To do

- Master bias variance images

- Classification of amplifiers according to the type of issue

- Parallel overscan

Smoothing for the wave cases?

- Serial overscan: need to assess which amplifiers require this correction

- Yellow corners

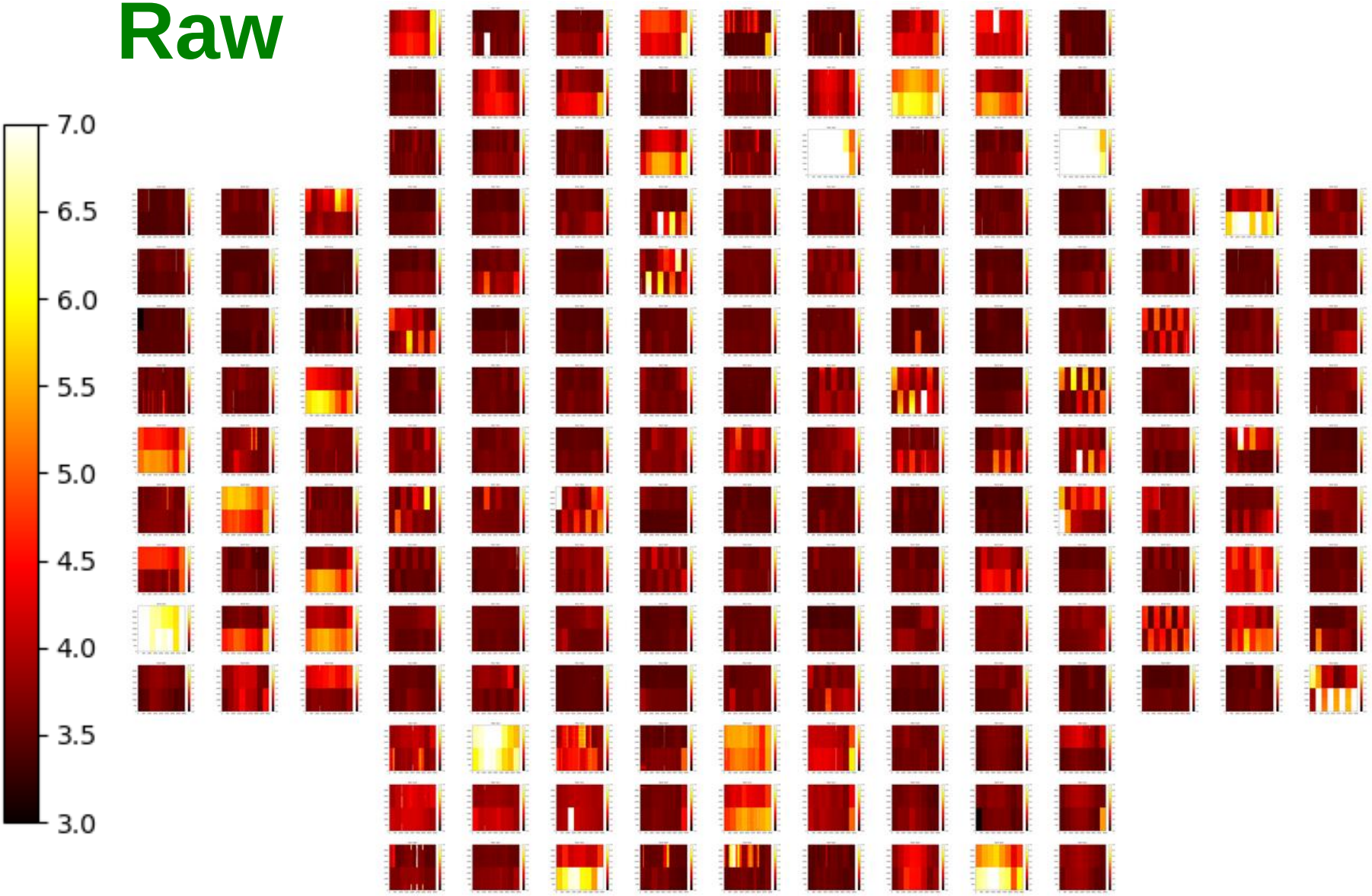
exposure time / sequence (previous image: flux and exposure time)

- Residual column structure on master bias after 2D correction

- Impact of 1D/2D overscan corrections on photometry

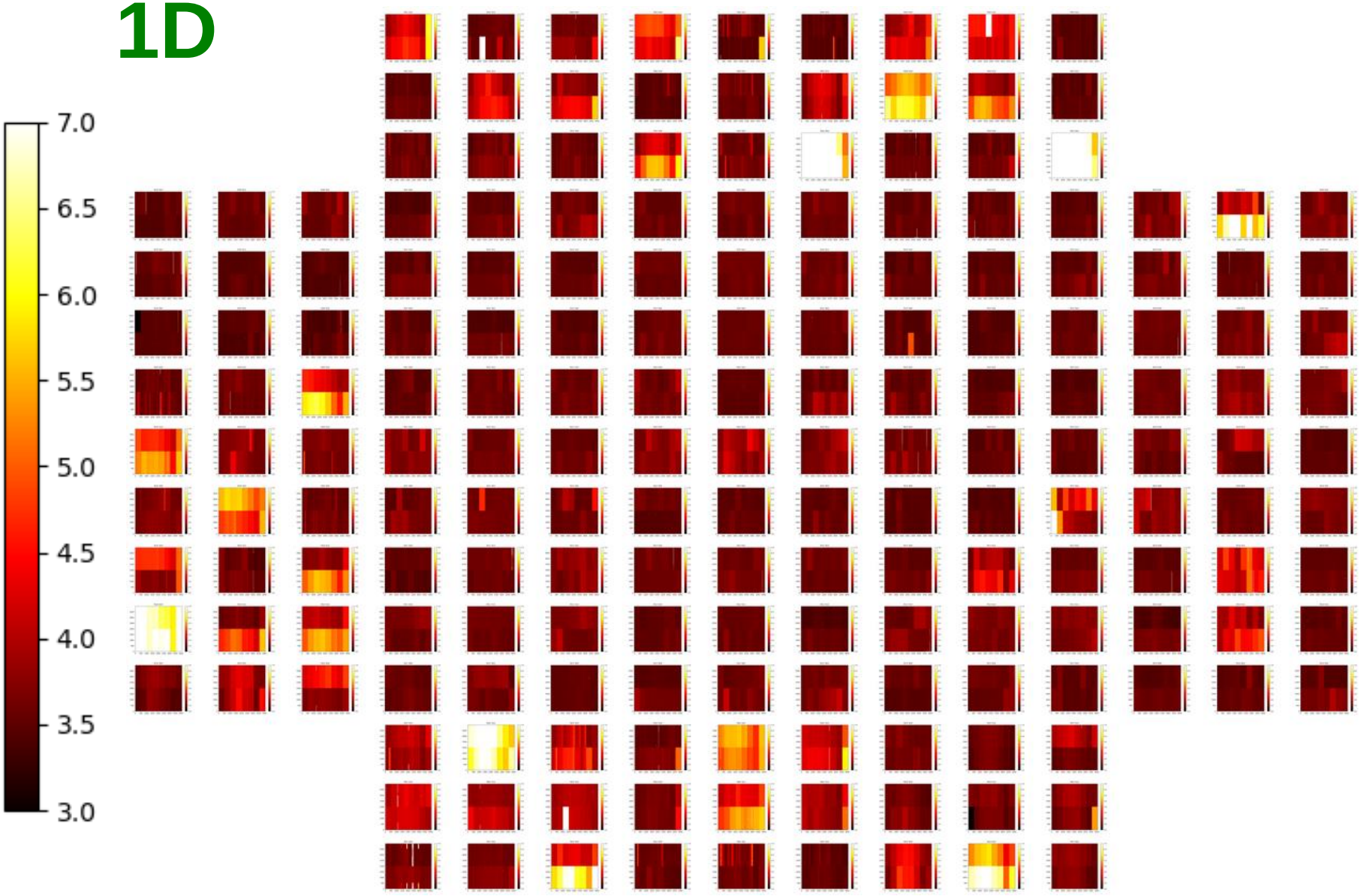
RMS by hand

Raw



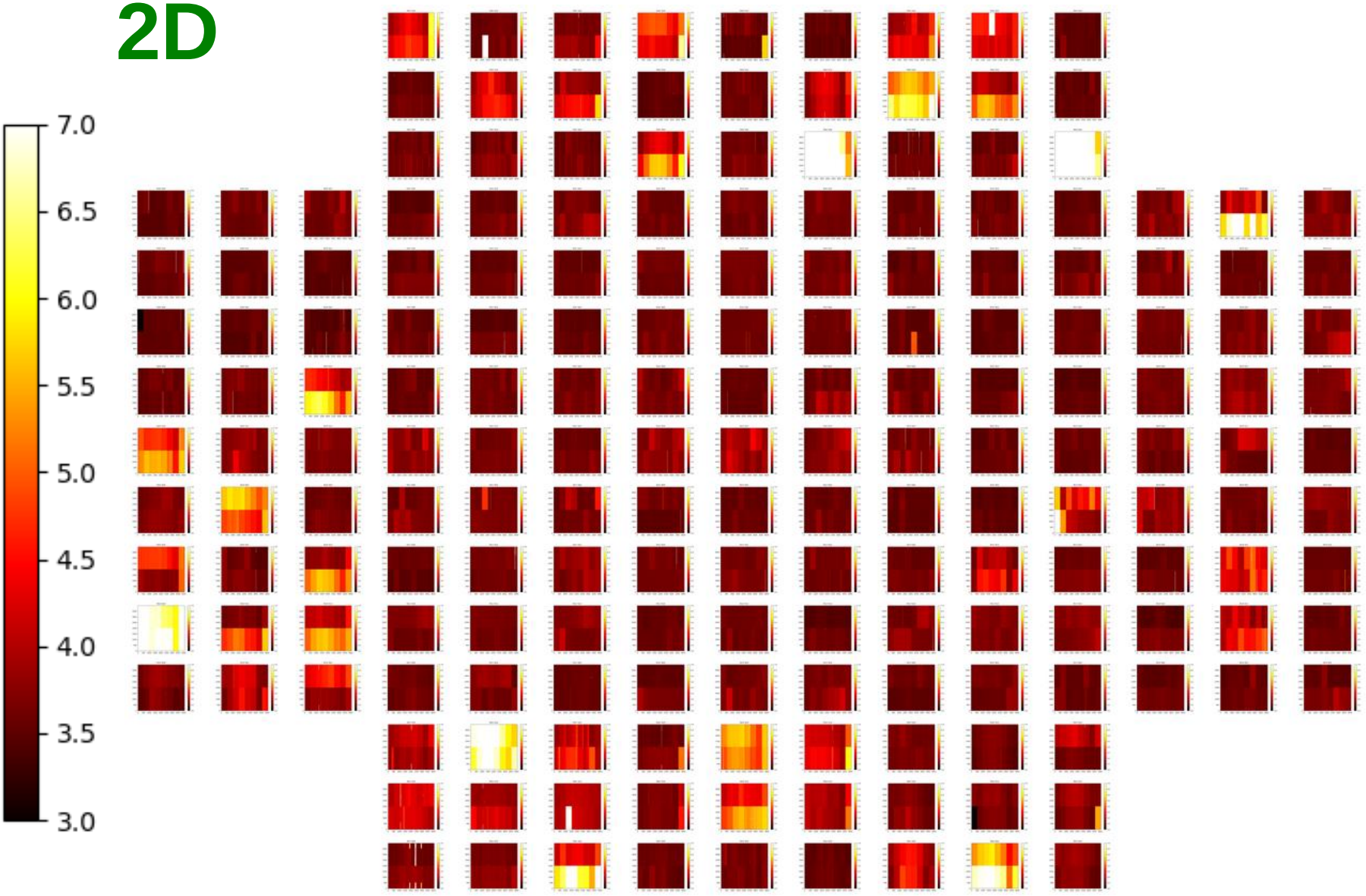
RMS by hand

1D



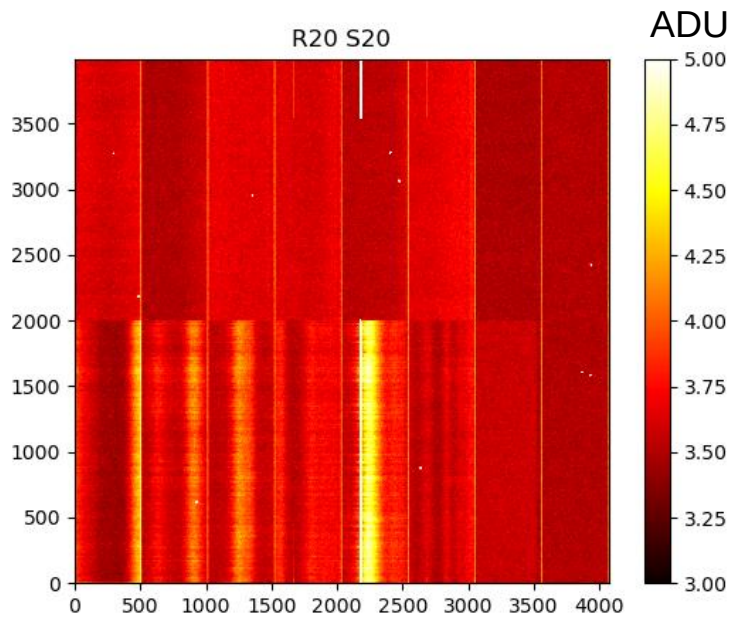
RMS by hand

2D



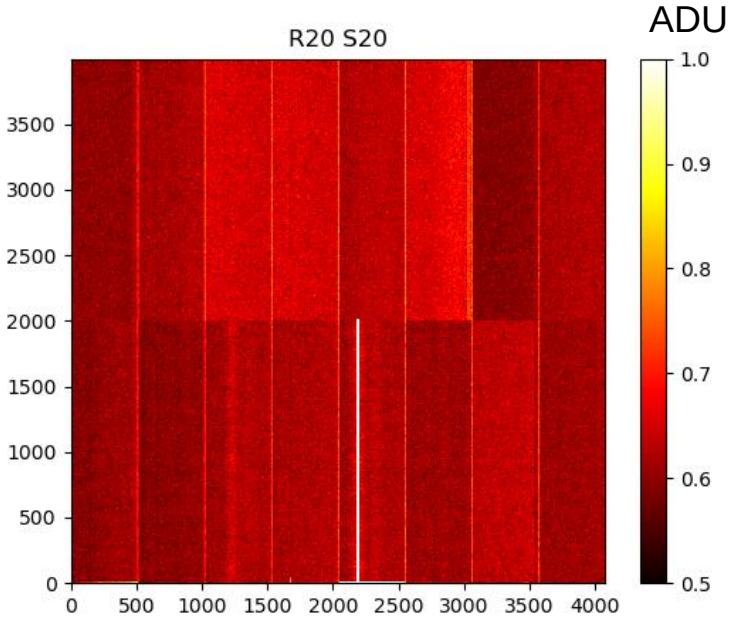
RMS comparison

By hand



afwMath.STDEV

DM



MEANCLIP for MB
calib.getVariance().getArray()
consistent with RMS/sqrt(20)?

RMS comparison

Chris Waters il y a 7 heures

I believe so, but to be exact, I sorted through the stacking code to make sure I understood what was happening. I've attached my final set of equations. The "default" setting is what `cpCombine` does currently: there is no weighting applied, so the input variances are dropped and the combined variance generated from the scatter of the input image values. There are two mutually exclusive `setCalcError` options that can be applied, and a variance weighting option, making six possible ways to combine the inputs using `afw.statisticsStack`. I think I've convinced myself that the default variance calculation is reasonable for biases and darks, and that the flat should be fine as long as the inputs do not have widely varying flux levels. In the equations below, `I` is the vector of input image values for a given pixel, `V` is the same for the input variance values, and `W = 1 / V`. (modifié)

stacking.txt

```
Default:
   image    = mean(I)
   variance = std(I)**2 / N

stats.setCalcErrorFromInputVariance(True)
   image    = mean(I)
   variance = mean(V) / N

stats.setCalcErrorMosaicMode(True)
   image    = mean(I)
   variance = mean(V)

Weighted:
   W = 1 / V
   image    = sqrt(sum(I**2 * W)) / sum(W)
   variance = (sum(I**2 * W) / sum(W) - image**2) * (1 / (sum(W)**2 / sum(W**2) - 1.0))

stats.setCalcErrorFromInputVariance(True)
   image    = sqrt(sum(I**2 * W)) / sum(W)
   variance = sum(I * W**2) * (N - 1)/N

stats.setCalcErrorMosaicMode(True)
   image    = sqrt(sum(I**2 * W)) / sum(W)
   variance = sum(I * W**2) * (N - 1)

```