**GGIR implementation guide:**

**Phase 1: Calibration, calculate ENMO and anglez**

Calibration:

We expect that for non-movement time the applied g-force acting on the sensor should be *1g.* Find non-movement time, compare data points during this time to unit sphere. Solve a fitting problem (apply scale and offset to data) to get those points as close as possible to unit sphere. Then apply this scale and offset to all data:

Where, si is the initial accelerometer data for each *i-th* dimension, ai is the scaling factor, bi is the offset factor, and is the calibrated accelerometer signal.

Summary process:

1. Find non-movement times: defined as when SD(acc[x,y,z])<0.013 over 10s window
   1. This is done by downsampling acc[x,y,z].
   2. Then SD over each window.
   3. Identify windows where SD, in all 3 axes, is < cutoff\_value (default 0.013). Keep only those windows.
      1. Also, “& npall( abs(roll\_mean(acc))<2)”, to ‘avoid clipping’ (removes with large mean no SD).
2. Find calibration error from only those windows (trim accel[x,y,z] to 10s windows that satisfy above criteria):
   1. Error defined as ; *norm(mean(acc)) -1*
   2. Iterative closest point fit until max\_iter or err<tol\_er
   3. Solve for the three scaling and offset values
   4. Check if new calibration error is < cal\_error\_start && 0.01 (apply offset + scale to windowed acc(x,y,z); (example from scikit)   
      *acc\_rm = (acc\_rm + offset) \* scale + tmp\_rm \* tmp\_scale*

*cal\_error\_end = around(mean(abs(norm(acc\_rm, axis=1) - 1)), decimals=5)*

*# assess if calibration error has been significantly improved*

*if (cal\_error\_end < cal\_error\_start) and (cal\_error\_end < 0.01):*

*return True, offset, scale, tmp\_scale, tmp\_mean*

*else:*

*return False, offset, scale, tmp\_scale, tmp\_mean*

* 1. Can include temperature data too (FUTURE)
  2. If successful calibration, apply calibration to raw accelerometer data. Return calibrated data, as well as scale, offset, and final cal\_error.

**References**

Source code : <https://rdrr.io/cran/GGIR/src/R/g.calibrate.R>

Python implementation from Pfizer : <https://github.com/pfizer-opensource/scikit-digital-health/blob/main/src/skdh/preprocessing/calibrate.py>

Paper:

<https://journals.physiology.org/doi/epdf/10.1152/japplphysiol.00421.2014>

**Metric calculations:**

Key physical activity metrics are calculated from the calibrated, raw accelerometer data. They are then downsampled (mean over the window length) over the three “epochs”. GGIR uses three window lengths, with some rules about length of each: shortest must be at least 1 second, the largest window must be a multiple of the second. Key metrics we will focus on in Phase 1, are ENMO (Euclidean norm, minus one) and anglez (angle relative to z-axis). These are calculated at the sample level and then a moving mean over the desired window lengths is applied. These metrics are used for all subsequent physical activity measures and sleep detection (in GGIR).

Key note about ENMO: default in GGIR and scikit is to trim any negative values to 0. Scikit has option to keep absolute values. Some questions about these metrics, mostly for the future:

1. Should we include abs(ENMO)?
2. Other desirable metrics, there is a long list included in GGIR, including filtering methods? are these used in literature?
3. GGIR calculates the anglez metric using a rolling median of acc(x,y,z). This rolling median is calculated over a window that is defined as: 5\* sampling\_rate +1, unless sampling\_rate is > 10, then it is hardcoded to 5\*10 +1. Scikit does not do this (and in my preliminary test I did not either and it didn’t seem to cause any issues). Not sure the need for this, especially since angle\_z is downsampled after being calculated.

**References**

GGIR source : <https://rdrr.io/cran/GGIR/src/R/g.applymetrics.R>

Python Scikit implementation : <https://github.com/pfizer-opensource/scikit-digital-health/blob/main/src/skdh/activity/metrics.py>