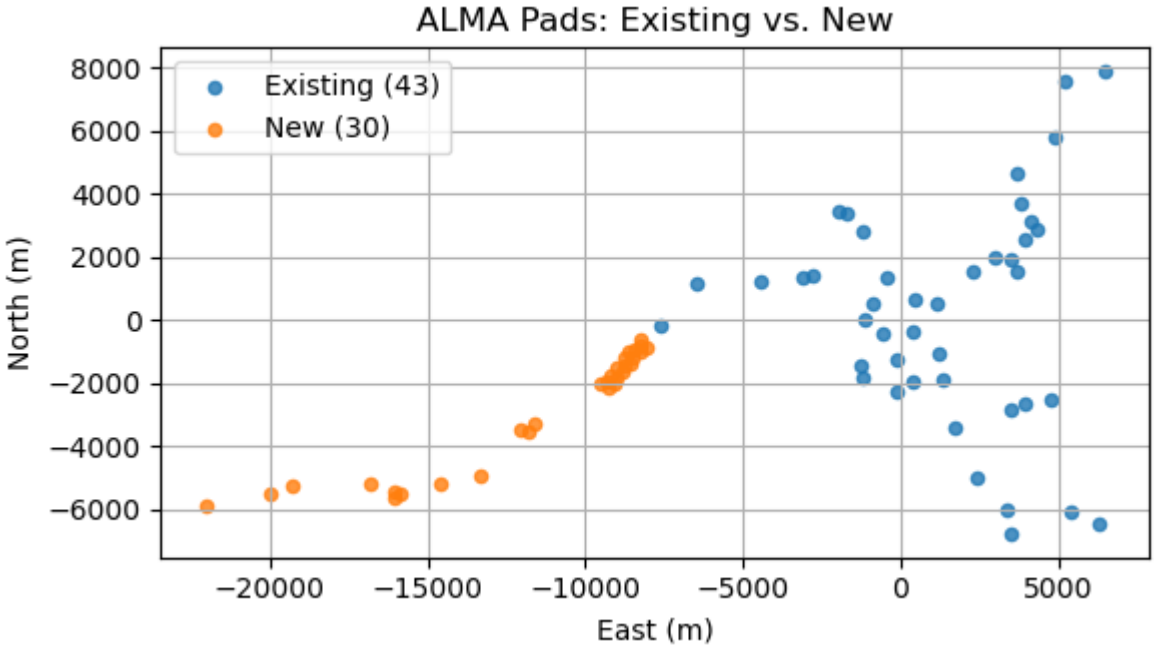
- `regions_kml` – KML of "region pads" (optional reuse set)
- `favored_weight` – weight for reuse candidates vs grid (e.g., 0.75)
- `include_reuse` – allow selecting pads from `regions_kml`
- `n_new` – how many pads to add
- `spacing` – grid spacing for candidate generation
- `min_bl` – minimum baseline to any **existing** pad (m)
- `R_max` – max radius for candidates from array center (m)
- `inner_radius`, `inner_ratio`, `outer_ratio` – define inner/outer split and the desired mix
- `ratio_penalty_weight` – strength of penalty for deviating from that mix (0 = ignore)
- `transition_width` – softens the inner/outer transition around `inner_radius`
- `cable_weight` – cost term that discourages long cable runs (to reference point)
- `density_radius`, `density_weight` – penalize crowding to spread pads out
- `sigma_b`, `fixed_r_max` – shape & extent of the **target baseline distribution** (histogram binning + goal)
- `use_open_elevation` – fetch z from Open-Elevation
- `absolute_z` – if true, write absolute elevation; if false, write **LOC-relative** `z = pad – origin`

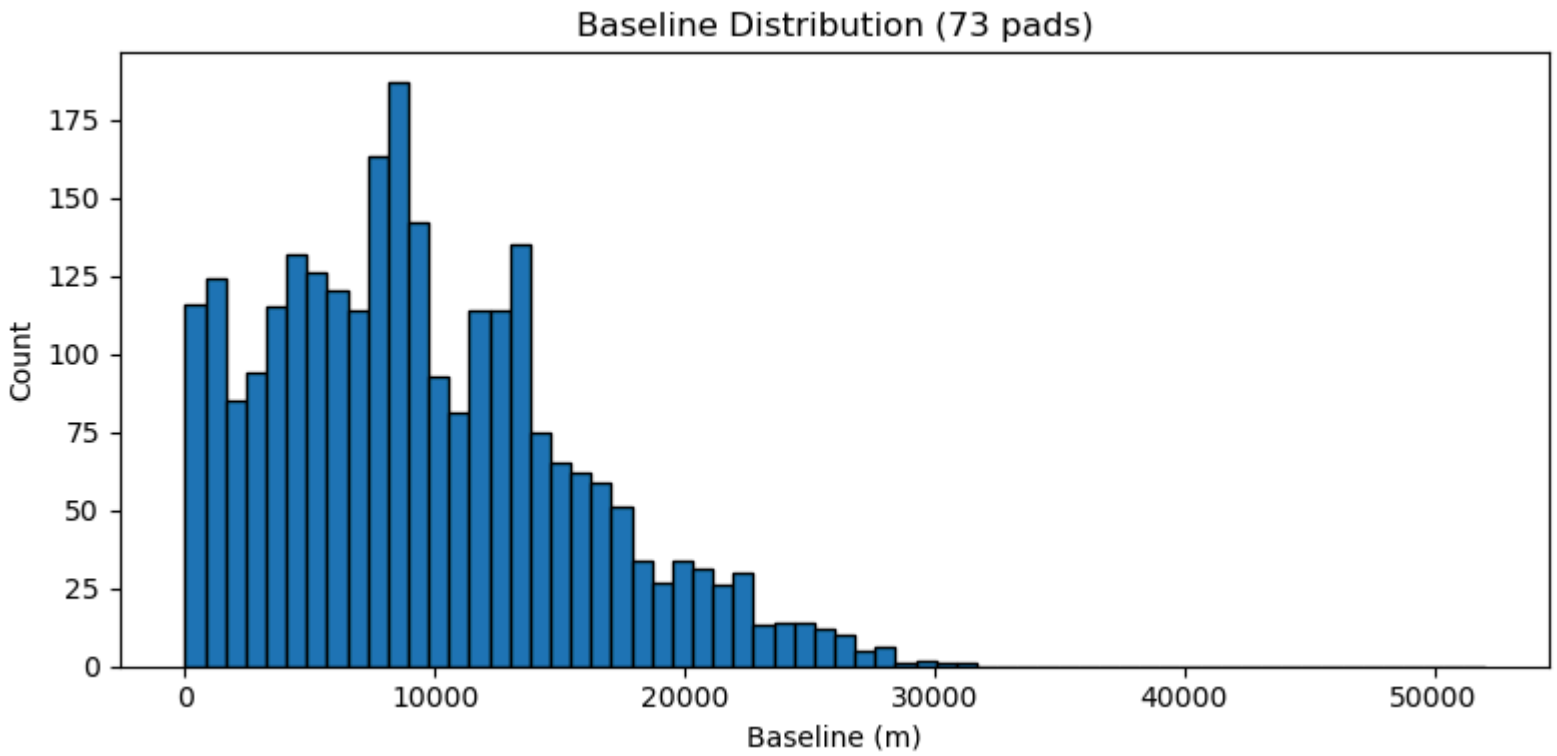
Outputs:

- `out_new` – CFG of **new pads only**
- `out_plus57` – **existing + new** combined CFG
- Figures saved:
  - `pads_scatter.png`
  - `baseline_hist.png`

In [32]:
```
import argparse
from pad_placer_alma import load_data, optimize_pads, add_open_elevation_z, write_outputs, generate_plots

example_args = argparse.Namespace(
    cfg_path="alma.cycle11.10.cfg",
    kml_path="RegionsOSF.kml",
    poly_layers=['Regions OSF'],
    regions_kml="PadsAOS.kml",
    out_new="new_pads_57.cfg",
    out_plus57="alma.cycle11.10.plus_new.cfg",
    n_new=30, spacing=25.0, min_bl=200, R_max=30000, inner_radius=4700,
    favored_weight=0.75, include_reuse=False,
    cable_weight=5e-6, inner_ratio=2, outer_ratio=1,
    fixed_r_max=52000.0, ratio_penalty_weight=0,
    transition_width=6000.0, density_radius=1900.0, density_weight=0.4,
    sigma_b=24000, use_open_elevation=True, absolute_z=False
)

# 2) Load & optimize
existing_df, existing_pts, ex_weights, ctr_start, allowed_area, region_df, crs_loc, lat0, lon0 = load_data(example_args)
new_df, _, cand_df, sel_df, cand_pts, sel, fixed_r_max = optimize_pads(
    example_args, existing_df, existing_pts, ex_weights, ctr_start, allowed_area, region_df
)

# 3) Elevations (z in LOC: pad_elev - origin_elev)
if example_args.use_open_elevation:
    new_df = add_open_elevation_z(new_df, crs_loc, lat0, lon0, relative_to_origin=not example_args.absolute_z)

# 4) Save + plots
write_outputs(example_args, new_df, existing_df)
generate_plots(new_df, existing_df, fixed_r_max)
```

```
2025-08-10 20:10:13,780 - INFO - Loaded 170 region pads
2025-08-10 20:10:18,397 - INFO - Clipped 0 candidates beyond 30000 m
2025-08-10 20:10:18,402 - INFO - Total candidates: 1180 | Region: 0 | Inner: 0
2025-08-10 20:10:18,453 - INFO - Step 1/30: added 16, cost=4797.3
2025-08-10 20:10:18,478 - INFO - Step 2/30: added 93, cost=5020.7
2025-08-10 20:10:18,503 - INFO - Step 3/30: added 5, cost=5313.3
2025-08-10 20:10:18,526 - INFO - Step 4/30: added 128, cost=5681.2
2025-08-10 20:10:18,550 - INFO - Step 5/30: added 105, cost=6215.9
2025-08-10 20:10:18,573 - INFO - Step 6/30: added 235, cost=6916.0
2025-08-10 20:10:18,594 - INFO - Step 7/30: added 8, cost=7598.1
2025-08-10 20:10:18,617 - INFO - Step 8/30: added 1179, cost=8487.7
2025-08-10 20:10:18,640 - INFO - Step 9/30: added 21, cost=9071.3
2025-08-10 20:10:18,662 - INFO - Step 10/30: added 567, cost=10134.4
2025-08-10 20:10:18,686 - INFO - Step 11/30: added 18, cost=11027.0
2025-08-10 20:10:18,708 - INFO - Step 12/30: added 1088, cost=12280.2
2025-08-10 20:10:18,730 - INFO - Step 13/30: added 171, cost=13361.6
2025-08-10 20:10:18,755 - INFO - Step 14/30: added 374, cost=14874.9
2025-08-10 20:10:18,778 - INFO - Step 15/30: added 1129, cost=16243.5
2025-08-10 20:10:18,802 - INFO - Step 16/30: added 39, cost=17765.7
2025-08-10 20:10:18,838 - INFO - Step 17/30: added 546, cost=19762.2
2025-08-10 20:10:18,868 - INFO - Step 18/30: added 1096, cost=21578.7
2025-08-10 20:10:18,891 - INFO - Step 19/30: added 94, cost=23577.2
2025-08-10 20:10:18,917 - INFO - Step 20/30: added 421, cost=26092.2
2025-08-10 20:10:18,939 - INFO - Step 21/30: added 845, cost=28479.9
2025-08-10 20:10:18,964 - INFO - Step 22/30: added 480, cost=31100.5
2025-08-10 20:10:18,987 - INFO - Step 23/30: added 943, cost=33911.7
2025-08-10 20:10:19,009 - INFO - Step 24/30: added 719, cost=37003.2
2025-08-10 20:10:19,032 - INFO - Step 25/30: added 884, cost=40349.2
2025-08-10 20:10:19,056 - INFO - Step 26/30: added 728, cost=43947.6
2025-08-10 20:10:19,081 - INFO - Step 27/30: added 692, cost=47806.1
2025-08-10 20:10:19,107 - INFO - Step 28/30: added 970, cost=51926.6
2025-08-10 20:10:19,131 - INFO - Step 29/30: added 350, cost=56300.8
2025-08-10 20:10:19,157 - INFO - Step 30/30: added 243, cost=61061.9
2025-08-10 20:10:19,160 - INFO - No region pads reused.
2025-08-10 20:10:19,502 - INFO - Wrote new pads to new_pads_57.cfg
2025-08-10 20:10:19,506 - INFO - Wrote combined config to alma.cycle11.10.plus_new.cfg
```


ALMA Pads: Existing vs. New
Existing (43)
New (30)
North (m)
East (m)
8000
6000
4000
2000
0
−2000
−4000
−6000
−20000
−15000
−10000
−5000
0
5000

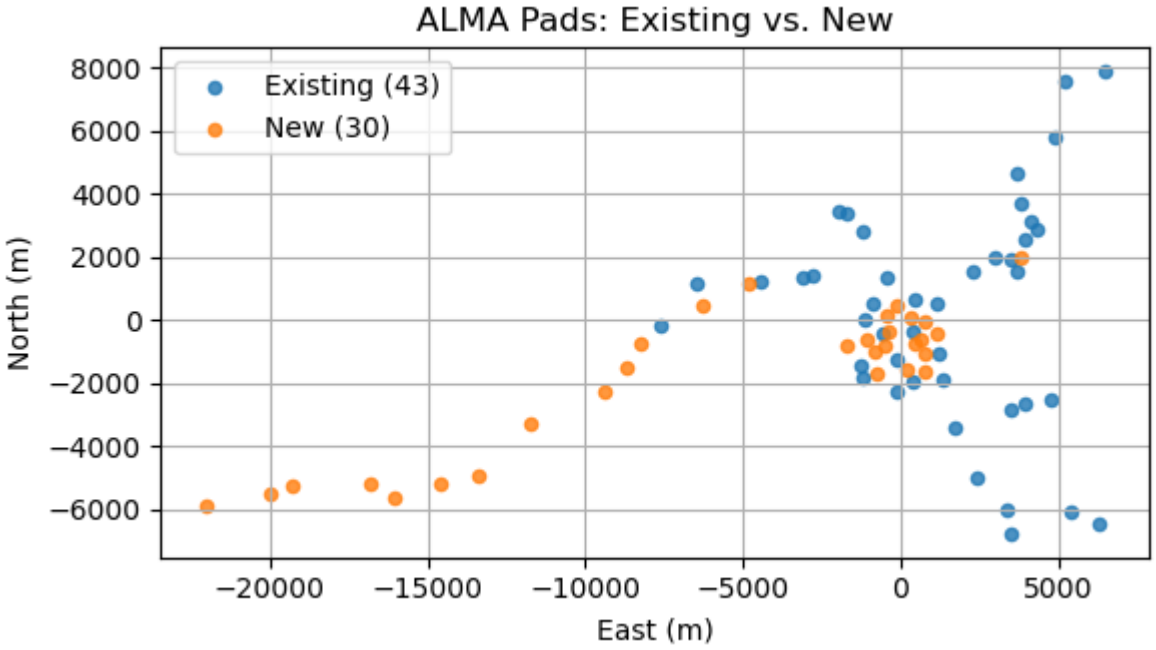


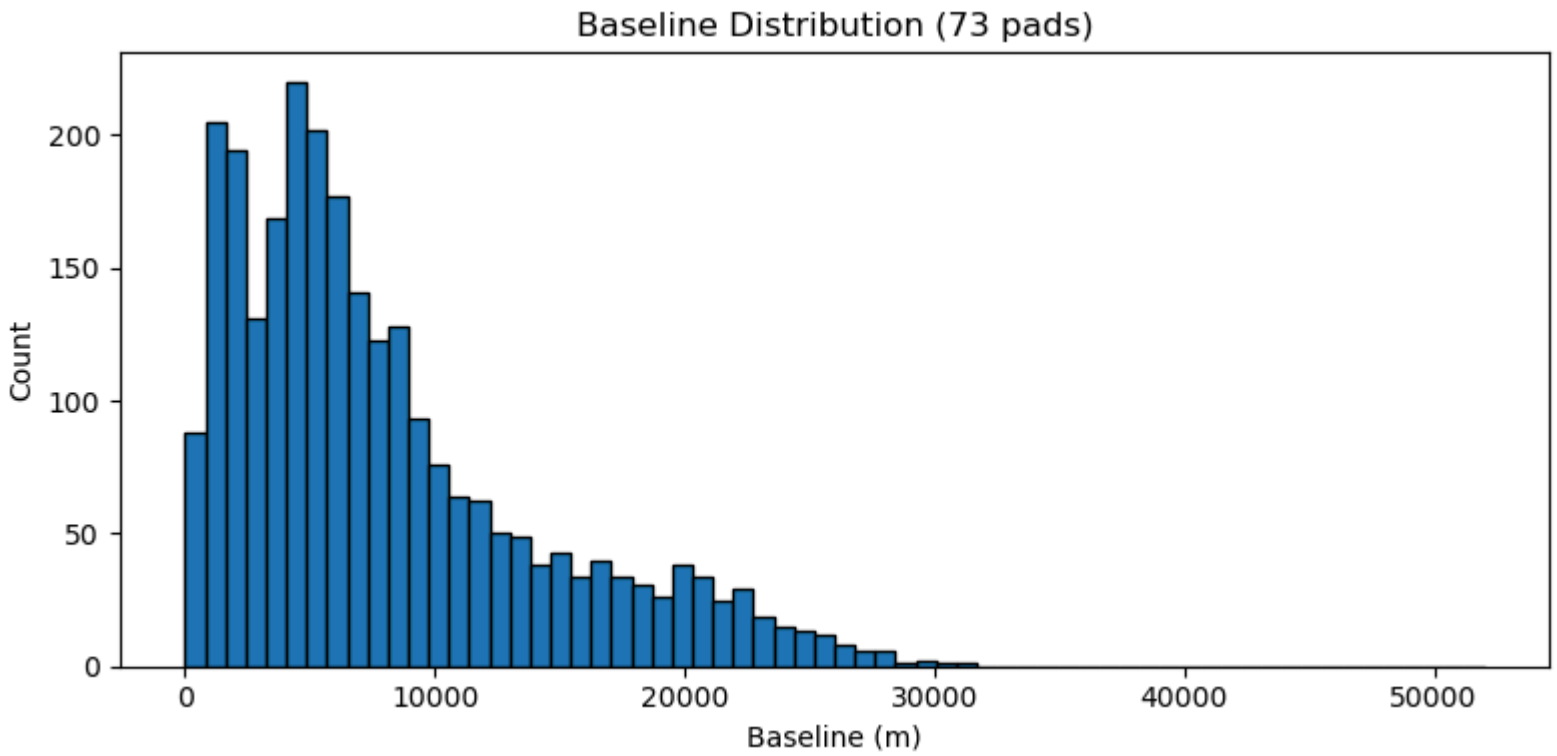
And if we switch "include_reuse" to True, and give a slight favor to reuse existing pads:

In [30]:
```
import argparse
from pad_placer_alma import load_data, optimize_pads, add_open_elevation_z, write_outputs, generate_plots

example_args = argparse.Namespace(
    cfg_path="alma.cycle11.10.cfg",
    kml_path="RegionsOSF.kml",
    poly_layers=['Regions OSF'],
    regions_kml="PadsAOS.kml",
    out_new="new_pads_57.cfg",
    out_plus57="alma.cycle11.10.plus_new.cfg",
    n_new=30, spacing=25.0, min_bl=200, R_max=30000, inner_radius=4700,
    favored_weight=0.99, include_reuse=True,
    cable_weight=5e-6, inner_ratio=2, outer_ratio=1,
    fixed_r_max=52000.0, ratio_penalty_weight=0,
```

```
    transition_width=6000.0, density_radius=1900.0, density_weight=0.4,
    sigma_b=24000, use_open_elevation=True, absolute_z=False
)

# 2) Load & optimize
existing_df, existing_pts, ex_weights, ctr_start, allowed_area, region_df, crs_loc, lat0, lon0 = load_data(example_args)
new_df, _, cand_df, sel_df, cand_pts, sel, fixed_r_max = optimize_pads(
    example_args, existing_df, existing_pts, ex_weights, ctr_start, allowed_area, region_df
)

# 3) Elevations (z in LOC: pad_elev - origin_elev)
if example_args.use_open_elevation:
    new_df = add_open_elevation_z(new_df, crs_loc, lat0, lon0, relative_to_origin=not example_args.absolute_z)

# 4) Save + plots
write_outputs(example_args, new_df, existing_df)
generate_plots(new_df, existing_df, fixed_r_max)
```

```
2025-08-10 20:09:47,377 - INFO - Loaded 170 region pads
2025-08-10 20:09:52,722 - INFO - Clipped 0 candidates beyond 30000 m
2025-08-10 20:09:52,728 - INFO - Total candidates: 1350 | Region: 170 | Inner: 152
2025-08-10 20:09:52,949 - INFO - Step 1/30: added 1320, cost=5740962.1
2025-08-10 20:09:53,028 - INFO - Step 2/30: added 94, cost=5806923.4
2025-08-10 20:09:53,088 - INFO - Step 3/30: added 16, cost=5820026.2
2025-08-10 20:09:53,136 - INFO - Step 4/30: added 5, cost=5838776.8
2025-08-10 20:09:53,178 - INFO - Step 5/30: added 199, cost=5859403.5
2025-08-10 20:09:53,219 - INFO - Step 6/30: added 787, cost=5884438.5
2025-08-10 20:09:53,252 - INFO - Step 7/30: added 17, cost=5910161.3
2025-08-10 20:09:53,287 - INFO - Step 8/30: added 1243, cost=5938025.0
2025-08-10 20:09:53,319 - INFO - Step 9/30: added 1234, cost=6032017.4
2025-08-10 20:09:53,357 - INFO - Step 10/30: added 100, cost=6130079.2
2025-08-10 20:09:53,383 - INFO - Step 11/30: added 1230, cost=6172400.8
2025-08-10 20:09:53,411 - INFO - Step 12/30: added 289, cost=6279970.9
2025-08-10 20:09:53,438 - INFO - Step 13/30: added 1231, cost=6326507.1
2025-08-10 20:09:53,467 - INFO - Step 14/30: added 1321, cost=6438605.3
2025-08-10 20:09:53,495 - INFO - Step 15/30: added 1225, cost=6552740.5
2025-08-10 20:09:53,526 - INFO - Step 16/30: added 1221, cost=6668902.0
2025-08-10 20:09:53,556 - INFO - Step 17/30: added 1232, cost=6787253.0
2025-08-10 20:09:53,583 - INFO - Step 18/30: added 1226, cost=6908082.9
2025-08-10 20:09:53,612 - INFO - Step 19/30: added 1233, cost=7032364.7
2025-08-10 20:09:53,640 - INFO - Step 20/30: added 1227, cost=7158075.3
2025-08-10 20:09:53,668 - INFO - Step 21/30: added 1224, cost=7287233.6
2025-08-10 20:09:53,698 - INFO - Step 22/30: added 1332, cost=7417047.2
2025-08-10 20:09:53,727 - INFO - Step 23/30: added 1106, cost=7549961.9
2025-08-10 20:09:53,758 - INFO - Step 24/30: added 1208, cost=7608101.9
2025-08-10 20:09:53,788 - INFO - Step 25/30: added 21, cost=7748290.6
2025-08-10 20:09:53,820 - INFO - Step 26/30: added 1209, cost=7811004.5
2025-08-10 20:09:53,866 - INFO - Step 27/30: added 1229, cost=7954056.5
2025-08-10 20:09:53,897 - INFO - Step 28/30: added 8, cost=8100706.0
2025-08-10 20:09:53,928 - INFO - Step 29/30: added 1223, cost=8167295.0
2025-08-10 20:09:53,960 - INFO - Step 30/30: added 1212, cost=8317146.2
2025-08-10 20:09:53,965 - INFO - Reused 19 region pads: ['W203', 'A130', 'A121', 'A117', 'A118', 'W204', 'A112', 'A108', 'A11
9', 'A113', 'A120', 'A114', 'A111', 'P405', 'A095', 'A096', 'A116', 'A110', 'A099']
2025-08-10 20:09:54,483 - INFO - Wrote new pads to new_pads_57.cfg
2025-08-10 20:09:54,488 - INFO - Wrote combined config to alma.cycle11.10.plus_new.cfg
```


ALMA Pads: Existing vs. New
Existing (43)
New (30)
North (m)
East (m)
8000
6000
4000
2000
0
−2000
−4000
−6000
−20000
−15000
−10000
−5000
0
5000



ALMA Simulation Pipeline — Quick Guide

This guide explains how to set up and use two Python scripts for ALMA simulations using CASA (tested to work on CASA v6.7.0.31 (might work on other versions as well)):

- `generate_sky_model.py` : Creates a FITS sky model with 8 Gaussian sources in a ring plus one central source. Configurable (e.g., declination, flux, FWHM, r, Cell Size, Imsize).
- `run_simalma_pipeline.py` : Automates sky model generation, `simalma` simulations, `tclean` imaging, and logs metrics (RMS, peak flux, beam size).

Workflow: Generate sky models → Simulate observations → Image and log results.

Prerequisites

1. **CASA Installation**
2. **ALMA Config Files**:
   - you can either use "standard"-cfgs included in CASA
   - or use your own custom-cfgs, they need to be placed in "casa/data/alma/simmos/" to work (same directory as the standard-cfgs reside in).
   - I found that CASA do not like when you name them whatever you want, the cfgs must have "alma.cycleXX" in order to work! => you can name them "alma.cycleXX.YY" as well (and possibly other names as well, but the "alma.cycle"-part seems to be mandatory).
3. **Python Dependencies**:

- **For Scripts**: All required packages (`os` , `sys` , `argparse` , `logging` , `json` , `math` , `subprocess` , `numpy` , `pathlib` , `casatools` , `casatasks`) are included in CASA 6.7.0.31 (no packages should thus be required to download). **No external installation needed**.

1. Inputs

Config files:

- **params.json** — master control for the pipeline run
- **sky_config.json** — Gaussian source model (ring of points)
- **sky_config_ring.json** — Concentric ring model (uniform disks)

Scripts:

- **generate_sky_model.py** — builds Gaussian-ring sky model FITS for CASA
- **generate_ring_sky_model.py** — builds concentric-ring sky model FITS for CASA
- **run_simalma_pipeline.py** — main driver: generates sky, runs `simalma` + `tclean` , exports FITS

2. How it works

1. **Sky model generation**

- Select script via `sky_script` in `params.json`
- Reads base config (`sky_config*.json`)
- Optionally overrides geometry/flux based on beam size (for `ring` script)
- Outputs `.fits` sky model

2. **Simulation**

- Finds ALMA config file (`config_map` in params)
- Computes beam size from baselines (L80)
- Picks cell size = beam / `sampling_factor`
- Runs CASA `simalma` to generate noisy visibilities

3. **Imaging & Cleaning**

- Runs `tclean` twice: dirty + cleaned image
- Auto-sets multiscale sizes from beam in pixels
- Threshold = `threshold_factor` × RMS (dirty)
- Saves `.image` and `.fits` products

4. **Logging**

- Stores RMS, peak flux, beam size, and parameters in `pipeline_log.json`

3. Key parameters (from params.json)

- `project_base` — root folder for results
- `config_map` — ALMA config file → short tag
- `integration_times` — total integration times (`"4h"` , `"8h"` , etc.)
- `declinations` — target declinations
- `center_freq` , `width` — band center & bandwidth
- `pwv` — precipitable water vapor (mm)
- `phasecenter_ra` — RA center (Dec comes from loop)
- `weightings` — `natural` , `uniform` , or `briggs`
- `robust_values` — robust settings for `briggs`
- `uvtaper_values` — uv-taper settings

- `niter`, `deconvolver`, `threshold_factor` — clean settings
- `fov_arcsec`, `sampling_factor` — image size control
- `sky_script`, `sky_config` — which sky generator and config JSON to use
- `casa_bin` — path to CASA executable
- `extra_dirs` — extra search paths for ALMA config files

4. Running the pipeline

Note: since the pipeline needs CASA to run ive tried to show step by step on how to get it to run (for the nine Gaussians skymodel).

- Save `generate_sky_model.py` and `run_simalma_pipeline.py` in your working directory (Also save `params.json` and `sky_config.json` after changing to desired values, also make excecutable; chmod +x ~/run_simalma_pipeline.py etc).
- **The following is params.json, change as desired (PS you need to change `casa_bin` to YOUR path):**

```
cat > ~/params.json << 'EOF'
{
  "project_base": "alma2040_pipeline",          // Base directory where all simulation results will be stored
  "config_map": {
    "alma.cycle11.10.cfg": "C43-10"             // Mapping of ALMA configuration file name -> short tag for labeling outputs
  },
  "integration_times": ["0.5h", "1h"],          // Total integration times to simulate
  "declinations": ["-23d00m00.00"],             // Target declinations (format: ±DDdMMmSS.ss)
  "center_freq": "343.5GHz",                    // Observation frequency (GHz)
  "width": "7.5GHz",                            // Total bandwidth for simulation (GHz)
  "pwv": 1.262,                                 // Precipitable Water Vapor (mm) — controls atmospheric transparency in simalma
  "phasecenter_ra": "12h00m00.00s",             // RA of the phase center (format: HHhMMmSS.ss)
  "weightings": ["briggs", "uniform"],          // Imaging weightings to use in tclean: natural, uniform, briggs
  "robust_values": [-1, 0.5],                   // Robust parameters to try when weighting = briggs
  "uvtaper_values": ["0.0arcsec"],              // uv-taper values for imaging resolution control (arcsec)
  "niter": 7000,                                // Number of iterations for deconvolution (tclean)
  "deconvolver": "multiscale",                  // Deconvolver type (tclean) — e.g. hogbom, multiscale
  "threshold_factor": 1.5,                      // Clean threshold multiplier relative to dirty RMS
  "fov_arcsec": 0.3072,                         // Field of view in arcseconds
  "sampling_factor": 5,                         // Number of pixels across the beam (beam / sampling_factor = cell size)
  "sky_script": "generate_sky_model.py",        // Which script to use to generate the sky model
  "sky_config": "sky_config.json",              // JSON configuration file for the sky model
  "casa_bin": "/home/hampus/casa-6.7.0-31-py3.10.el8/bin/casa", // Path to CASA binary to run simalma/tclean
  "log_file": "pipeline_test.log",              // Log file for pipeline output
  "extra_dirs": []                              // Extra directories to search for ALMA config files (in addition to defaults)
}
EOF
```

- **The following is sky_config.json, change as desired:**

```
cat > ~/sky_config.json << 'EOF'
{
  "ra_center": "12h00m00.00s",          // RA center for sky model (HHhMMmSS.ss)
  "dec_center": "-23d00m00.00",         // Dec center for sky model (±DDdMMmSS.ss)
  "flux": 0.00027,                      // Flux per source (Jy) in the ring
  "freq": "343.5GHz",                   // Observation frequency
  "n_ring": 8,                          // Number of sources placed evenly on the ring
  "radius": 0.024,                      // Radius of the ring (arcsec)
  "major_beam": "0.012arcsec",          // Gaussian major axis for each source
  "minor_beam": "0.012arcsec",          // Gaussian minor axis for each source
  "pa_beam": "0deg",                    // Position angle of each Gaussian source
  "output_base": "eightPlusCenter",     // Base name for output FITS and image files
  "im_shape": [128, 128],               // Image dimensions in pixels (x, y)
  "cell_size": 0.0024,                  // Pixel scale in arcsec
  "log_file": "sky_test.log"            // Log file name for sky model generation
}
EOF
```

Both of these are available on github. After changing as desired, and after placing the updated .json files (and scripts) in your working directory, you can now run the pipeline:

-To run pipeline do: `casa --nogui --nologger -c run_simalma_pipeline.py params.json`

-The following is a screenshot from the terminal:

```
(base) hampus@Hampus:~$ casa --nogui --nologger -c run_simalma_pipeline.py params.json

optional configuration file not found, continuing CASA startup without it

Using matplotlib backend: tkagg
CASA 6.7.0.31 -- Common Astronomy Software Applications [6.7.0.31]
2025-08-10 21:07:17,827 - INFO - Found config file at /home/hampus/.casa/data/alma/simmos/alma.cycle11.10.cfg
2025-08-10 21:07:17,831 - INFO - [C43-10] Estimated beam FWHM ≈ 0.0118 arcsec
2025-08-10 21:07:17,831 - INFO - Cell size set to 0.0023630372818205 44arcsec
2025-08-10 21:07:17,835 - INFO - Adjusted imsize from 132 to efficient 144
2025-08-10 21:07:17,835 - INFO - Generating sky model for declination -23d00m00.00
2025-08-10 21:07:17,836 - INFO - Running sky generator with: ['/home/hampus/casa-6.7.0-31-py3.10.el8/bin/casa', '--nogui', '--nologger', '-c', 'generate_sky_model.py', 'skyconfig_m2300m0000_C43-10.json']

optional configuration file not found, continuing CASA startup without it

Using matplotlib backend: tkagg
CASA 6.7.0.31 -- Common Astronomy Software Applications [6.7.0.31]
2025-08-10 21:07:19,381 - DEBUG - sys.argv: ['generate_sky_model.py', 'skyconfig_m2300m0000_C43-10.json']
2025-08-10 21:07:19,381 - DEBUG - CASA_ARGUMENTS: Not set
2025-08-10 21:07:19,381 - DEBUG - Validating RA: 12h00m00.00s
2025-08-10 21:07:19,381 - DEBUG - Successfully converted RA 12h00m00.00s to {'unit': 'rad', 'value': 3.141592653589793}
2025-08-10 21:07:19,381 - DEBUG - Validating Dec: -23d00m00.00
2025-08-10 21:07:19,381 - DEBUG - Successfully converted Dec -23d00m00.00 to {'unit': 'rad', 'value': -0.4014257279586958}
2025-08-10 21:07:19,382 - INFO - Generating sky model with declination -23d00m00.00
2025-08-10 21:07:19,492 - INFO - ✅ Done: eightPlusCenter_C43-10_decm2300m0000.fits created at declination -23d00m00.00
2025-08-10 21:07:19,796 - INFO - simalma: {'project': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000', 'skymodel': '/home/hampus/eightPlusCenter_C43-10_decm2300m0000.fits', 'inc
enter': '343.5GHz', 'inwidth': '7.5GHz', 'integration': '10s', 'totaltime': '0.5h', 'hourangle': 'transit', 'pwv': 1.262, 'cell': '0.0023630372818205 44arcsec', 'antennalist': '/home/hampus/.casa/data/alma/simm
os/alma.cycle11.10.cfg', 'image': True, 'setpointings': True, 'graphics': 'both', 'verbose': True, 'indirection': 'J2000 12h00m00.00s -23d00m00.00', 'overwrite': True, 'dryrun': False, 'mapsize': ['0.3072arcse
c']}
```

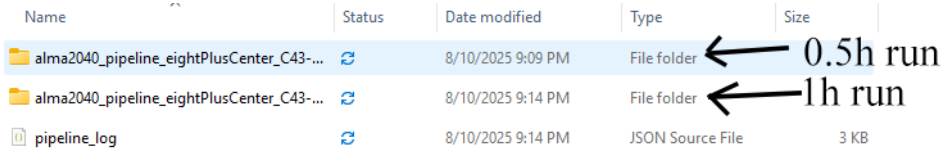
-And then it will run for a while depending on how many various observation times, weightings, etc you chosed:

```
BEAM MAJOR FROM PSF: 0.011012126691639423
2025-08-10 21:07:41,479 - INFO - Updated multiscale sizes for taper=0.0arcsec: [0, 5, 9, 23] (beam_pix=4.7)
2025-08-10 21:07:41,481 - INFO - rms_dirty = 7.651e-05 Jy/beam → threshold = 0.00011476994699626821 Jy/beam
2025-08-10 21:07:41,481 - INFO - Running tclean with parameters: {'vis': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000.alma.cycle11.10.noisy.ms', 'imagename': 'eightPlusCenter_
C43-10_decm2300m0000_C43-10_t0.5_m23_Br-1_t0p0_d144', 'weighting': 'briggs', 'cell': '0.0023630372818205 44arcsec', 'imsize': 144, 'deconvolver': 'multiscale', 'scales': [0, 5, 9, 23], 'niter': 7000, 'threshold
': np.float64(0.00011476994699626821), 'phasecenter': 'J2000 12h00m00.00s -23d00m00.00', 'uvtaper': ['0.0arcsec'], 'interactive': False, 'savemodel': 'modelcolumn', 'robust': -1}

0%....10....20....30....40....50....60....70....80....90....100%
2025-08-10 19:07:59     WARN    SDAlgorithmMSClean::takeOneStep (file /source/casa6/casatools/src/code/synthesis/ImagerObjects/SDAlgorithmMSClean.cc, line 185) MSClean minor cycle stopped at large scale negati
ve or diverging

0%....10....20....30....40....50....60....70....80....90....100%
2025-08-10 21:08:20,565 - INFO - Post-clean residual RMS = 4.355e-05
2025-08-10 21:08:20,604 - INFO - rms_cleaned = 5.350500830901598e-05
2025-08-10 21:08:20,606 - INFO - rms_cleaned_res = 4.3547631630364 74e-05
2025-08-10 21:08:20,614 - INFO - Running tclean (dirty) with parameters: {'vis': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000.alma.cycle11.10.noisy.ms', 'imagename': 'eightPlu
sCenter_C43-10_decm2300m0000_C43-10_t0.5_m23_Br0p5_t0p0_d144_dirty', 'weighting': 'briggs', 'cell': '0.0023630372818205 44arcsec', 'imsize': 144, 'niter': 0, 'phasecenter': 'J2000 12h00m00.00s -23d00m00.00', 'u
vtaper': ['0.0arcsec'], 'interactive': False, 'calcpsf': True, 'calcres': True, 'robust': 0.5}
2025-08-10 19:08:29     WARN    task_tclean::SIImageStore::restore (file /source/casa6/casatools/src/code/synthesis/ImagerObjects/SIImageStore.cc, line 2298)   Restoring with an empty model image. Only residua
ls will be processed to form the output restored image.
BEAM MAJOR FROM PSF: 0.012508489191532135
2025-08-10 21:08:29,604 - INFO - Updated multiscale sizes for taper=0.0arcsec: [0, 5, 11, 26] (beam_pix=5.3)
2025-08-10 21:08:29,607 - INFO - rms_dirty = 3.992e-05 Jy/beam → threshold = 5.987918240311304e-05 Jy/beam
2025-08-10 21:08:29,608 - INFO - Running tclean with parameters: {'vis': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000.alma.cycle11.10.noisy.ms', 'imagename': 'eightPlusCenter_
C43-10_decm2300m0000_C43-10_t0.5_m23_Br0p5_t0p0_d144', 'weighting': 'briggs', 'cell': '0.0023630372818205 44arcsec', 'imsize': 144, 'deconvolver': 'multiscale', 'scales': [0, 5, 11, 26], 'niter': 7000, 'thresho
ld': np.float64(5.987918240311304e-05), 'phasecenter': 'J2000 12h00m00.00s -23d00m00.00', 'uvtaper': ['0.0arcsec'], 'interactive': False, 'savemodel': 'modelcolumn', 'robust': 0.5}

0%....10....20....30....40....50....60....70....80....90....100%

0%....10....20....30....40....50....60....70....80....90....100%
2025-08-10 21:09:11,772 - INFO - Post-clean residual RMS = 2.610e-05
2025-08-10 21:09:11,803 - INFO - rms_cleaned = 3.419104006726397e-05
2025-08-10 21:09:11,806 - INFO - rms_cleaned_res = 2.6099597398914048e-05
2025-08-10 21:09:11,814 - INFO - Running tclean (dirty) with parameters: {'vis': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000.alma.cycle11.10.noisy.ms', 'imagename': 'eightPlu
sCenter_C43-10_decm2300m0000_C43-10_t0.5_m23_U_t0p0_d144_dirty', 'weighting': 'uniform', 'cell': '0.0023630372818205 44arcsec', 'imsize': 144, 'niter': 0, 'phasecenter': 'J2000 12h00m00.00s -23d00m00.00', 'uvta
per': ['0.0arcsec'], 'interactive': False, 'calcpsf': True, 'calcres': True}
2025-08-10 19:09:19     WARN    task_tclean::SIImageStore::restore (file /source/casa6/casatools/src/code/synthesis/ImagerObjects/SIImageStore.cc, line 2298)   Restoring with an empty model image. Only residua
ls will be processed to form the output restored image.
BEAM MAJOR FROM PSF: 0.011000969447 19553
2025-08-10 21:09:19,200 - INFO - Updated multiscale sizes for taper=0.0arcsec: [0, 5, 9, 23] (beam_pix=4.7)
2025-08-10 21:09:19,203 - INFO - rms_dirty = 8.129e-05 Jy/beam → threshold = 0.00012193464286344 1 Jy/beam
2025-08-10 21:09:19,204 - INFO - Running tclean with parameters: {'vis': 'alma2040_pipeline_eightPlusCenter_C43-10_decm2300m0000_C43-10_0.5h_-23_000000.alma.cycle11.10.noisy.ms', 'imagename': 'eightPlusCenter_
C43-10_decm2300m0000_C43-10_t0.5_m23_U_t0p0_d144', 'weighting': 'uniform', 'cell': '0.0023630372818205 44arcsec', 'imsize': 144, 'deconvolver': 'multiscale', 'scales': [0, 5, 9, 23], 'niter': 7000, 'threshold':
 np.float64(0.00012193464286344 1), 'phasecenter': 'J2000 12h00m00.00s -23d00m00.00', 'uvtaper': ['0.0arcsec'], 'interactive': False, 'savemodel': 'modelcolumn'}

0%....10....20....30....40....50....60....70....80....90....100%

0%....10....20....30....40....50....60....70....80....90....100%
2025-08-10 21:09:49,464 - INFO - Post-clean residual RMS = 4.593e-05
2025-08-10 21:09:49,496 - INFO - rms_cleaned = 5.612191964702 1466e-05
2025-08-10 21:09:49,498 - INFO - rms_cleaned_res = 4.5932633419195276e-05
```

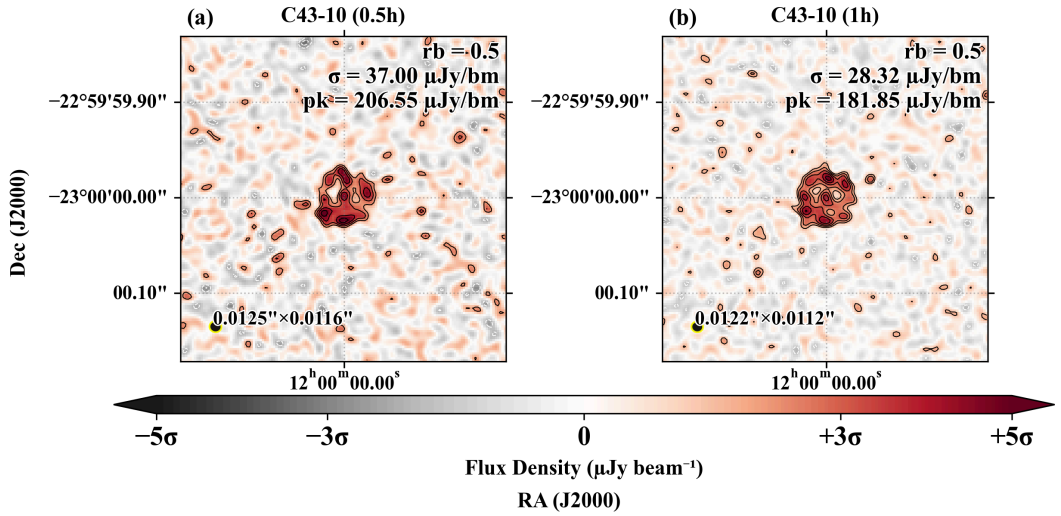
-When it is done running you will have all the data/results from the simulations in `alma2040_pipeline` (i.e. the project base you specified):

```
0%....10....20....30....40....50....60....70....80....90....100%

0%....10....20....30....40....50....60....70....80....90....100%

0%....10....20....30....40....50....60....70....80....90....100%

0%....10....20....30....40....50....60....70....80....90....100%
2025-08-10 21:14:41,307 - INFO - Post-clean residual RMS = 3.108e-05
2025-08-10 21:14:41,371 - INFO - rms_cleaned = 3.8244964641959 99e-05
2025-08-10 21:14:41,374 - INFO - rms_cleaned_res = 3.1077732440142 29e-05
2025-08-10 21:14:41,383 - INFO - Pipeline complete. Log at /home/hampus/alma2040_pipeline/pipeline_log.json
X connection to :0 broken (explicit kill or server shutdown).
(base) hampus@Hampus:~$ cd ~/alma2040_pipeline
mv * /mnt/c/Users/hampu/OneDrive/Skrivbord/alma2040/CASA/files/
(base) hampus@Hampus:~/alma2040_pipeline$
```

-In the following folders you will have all the .fits files, as well as all the data from simalma like .psf, .residual, .model, ...:



-You can now plot the resulting .fits files however you like. The following is from the run we just did:



In [ ]: