MeerKLASS Map-making Software

Amadeus Wild - SKA cosmology SWG meeting Jan 2023
University of the Western Cape
University of Manchester

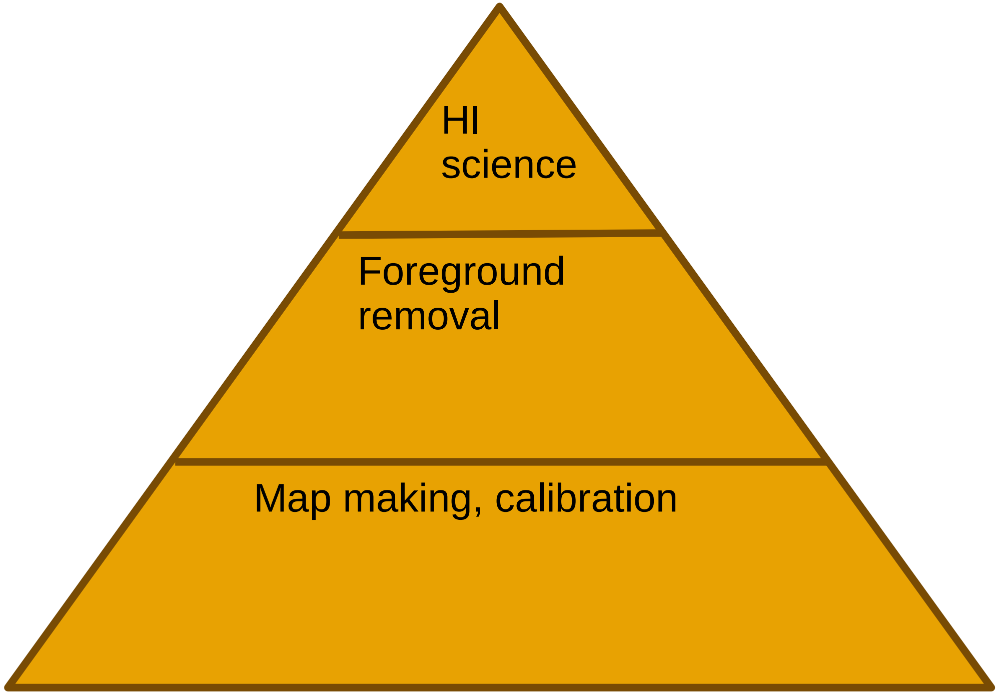
Ivory & MuSEEK

map-making packages

`github.com/meerklass/museek`

- Inspired by HIDE&SEEK: modular plugin-based architecture
- Ivory: workflow manager
  MuSEEK: map making
- Developed using MeerKAT (for SKA)
- Plugins are isolated and do not share responsibilities

~~Food~~ Data Pyramid

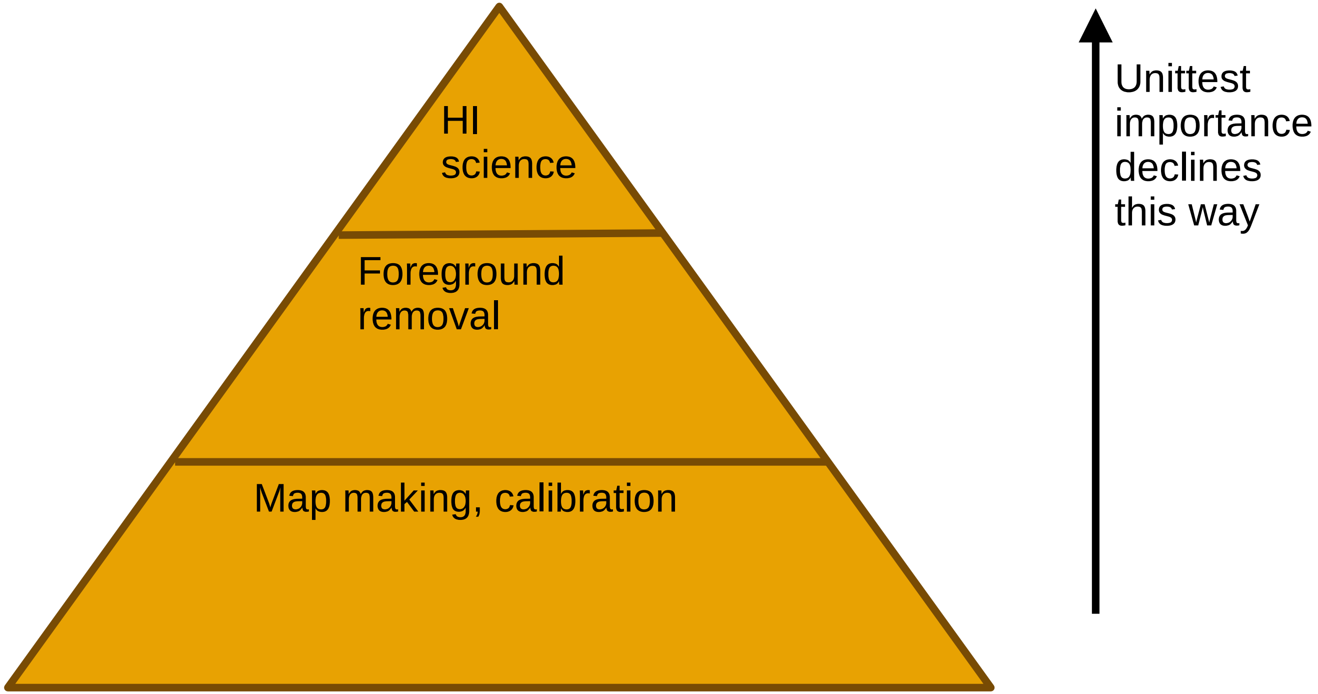


HI science

Foreground removal

Map making, calibration

~~Food~~ Data Pyramid



HI science

Foreground removal

Map making, calibration

Unittest importance declines this way

~~Food~~ Data Pyramid

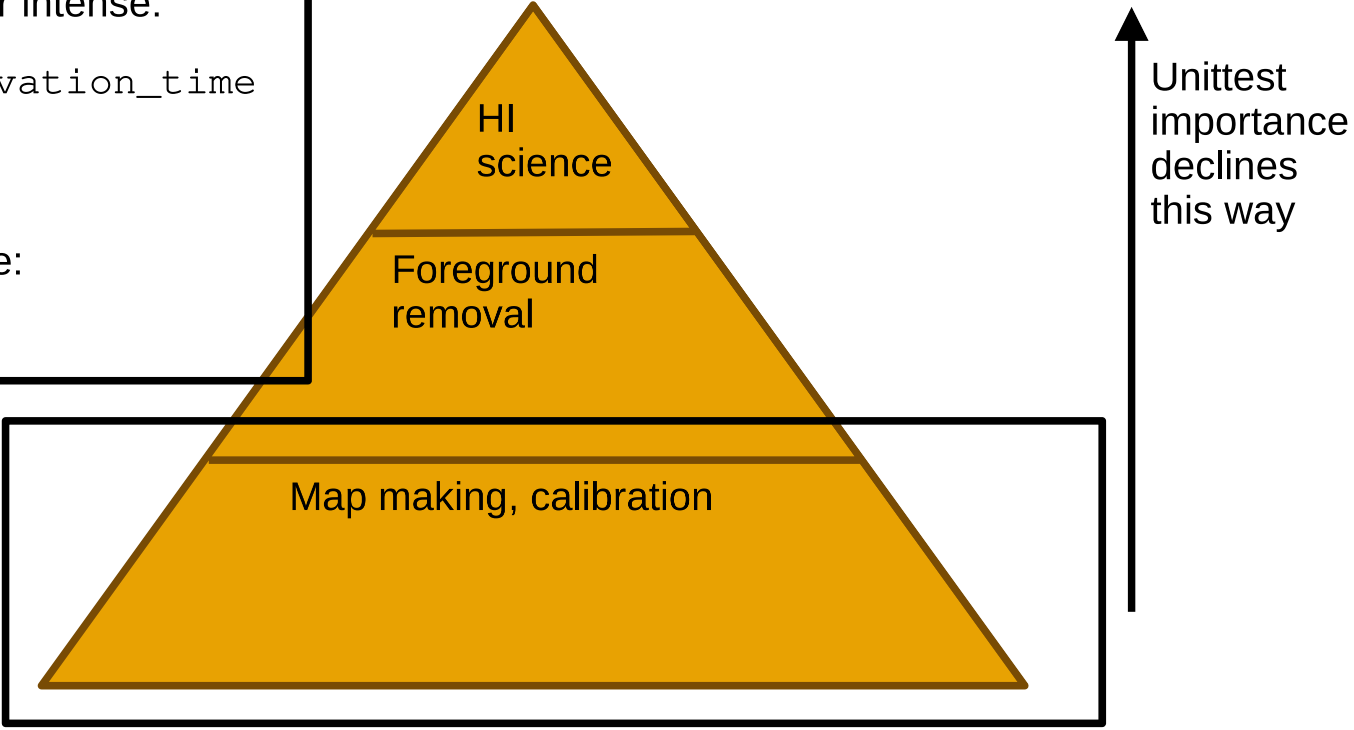
Map making can be labour intense:

```
labour_time ∝ observation_time
```

(not good with SKA1…)

Good software should give:

```
labour_time = const
```

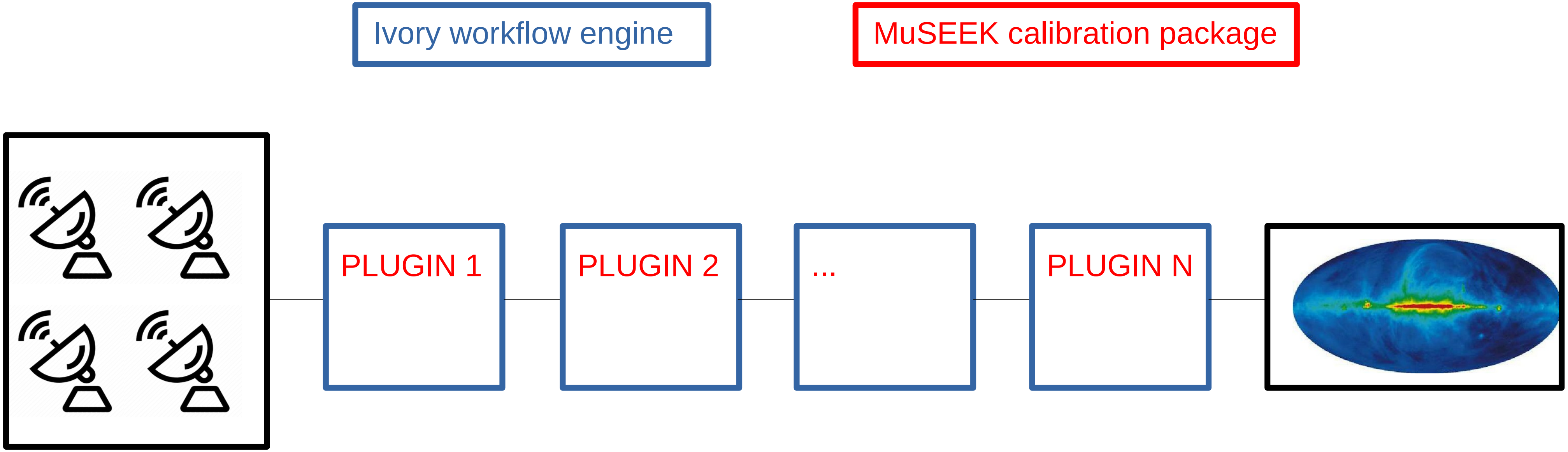


HI science

Foreground removal

Map making, calibration

Unittest importance declines this way



Ivory workflow engine

MuSEEK calibration package

PLUGIN 1

PLUGIN 2

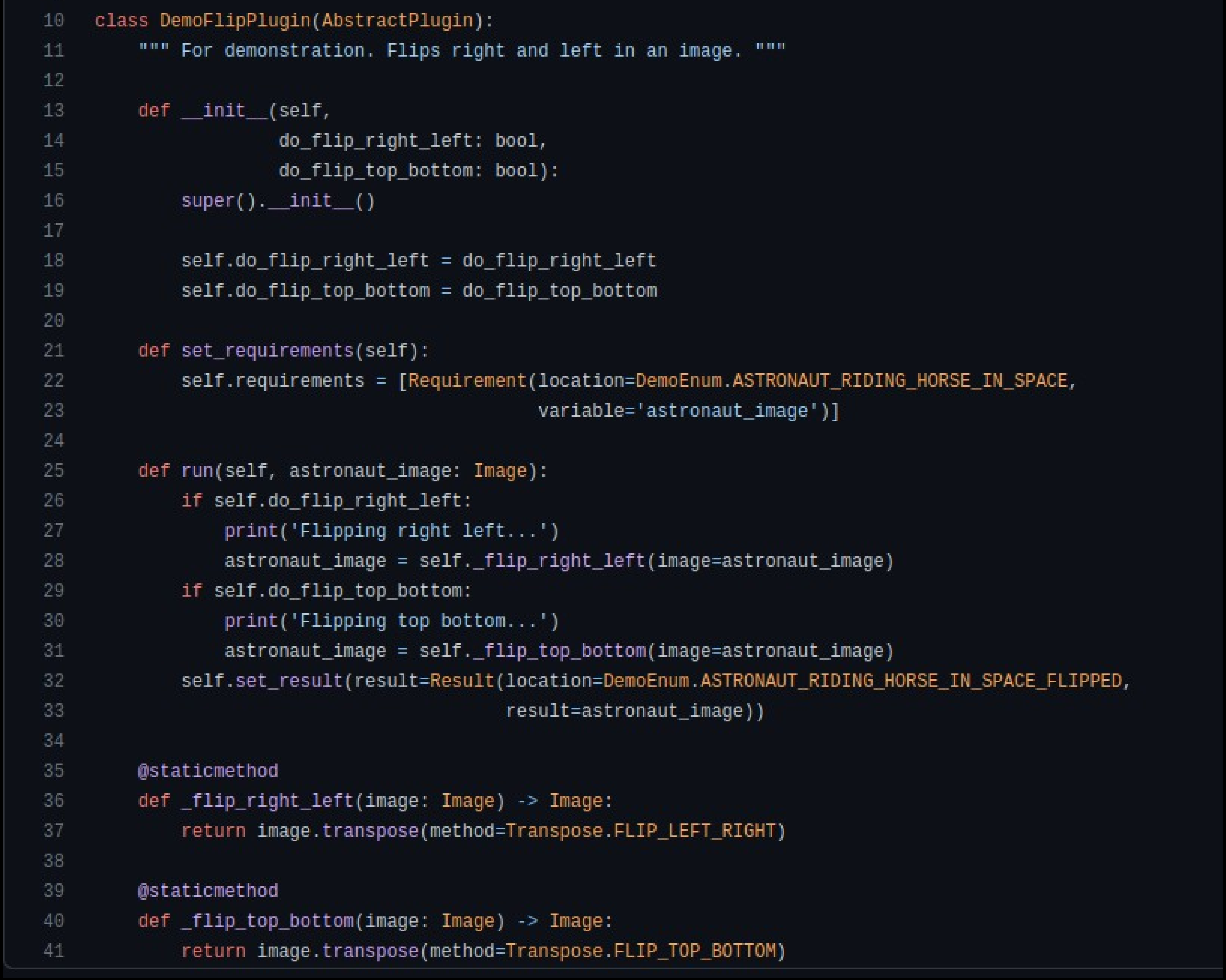
...

PLUGIN N

- One plugin needs **no knowledge about the implementation of the rest of the pipeline.**

- Different instruments have different calibration requirements, but generalisation, e.g. from MeerKAT to SKA1 must be made as easy as possible

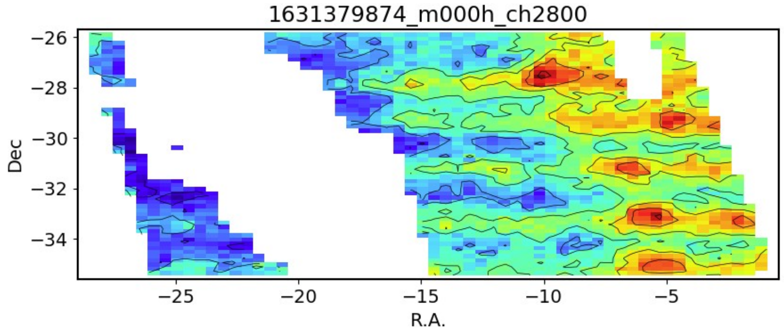
- Useful approach when calibration strategies quickly need to be changed.



```python
class DemoFlipPlugin(AbstractPlugin):
    """ For demonstration. Flips right and left in an image. """

    def __init__(self,
                 do_flip_right_left: bool,
                 do_flip_top_bottom: bool):
        super().__init__()

        self.do_flip_right_left = do_flip_right_left
        self.do_flip_top_bottom = do_flip_top_bottom

    def set_requirements(self):
        self.requirements = [Requirement(location=DemoEnum.ASTRONAUT_RIDING_HORSE_IN_SPACE,
                                         variable='astronaut_image')]

    def run(self, astronaut_image: Image):
        if self.do_flip_right_left:
            print('Flipping right left...')
            astronaut_image = self._flip_right_left(image=astronaut_image)
        if self.do_flip_top_bottom:
            print('Flipping top bottom...')
            astronaut_image = self._flip_top_bottom(image=astronaut_image)
        self.set_result(result=Result(location=DemoEnum.ASTRONAUT_RIDING_HORSE_IN_SPACE_FLIPPED,
                                      result=astronaut_image))

    @staticmethod
    def _flip_right_left(image: Image) -> Image:
        return image.transpose(method=Transpose.FLIP_LEFT_RIGHT)

    @staticmethod
    def _flip_top_bottom(image: Image) -> Image:
        return image.transpose(method=Transpose.FLIP_TOP_BOTTOM)
```

Example use-case

- Imagine: A new observation is affected by unknown systematics
- A correction requires in-depth understanding of the specific problem
- Understanding pipeline implementation specifics should not be needed

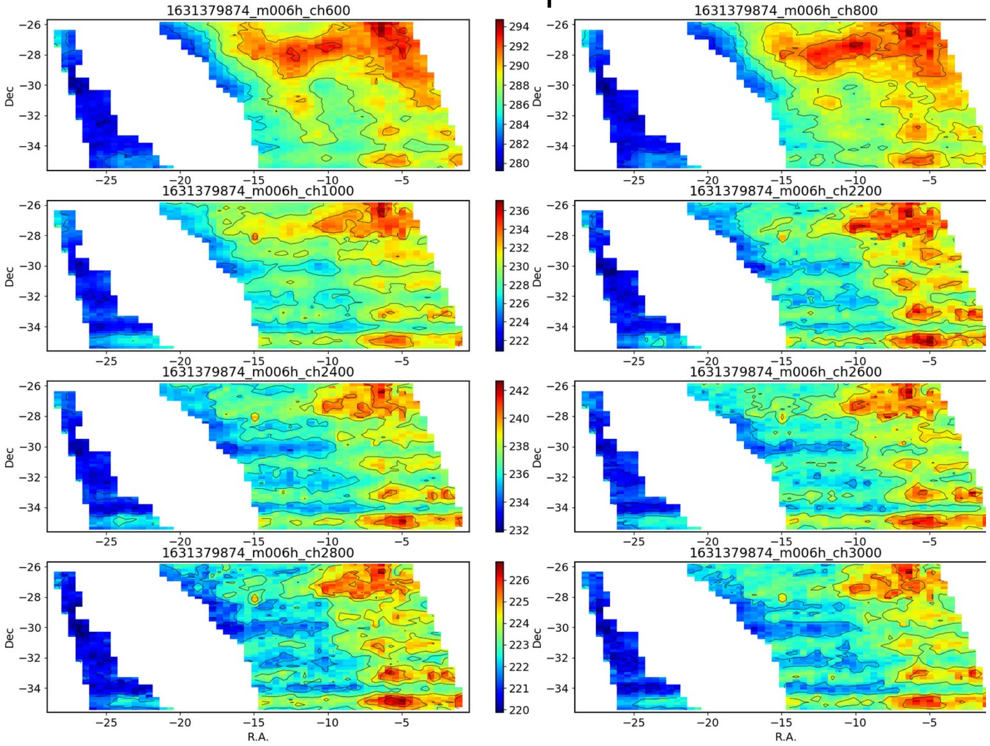


We dubbed this the ‹zebra› and explain as follows:

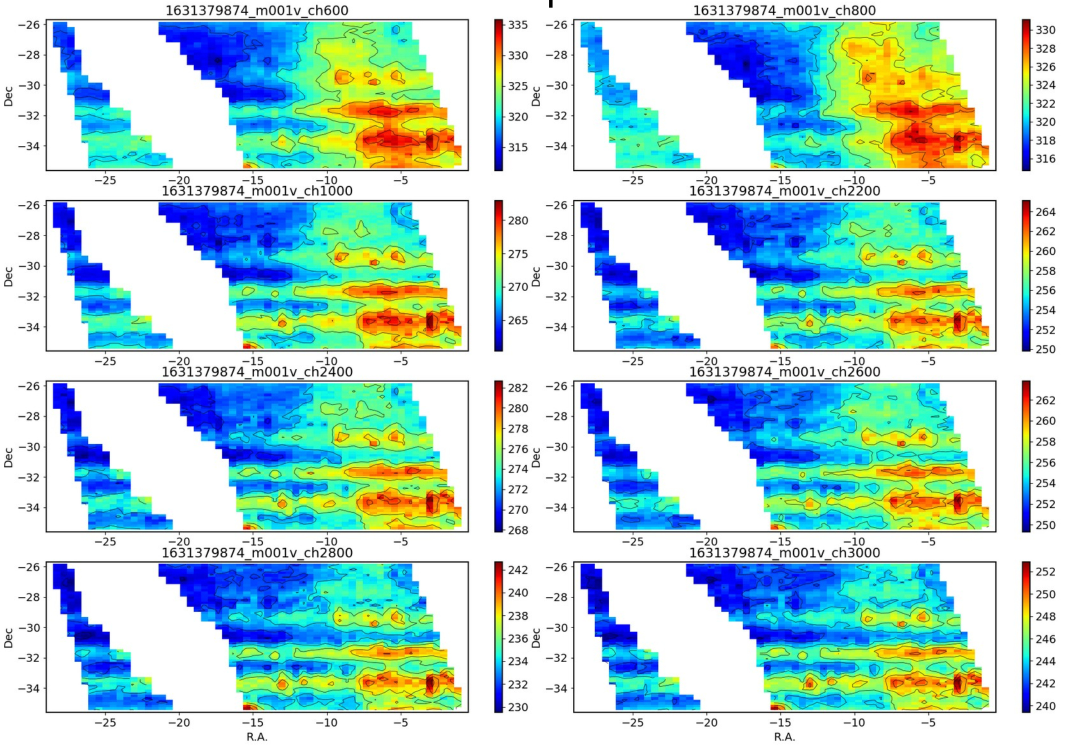
- A mobile communication tower <u>behind</u> the dishes injects a lot of power into a few channels, enough to affect the gain throughout all channels, hence arises an azimuth dependence of the gain

Example use-case

Dish 6 pol h



Dish 1 pol v

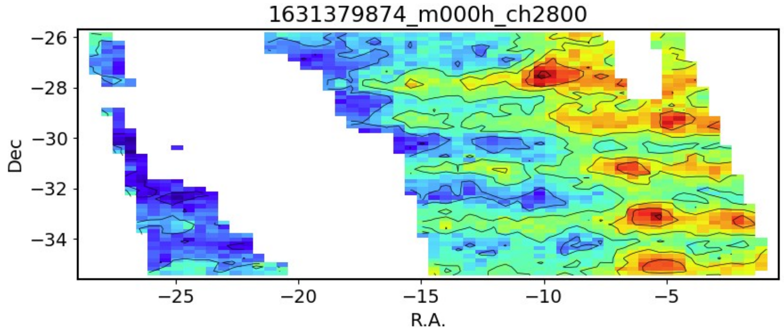


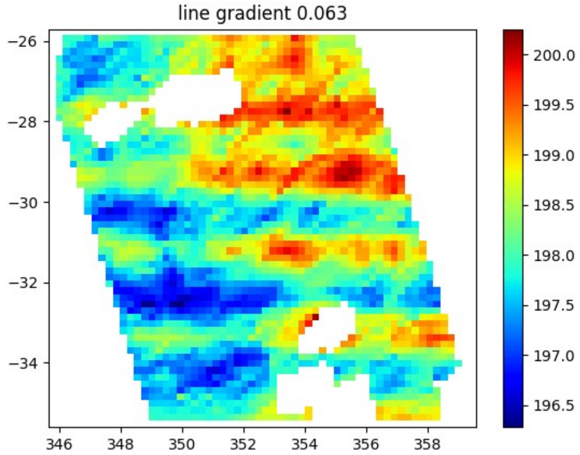
We dubbed this the ‹zebra› and explain as follows:

- A mobile communication tower behind the dishes injects a lot of power into a few channels, enough to affect the gain throughout all channels, hence arises an azimuth dependence of the gain

Example use-case

- Imagine: A new observation is affected by unknown systematics
- A correction requires in-depth understanding of the specific problem
- Once understood, the fix is implemented as a new plugin.
- This implementation requires no knowledge of the implementation of the calibration pipeline.





Summary

- Having many dependants, map-making needs to be well tested and robust
- The learning curve must be flat to enhance collaboration
- At the same time the architecture needs to be quick to react to changing requirements
- The purpose-built *ivory* workflow manager facilitates that

«To measure the sky, with precision and care
The gain must be known, beyond all compare
From faint signals, to the brightest of stars
Accurate measurements, from afar

But gain can vary, with time and place
So calibration, is a crucial race
To ensure accuracy, in all we do
Gain calibration, a task that is true»