Supplementary Material for: "Continuous Glucose Deviation Interval and Variability Analysis (CG-DIVA): A Novel Approach for the Statistical Accuracy Assessment of Continuous Glucose Monitoring Systems" by

Eichenlaub, Stephan, Waldenmaier, Pleus, Rothenbühler, Haug, Hinzmann, Thomas, Jendle, Diem and Freckmann

Calculation of tolerance intervals

As mentioned in the main text, the tolerance intervals are calculated using a bias-corrected and accelerated (BCa) bootstrap method accounting for the clustered structure of the data.[21,22] This procedure is carried out as follows:

1. Generate a single bootstrap sample dataset comprised of a randomly sampled selection of sensors (with replacement) and their data containing the same number sensors as the original dataset. As the sampling is done with respect to the sensors and not the individual data points, the clustered structure of the data is accounted for.
2. Calculate the following quantiles of the deviations within the comparator glucose ranges with respect to the corresponding interval sizes.
    a. Comparator glucose <70 mg/dL (<3.9 mmol/L)
        i. Interval 1 (85%) [0.075,0.925]
        ii. Interval 2 (98%) [0.01,0.99]
    b. Comparator glucose 70-180 mg/dL (3.9-10.0 mmol/L)
        i. Interval 1 (70%) [0.15,0.85]
        ii. Interval 2 (99%) [0.005,0.995]
    c. Comparator glucose >180 mg/dL (>10.0 mmol/L)
        i. Interval 1 (80%) [0.1,0.9]
        ii. Interval 2 (99%) [0.005,0.995]
    d. Total comparator glucose
        i. Interval 1 (87%) [0.065,0.935]
3. Repeat steps 1 and 2 until a total of 10,000 bootstrap samples are generated.

4. Calculate the tolerance intervals with 95% confidence by determining the 0.025 quantile of the lower and the 0.975 quantile of the upper interval limits from the bootstrapped dataset using the BCa method. The BCa method adjusts these quantiles to account for any bias and yield a more reliable estimate of the 95 % confidence intervals.

This procedure was implemented in Python version 3.8 and R version 4.1.2. Using a standard PC (Intel® Core™ i7-7700 CPU @ 3.6 GHz, 8GB RAM) the processing time per CGM dataset was approximately 100 seconds in Python and 400 seconds in R.

The reproducibility at a number of bootstrap replications of 10,000 was validated by repeating the entire process 50 times with different seeds for the random generator.