How do I vary the hot score threshold with range?

(The cold score parameter is deprecated: you should always set it to the same value as the hot score threshold.)
There is no built-in way to vary the hot score threshold - a single value for each parameter is used in every sweep, and each sample is classified as "hot" or "cold" depending on whether its score is >= or < the hot score threshold.
However, you can simply write your own classification function to do things differently. This takes the form of a CLASSIFY hook function, which is called after radR does its usual classification of samples, and before they are joined into patches.

For example, the following script (attached below) reduces the hot score threshold by 0.5 z-units for every km of range:

```
## a classification threshold that varies the hot score threshold with ra
##
## You can save this file in the radR/scripts folder so that it can be ea
## sourced from the radR menu.
##
## Here, we use the GUI-defined hot score threshold, but reduce it linea
## a fixed amount per km of range.

threshold.decrease.per.km <- 0.5

## a hook function which will reclassify samples based on the changing ho

myclassify <- function(classmat, n) {
  ## create a hot score threshold vector (one entry for each sample slot
  ## You can move this calculation outside this function if you will not
  ## be changing threshold.decrease.per.km while radR runs.
  ## We do a global-level assignment (<<-) so the vector can be examined

  hst <<- as.integer(1024 * (RSS$blip.score.threshold[1] - (0:(RSS$scan.i
    threshold.decrease.per.km * RSS$scan.info$sample.dist / 1000))

  ## reset the class to COLD
  classmat[] <- as.integer(RSS$CLASS.VAL$cold)

  ## compare each sample's score to the vector of thresholds; "hot" is a
```

```
  hot <- RSS$score.mat[] >= hst
  classmat[hot] <- as.integer(RSS$CLASS.VAL$hot)
  nhot <- sum(hot)
  return(c(length(classmat) - nhot, nhot, 0))
}

## add this function to the CLASSIFY hook
rss.add.hook("CLASSIFY", myclassify)
```

There are a few key points:

- the score matrix is integer-valued; the low-order 10 bits of each score is fractional, so the true score is obtained by dividing the matrix values by 1024
- the classify hook function must return a vector of 3 integers:
  - number of cold samples
  - number of hot samples
  - 0 (an obsolete entry based on the deprecated cold score threshold)
- samples and scores are stored in pulse-major order, so each pulse's data are contiguous in memory
- the CLASSIFY hook functions are only called after learning has finished (otherwise the estimates for means and deviations of the stats cells would not exist, and no samples scores could be calculated)