Journal for the Measurement of Physical Behaviour, (Ahead of Print)
https://doi.org/10.1123/jmpb.2019-0063

Human Kinetics
ORIGINAL RESEARCH

Converting Raw Accelerometer Data to Activity Counts Using Open-Source Code: Implementing a MATLAB Code in Python and R, and Comparing the Results to ActiLife

Ruben Brondeel
Ghent University and Research Foundation—Flanders (FWO)

Yan Kestens
University of Montreal

Javad Rahimipour Anaraki
Memorial University of Newfoundland

Kevin Stanley
University of Saskatchewan

Benoit Thierry
University of Montreal

Daniel Fuller
Memorial University of Newfoundland

Background: Closed-source software for processing and analyzing accelerometer data provides little to no information about the algorithms used to transform acceleration data into physical activity indicators. Recently, an algorithm was developed in MATLAB that replicates the frequently used proprietary ActiLife activity counts. The aim of this software profile was (a) to translate the MATLAB algorithm into R and Python and (b) to test the accuracy of the algorithm on free-living data. ***Methods:*** As part of the INTErventions, Research, and Action in Cities Team, data were collected from 86 participants in Victoria (Canada). The participants were asked to wear an integrated global positioning system and accelerometer sensor (SenseDoc) for 10 days on the right hip. Raw accelerometer data were processed in ActiLife, MATLAB, R, and Python and compared using Pearson correlation, interclass correlation, and visual inspection. ***Results:*** Data were collected for a combined 749 valid days (>10 hr wear time). MATLAB, Python, and R counts per minute on the vertical axis had Pearson correlations with the ActiLife counts per minute of .998, .998, and .999, respectively. All three algorithms overestimated ActiLife counts per minute, some by up to 2.8%. ***Conclusions:*** A MATLAB algorithm for deriving ActiLife counts was implemented in R and Python. The different implementations provide similar results to ActiLife counts produced in the closed source software and can, for all practical purposes, be used interchangeably. This opens up possibilities to comparing studies using similar accelerometers from different suppliers, and to using free, open-source software.

Keywords: actiGraph, activity counts, INTERACT Study, objective physical activity, raw accelerometer data

Valid and reproducible measurements of physical activity are important for the surveillance of physical activity and for population health research (Hallal et al., 2012). Physical activity is commonly measured using accelerometer devices (Bassett, Troiano, Mcclain, & Wolff, 2015). Accelerometer devices provide measures of acceleration in three dimensions at a timescale below changes in human behavior (e.g., 0.01 s) for a relatively long, continuous observation period (e.g., 7 days; Plasqui & Westerterp, 2007). Acceleration data can be processed and used to derive a range of physical activity indicators (Bassett et al., 2015). Typical indicators of physical activity are the type of physical activity (e.g., walking), physical activity intensity levels (e.g., categorization in sedentary behavior, light, moderate, and vigorous physical activity per minute), or the accumulation of physical activity over time (e.g., total daily volume of physical activity; Bassett et al., 2015]).

Software programs are available for processing and analyzing acceleration data using different types of methods including Euclidean Norm Minus One (ENMO), Activity Index, and ActiGraph counts. For example, open-source software includes the R-package GGIR (van Hees et al., 2019) and OMGui (Open Lab at Newcastle University). The large number of different physical activity indicators make comparisons between studies difficult, if not impossible (Bai et al., 2016). ActiGraph accelerometers and software (ActiGraph LLC, Pensacola, FL) have been used extensively in health research (Bassett et al., 2015). Unfortunately, the dominance of ActiGraph devices has led to a dependence in health research on physical activity indicators calculated using the closed-source ActiLife activity counts measure and the ActiLife software. The dependence on closed-source methods and software is a problem for several reasons. First, researchers using devices other than the ActiGraph cannot easily compare their results to a large body of work that is based on ActiGraph accelerometers. This is unfortunate, as other devices can be advantageous for health research because they can be cheaper, smaller, easier to wear, and/or based on open-source principles. Second, researchers cannot improve on the activity counts algorithm. The dependency on close-source software therefore limits scientific knowledge.

In response, Brønd, Andersen, and Arvidsson (2017) have developed an open-source algorithm for calculating activity counts that replicate proprietary ActiLife activity counts and have made the algorithm freely available (Brønd). Although some small differences in raw accelerometer data have been found between

Brondeel is with the Department of Movement and Sports Sciences, Ghent University, Gent, Belgium; and the Research Foundation—Flanders (FWO), Brussels, Belgium. Kestens and Thierry are with the University of Montreal, Montreal, Canada. Anaraki and Fuller are with the Memorial University of Newfoundland, St. John's, Canada. Stanley is with the University of Saskatchewan, Saskatoon, Canada. Brondeel (ruben.brondeel@ugent.be) is corresponding author.

different brands of devices (Hildebrand, Van Hees, Hansen, & Ekelund, 2014; Rowlands & Stiles, 2012; Rowlands et al., 2015); most accelerometer devices on the market include comparable accelerometer integrated circuits. Therefore, Brønd's algorithm should, in theory, allow researchers to calculate activity counts comparable to those obtained by ActiLife, independent of the device. A drawback of the Brønd algorithm, is its implementation in MATLAB (MathWorks, Natick, MA), a commercial signal analysis software program, which is not open-source and not free to use. One option to apply the Brønd algorithm is to use Octave (Eaton, Bateman, & Hauberg, 2013), an open-source software package that interprets code in the same way as MATLAB. However, as few health researchers use Octave, the aim in this article was to provide implementation of the Brønd algorithm in commonly used free, open-source programming languages R (R Core Team; Vienna, Austria) and Python (Van Rossum & Drake, 1995).

In this software profile, the different stages of the Brønd algorithm are presented. Second, results testing the MATLAB, R, and Python activity counts in comparison to the ActiLife counts on free-living data collected in Victoria, Canada, are presented. We compared activity counts per second, and activity counts summed per minute and per day. We further compared the activity counts summed per minute, but now within three categories according to their estimated activity intensity; that is, we compared counts per minute (CPM) for minutes with sedentary behavior (SB), for minutes with light physical activity (LPA), and for minutes with moderate-to-vigorous physical activity (MVPA), separately. In a final comparison, we examine the detected time spent in the three levels of activity intensity levels and the non-wear time. Finally, the results are discussed, and strengths and limitations were acknowledged. It should be noted that the objective of this software profile was not to promote ActiLife activity counts as the best way to measure physical activity; but rather to develop and publish a new tool to compare results between different types of accelerometers and the large body of previous ActiGraph-/ActiLife-based studies. The R algorithm is published as an R package (Brondeel et al., 2019) under a GNU General Public License and with ongoing support by the authors (see R package for more details); the R and Python implementations can be found freely online, alongside the original MATLAB implementation (Brønd; INTERACT, 2019).

Short Overview of the Brønd Algorithm

The algorithm to calculate activity counts includes eight steps, previously described in more detail (Brønd et al., 2017). However, if data are collected at a higher sampling frequency, a preliminary step is to resample the data to 30 Hz. First, on the 30 Hz data, a band-pass filter is applied, filtering out the lowest and highest frequencies in the signal. Second, a standard transfer filter is applied, using the filter coefficients provided by Brønd et al (Brønd, 2019). Third, the data are resampled to 10 Hz, by selecting every third observation. Fourth, the data are leveled off to a constant (2.13); that is, values higher than this threshold were replaced by this value. In steps five and six, the absolute values are taken, and the lowest values are replaced by a constant (0.068). In the seventh step, the data are reduced to the 8-bit analog-to-digital converter resolution, which means the values are floored (i.e., rounded to a lowest of predefined values) to one of 128 values evenly spaced between 0 and 2.13. Finally, the 10 values per second are summed, resulting in a measure of activity counts per second.

Comparing the Three Implementations of the Algorithm to the ActiLife Algorithm

Raw Accelerometer Data

To compare the MATLAB, R, and Python implementations of the ActiLife activity counts algorithm to the ActiLife algorithm, accelerometer data were used from 86 participants recruited for the INTERventions, Research, and Action in Cities Team study (Kestens et al., 2019). Participants were asked to wear a SenseDoc device for 10 days on the right hip from waking to bedtime. The SenseDoc device is an integrated global positioning system and accelerometer sensor that has been used in previous research (Kestens et al., 2016). It has similar accelerometer technology to the ActiGraph GT3X. SenseDoc devices include ADXL345 accelerometers, while ActiGraph GT3X devices use ADXL335 accelerometers, both produced by Analog Devices, Inc (Norwood, MA). Note, it is not known to us which internal circuit current models of ActiGraph devices use. As the same algorithm is used to calculate activity counts in ActiLife, we presume that the raw accelerometer data are similar to the older GT3X device. Following settings in SenseAnalytics (version 1.7; Montreal, Canada) were used to initialize the SenseDoc devices: a measurement range of ±8 and a sampling rate of 50 Hz. The SenseDoc (MobiSens, http://mobysens.com/) has a sensitivity of 4 mg/least significant bit.

Data Preprocessing

Raw accelerometer data in *g* units were extracted with the SenseAnalytics software from SenseDoc devices, and stored in.csv files. MATLAB, R, and Python codes worked directly with the.csv files. For ActiLife, the raw accelerometer data were first converted, using Python, to ActiGraph binary files (.gt3x). The code to convert.csv-files to.gt3x-files is available upon request.

Data Processing

The raw data (.gt3x files) were processed using ActiLife to obtain ActiGraph counts per second on one side. Brønd's MATLAB and our corresponding Python and R translations were applied to the raw data (.csv files) to obtain comparable, but open-source activity counts per second. The parameters (e.g., filter settings) are included in the codes (Brønd; INTERACT, 2019) and are the same for each programming language. However, the filters are implemented slightly differently in each language, which may create differences in the results.

Data Postprocessing

The activity counts from each of the four different processing methods on the raw data (ActiLife, MATLAB, R, and Python) were transformed into the following indicators using R (version 3.5.0, 64 bit; R core team, Auckland, New Zealand). Counts per second were aggregated to CPM and counts per day by summing the counts for these time units and this for each of the three axes. Physical activity intensity levels were calculated using vertical axis CPM with the following ranges: sedentary behavior (CPM 0–99), LPA (CPM 100–2,019), and moderate-to-vigorous physical activity (CPM ≥ 2,020) (Troiano et al., 2008). Nonwear was detected with the "PhysicalActivity" package, using the Choi algorithm (Choi, Cole, Liu, Matthews, & Buchowski, 2018) on vertical axis CPM (Choi, Ward, Schnelle, & Buchowski, 2012).

Analysis

ActiLife activity counts for the total wear time were compared with MATLAB, Python, and R activity counts at the second, minute, and day level. CPM, counts per day, and counts per second were compared with different agreement measures, including intra class correlations (calculated in R with "irr" package and settings: "one-way model" [row effects random]) and Pearson correlation coefficients. Bias was measured using mean difference, mean absolute difference, and mean percentage absolute difference.

The CPM were compared within physical activity intensity categories (SB, LPA, and MVPA), as detected with ActiLife counts. The comparison of CPM within intensity categories was necessary because agreement and bias measures of the overall CPM could have been influenced by the high presence of zero CPM values. Finally, ActiLife was compared with the three algorithms on its categorization of CPM into nonwear and the three activity intensity levels, using a cross-tabulation. Due to the high correspondence, no diagnostic testing, such as sensitivity and specificity, were calculated.

Results

A total of 86 people who used a bicycle at least once per month provided data for this study, 52% were female, and the median age was 42 years (interquartile range = [33; 55]). The data included 749 valid days (at least 10-hr wear time) and 649,301 observed minutes of data.

Table 1 presents the bias and agreement measures for counts per day, CPM, and counts per second on the vertical axis; comparing MATLAB, Python, and R on one hand, and ActiLife counts on the other hand. Pearson correlations and intraclass correlation coefficients were all above .985 for counts per second, exceeded .998 for CPM, and were greater than .998 for counts per day.

The CPM on the vertical axis is commonly used to detect nonwear and to determine activity intensity levels. The vertical CPM measures were further investigated. Figure 1 represents the associations for CPM using scatter plots. Visual inspection of Figure 1 shows the strong, linear correlations of MATLAB, Python, and R CPM with ActiLife CPM. Many dots on the plot are on the diagonal line or very close. In general, R, MATLAB, and Python CPM were slightly higher than ActiLife CPM. The largest deviations can be seen for a few observations, which underestimate ActiLife CPM. It should be noted, however, that the scatter plots include a preponderance of overlapping points, and the points at the extremes represent a small, nearly insubstantial portion of the total number of observations (total $n = 649{,}301$). Note, for the same reason Bland–Altman plots were not presented. The general overestimation of CPM by the three new algorithms compared with ActiLife can also be noted in Table 1. The mean differences between ActiLife CPM and MATLAB, R, and Python CPM (biases) are 4.58, 6.96, and 10.8 CPM, respectively. This corresponds to 1.2%, 1.8%, and 2.8% of the mean ActiLife CPM during wear time.

Table 2 presents the Pearson correlations between vertical axis CPM within three intensity levels of physical activity, SB, LPA, and MVPA. The intraclass correlation coefficient and correlations between Python, R, and MATLAB CPM compared with ActiLife CPM all exceeded .975. The lowest agreement was found for counts during SB. This might be explained by the fact that the algorithm has been optimized in lab settings, where there are typically higher levels of activity. However, further investigation is required to confirm this hypothesis and not within the scope of this paper. The R CPM had the highest agreement with the ActiLife CPM, compared with MATLAB and Python CPM. R CPM also resulted in the highest bias measurements of the three algorithms, with a mean deviation from ActiLife CPM of 1.6%, 1.9%, and 2.5% for CPM during SB, LPA, and MVPA, respectively. The mean deviation from ActiLife CPM during SB, LPA, and MVPA was 1.1%, 0.9%, and 1.6% for MATLAB and 0.8%, 1.8%, and 1.9% for Python.

Finally, the algorithms were compared with respect to detecting activity intensity levels. The cut points presented above were applied to the CPM obtained from ActiLife and the MATLAB, R, and Python implementations of the algorithm. Using ActiLife counts, 46% of observations were detected as nonwear. During wear time, the participants performed SB for 63% of the time (546 min/day on average), LPA 32% of the time (278 min/day on average), and MVPA 5% of the time (43 min/day on average). Table 3 shows a high correspondence (99% or higher) between the SB and MVPA detected using ActiLife counts and, R, MATLAB, and Python counts. For LPA, the algorithms corresponded highly with ActiLife (97% or higher). Finally, the detection of nonwear was independent of the algorithms, with algorithms corresponding 100%.

Discussion

Recently, Brønd and colleagues published an open-source method to calculate ActiLife counts in MATLAB (Brønd et al., 2017). The purpose of this article was to present implementations of the MATLAB code in R and Python, two free and widely used statistical programs. The accuracy of the algorithms in replicating ActiLife counts was tested in free-living conditions. It should be noted that the aim was not to promote activity counts as the most accurate way to capture bodily movement from accelerometer data, but rather provide a tool to compare results from new studies to older studies using ActiLife counts.

Based on indicators of bias and covariance, all algorithms provided activity counts that were similar to ActiLife counts. In all algorithms, there was a slight tendency to overestimate ActiLife counts. The data processing from raw data to activity counts can therefore be conducted in any of the programming languages reliably and consistently. The derived physical activity intensity indicators SB, LPA, and MVPA were similar when comparing the three new algorithms to ActiLife counts. Decisions about health outcomes would be unaffected by the choice of programming language. Being able to apply this algorithm to raw accelerometer data gives researchers an extra tool to compare their data to previously collected ActiGraph data, especially when the raw data from older studies may not be readily available. As the majority of accelerometers allow raw data collection in *g* units, this algorithm could be applied to newly collected data. Differences in device design, in the wear of the device (e.g., type of attachment to the body), sample rates, and study protocols, will still need to be taken into consideration when performing such comparisons. Finally, readers should note that older ActiGraph devices (e.g., ActiGraph 7164) directly produced counts, instead of providing raw accelerometer data that were then transformed to counts using the ActiLife software. These device-based counts are not necessarily the same as ActiLife activity counts (Cain, Conway, Adams, Husak, & Sallis, 2013). Comparison between our newly implemented methods and counts from these older devices should be done cautiously.

Table 1 Comparing ActiLife With MATLAB, Python, and R Vector Magnitude Counts per Day, per Minute, and per Second During Wear Time

	ActiLife counts per day	ActiLife counts per minute	ActiLife counts per second
Agreement			
Pearson correlations			
MATLAB	.998	.998	.987
Python	.998	.998	.986
R	.999	.998	.986
ICC			
MATLAB	.998	.998	.987
Python	.998	.998	.986
R	.998	.998	.986
Bias			
Means			
ActiLife	336,679 [94,054; 951,386]	388 [0; 3,314]	6.473 [0; 71]
MATLAB	342,798 [95,888; 947,140]	393 [0; 3,417]	6.551 [0; 73]
Python	342,798 [96,342; 952,321]	395 [0; 3,430]	6.591 [0; 74]
R	345,971 [97,677; 958,575]	399 [0; 3,440]	6.652 [0; 74]
Mean difference [quantiles: 2.5; 97.5]			
MATLAB	4,053 [–9,164; 19,229]	4.675 [–41; 114]	0.078 [–4; 6]
Python	6,119 [–7,923; 23,673]	7.059 [–35; 126]	0.118 [–4; 6]
R	9,292 [702; 26,184]	10.719 [–27; 135]	0.179 [–5; 7]
Mean absolute difference [quantiles: 2.5; 97.5]			
MATLAB	7,331 [402; 27,510]	16.369 [0; 125]	0.809 [0; 8]
Python	9,009 [850; 29,624]	17.442 [0; 136]	0.814 [0; 8]
R	10,430 [1,628; 28,251]	17.888 [0; 140]	0.919 [0; 9]
Mean percentage absolute difference [quantiles: 2.5; 97.5]			
MATLAB	2.043 [0.186; 3.969]	7.087 [0.000; 60.000]	5.128 [0.000; 60.000]
Python	2.649 [0.341; 4.625]	7.298 [0.000; 63.612]	5.214 [0.000; 60.000]
R	3.164 [0.989; 4.861]	7.598 [0.000; 60.000]	5.885 [0.000; 69.231]
N	749	649,301	38,958,061

Note. Bias = differences in means between MATLAB, Python or R counts, and ActiLife counts. *N* days = 749; *n* minutes (wear time only) = 649,301; *n* seconds = 38,958,060 (*n* minutes × 60). ICC = intraclass correlation coefficient.

The different implementations do not result in exactly the same CPM, even though the overall steps in the algorithm are identical. The reason is the implementation of the signal processing filter and resampling functions, which are used to filter nonhuman acceleration from the raw data. The original MATLAB code—like the R and Python codes—uses a resampling function (in case the sample rate is not 30 Hz) and two signal filters successively, an aliasing filter and a frequency band-pass filter (Brønd et al., 2017). Unfortunately, these functions are implemented slightly differently in the three programming languages. Testing the differences between

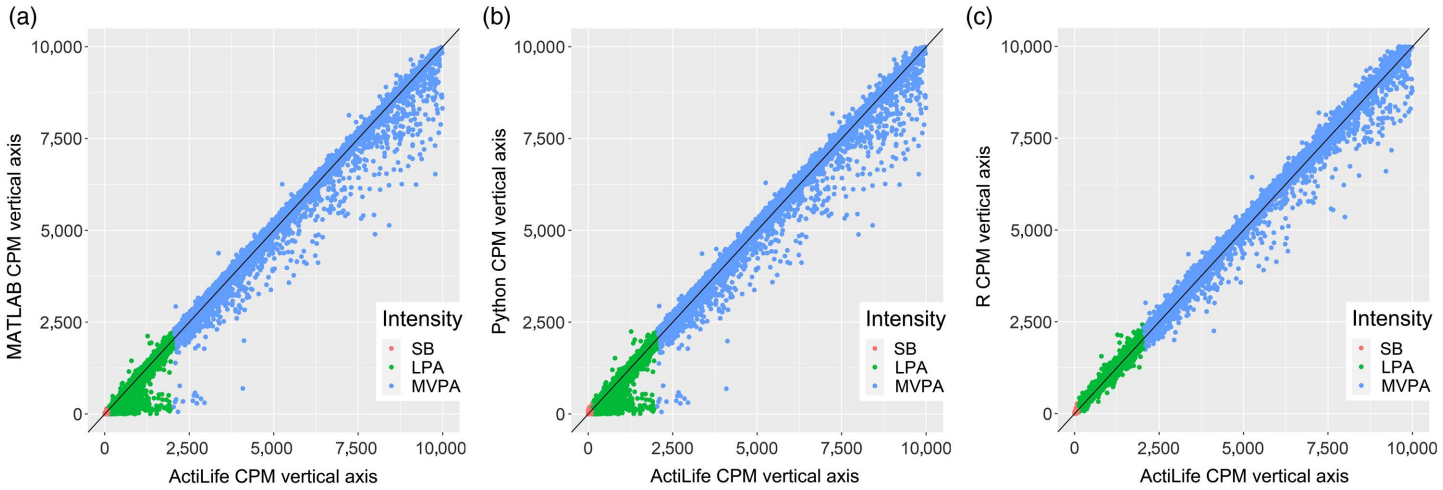

Figure 1 — ActiLife vertical axis CPM versus MATLAB (a), Python (b) and R (c) vertical axis CPM. CPM= counts per minute; SB = sedentary behavior (ActiLife CPM 0–99); LPA = light physical activity (ActiLife CPM 100–2,019); MVPA = moderate-to-vigorous physical activity (ActiLife CPM ≥ 2,020).

Table 2 Comparing ActiLife With MATLAB, Python, and R Vector Magnitude CPM During SB, LPA, and MVPA

	ActiLife CPM during SB	ActiLife CPM during LPA	ActiLife CPM during MVPA
Agreement			
Pearson correlations			
MATLAB	.985	.996	.993
Python	.978	.996	.993
R	.985	.998	.995
ICC			
MATLAB	.985	.995	.992
Python	.978	.995	.992
R	.984	.996	.994
Bias			
Means [quantiles: 2.5; 97.5]			
ActiLife	10.187 [0; 82]	577.317 [109; 1,779]	3,956.053 [2,061; 9,536]
MATLAB	10.076 [0; 82]	582.397 [105; 1,825]	4,018.673 [2,107; 9,263]
Python	10.106 [0; 83]	587.746 [106; 1,832]	4,031.736 [2,117; 9,270]
R	10.343 [0; 84]	595.108 [109; 1,848]	4,054.868 [2,128; 9,452]
Mean difference [quantiles: 2.5; 97.5]			
MATLAB	–0.111 [–8; 7]	5.080 [–70; 83]	62.620 [–328; 220]
Python	–0.081 [–8; 7]	10.429 [–60; 96]	75.684 [–324; 241]
R	0.156 [–8; 9]	17.791 [–45; 109]	98.815 [–177; 283]

(continued)

Table 2 *(continued)*

	ActiLife CPM during SB	ActiLife CPM during LPA	ActiLife CPM during MVPA
Mean absolute difference [quantiles: 2.5; 97.5]			
MATLAB	1.394 [0; 11]	26.168 [1; 96]	142.711 [9; 353]
Python	1.469 [0; 11]	27.727 [1; 107]	153.269 [11; 363]
R	1.471 [0; 12]	29.451 [1; 111]	151.106 [11; 349]
Mean percentage absolute difference [quantiles: 2.5; 97.5]			
MATLAB	8.071 [0; 100]	5.692 [0.173; 20.225]	3.612 [0.240; 8.417]
Python	8.195 [0; 100]	6.061 [0.192; 24.432]	3.906 [0.271; 8.872]
R	8.650 [0; 100]	6.099 [0.226; 19.205]	3.950 [0.303; 8.998]
N	408,861	208,121	32,319

Note. *n* SB = 433,020; *n* LPA = 219,918; *n* MVPA = 34,169. Intensity levels were determined on the vertical axis. CPM = counts per minute; SB = sedentary behavior; LPA = light physical activity; MVPA = moderate-to-vigorous physical activity; ICC = intraclass correlation coefficient.

Table 3 Percentage of Minutes Detected to the Same Category Using Activity Counts From the MATLAB, R, and Python Implementations, Compared With ActiLife Counts

ActiLife	MATLAB (%)	R (%)	Python (%)
Non-wear (*n* = 586,033)	100.0	100.0	100.0
SB (*n* = 433,020)	99.4	99.2	99.3
LPA (*n* = 219,918)	97.7	98.2	97.2
MVPA (*n* = 34,169)	99.4	99.5	99.7

Note. SB = sedentary behavior; LPA = light physical activity; MVPA = moderate-to-vigorous physical activity.

the MATLAB code and R code step-by-step, the major difference was in the resampling step. Where MATLAB uses an antialiasing low-pass filter in the resampling function (Center, 2020), the R code uses a simple nearest neighbor technique (Signal developers, 2014). Currently, the exact copy of the MATLAB resample function has not yet been implemented in R. In Python, there is a filter included in the resample function, comparable to—but not exactly the same as—the filter included in the MATLAB resample function. Comparing the MATLAB to the Python results step-by-step, small differences were found in both the resampling and the filtering; even though the differences were smaller compared with the differences between MATLAB and R. MATLAB is closed-source software, as a result it was not possible to find a perfect solution for the differences in implementation. Given the small average differences in the activity counts calculated with the three different programming languages, these languages can, for all practical purposes, be used interchangeably.

A strength of the comparisons provided in this article is the use of data obtained from free-living adults. The data represent measures of movement during a large range of physical behavior. One disadvantage of using this data is the use of a single device in combination with a fixed set of device settings. For example, data obtained at different sampling frequencies might result in activity counts that are more or less divergent between the programming languages. This is because ActiLife handles the filtering of accelerometer data differently depending on the sampling frequencies (Brønd & Arvidsson, 2016; Clevenger et al., 2019). The influence of this issue on our results is likely small given that these high frequencies are typically registered at high-speed running (or similar behavior), which is not common behavior in this data set. However, it might be a more important issue in other populations (e.g., observations of professional athletes during training sessions).

Conclusions

A method recently developed in MATLAB to calculate ActiLife counts was successfully implemented in R and Python, two open-source programming languages. Free, open-source software and our newly published method to replicate ActiLife activity counts in MATLAB, R, and Python can be applied on data obtained from different accelerometer brands and may allow researchers to reduce the costs of accelerometer studies, while still being able to compare their results to previously obtained results using proprietary ActiLife activity count measures. The authors do not promote the use of activity counts per se, as other physical activity indicators derived from accelerometer data might be better able to characterize bodily movement. The R package and Python code do provide an extra tool for comparing study results.

References

Bai, J., Di, C., Xiao, L., Evenson, K.R., LaCroix, A.Z., Crainiceanu, C.M., & Buchner, D.M. (2016). An activity index for raw accelerometry data and its comparison with other activity metrics. *PLoS One, 11*(8), e0160644. PubMed ID: 27513333 doi:10.1371/journal.pone.0160644

Bassett, D.R., Troiano, R.P., Mcclain, J.J., & Wolff, D.L. (2015). Accelerometer-based physical activity: Total volume per day and standardized measures. *Medicine & Science in Sports & Exercise, 47*(4), 833–838. PubMed ID: 25102292 doi:10.1249/MSS.0000000000000468

Brønd, J.C. (2019, June 7, 2019). ActiGraph counts. Retrieved from https://github.com/jbrond/ActigraphCounts

Brønd, J.C., Andersen, L.B., & Arvidsson, D. (2017). Generating ActiGraph counts from raw acceleration recorded by an alternative monitor. *Medicine & Science in Sports & Exercise, 49*(11), 2351–2360. PubMed ID: 28604558 doi:10.1249/MSS.0000000000001344

Brønd, J.C., & Arvidsson, D. (2016). Sampling frequency affects the processing of Actigraph raw acceleration data to activity counts. *Journal of Applied Physiology, 120*(3), 362–369. doi:10.1152/japplphysiol.00628.2015

Brondeel, R., Rahimipour Anaraki, J., Khataei Pour, S., & Fuller, D. (2019). activityCounts: Generating ActiLife Counts. R package version 0.1.2. https://CRAN.R-project.org/package=activityCounts

Cain, K.L., Conway, T.L., Adams, M.A., Husak, L.E., & Sallis, J.F. (2013). Comparison of older and newer generations of ActiGraph accelerometers with the normal filter and the low frequency extension. *International Journal of Behavioral Nutrition and Physical Activity, 10*(1), 51. doi:10.1186/1479-5868-10-51

Center, M.-H. (2020). Resample function—Resample uniform or nonuniform data to new fixed rate. Retrieved from https://nl.mathworks.com/help/signal/ref/resample.html

Choi, L., Ward, S.C., Schnelle, J.F., & Buchowski, M.S. (2012). Assessment of wear/nonwear time classification algorithms for triaxial accelerometer. *Medicine & Science in Sports & Exercise, 44*(10), 2009–2016. PubMed ID: 22525772 doi:10.1249/MSS.0b013e318258cb36

Choi, L.a., Cole, B., Liu, Z., Matthews, C.E., & Buchowski, M.S. (2018). PhysicalActivity: Process accelerometer data for physical activity measurement. R package version 0.2-2. Retrieved from https://CRAN.R-project.org/package=PhysicalActivity

Clevenger, K.A., Pfeiffer, K.A., Mackintosh, K.A., McNarry, M.A., Brønd, J., Arvidsson, D., & Montoye, A.H. (2019). Effect of sampling rate on acceleration and counts of hip-and wrist-worn ActiGraph accelerometers in children. *Physiological Measurement, 40*(9), 095008. PubMed ID: 31518999 doi:10.1088/1361-6579/ab444b

Eaton, J.W., Bateman, D., & Hauberg, S. (2013). GNU octave - A high-level interactive language for numerical computations. *GNU Octave*. Retrieved from https://octave.org/octave.pdf

Hallal, P.C., Andersen, L.B., Bull, F.C., Guthold, R., Haskell, W., Ekelund, U., & Workin, L.P.A.S. (2012). Global physical activity levels: Surveillance progress, pitfalls, and prospects. *The Lancet, 380*(9838), 247–257. doi:10.1016/S0140-6736(12)60646-1

Hildebrand, M., Van Hees, V.T., Hansen, B.H., & Ekelund, U. (2014). Age group comparability of raw accelerometer output from wrist-and hip-worn monitors. *Medicine & Science in Sports & Exercise, 46*(9), 1816–1824. PubMed ID: 24887173 doi:10.1249/MSS.0000000000000289

INTERACT. (2019). *Activity count algorithms*. Retrieved from https://github.com/TeamINTERACT/publications/tree/master/count_algorithm_2019_Brondeel

Kestens, Y., Chaix, B., Gerber, P., Despres, M., Gauvin, L., Klein, O., … Wasfi, R. (2016). Understanding the role of contrasting urban contexts in healthy aging: An international cohort study using wearable sensor devices (the CURHA study protocol). *BMC Geriatrics, 16*(1), 1–12.

Kestens, Y., Winters, M., Fuller, D., Bell, S., Berscheid, J., Brondeel, R., … Wasfi, R. (2019). INTERACT: A comprehensive approach to assess urban form interventions through natural experiments. *BMC Public Health, 19*(1), 1–11. doi:10.1186/s12889-018-6339-z

Open Lab at Newcastle University. *AX3 OMGUI configuration and analysis tool (version V43)*. Retrieved from https://github.com/digitalinteraction/openmovement/wiki/AX3-GUI

Plasqui, G., & Westerterp, K.R. (2007). Physical activity assessment with accelerometers: An evaluation against doubly labeled water. *Obesity, 15*(10), 2371–2379. PubMed ID: 17925461 doi:10.1038/oby.2007.281

Rowlands, A., & Stiles, V. (2012). Accelerometer counts and raw acceleration output in relation to mechanical loading. *Journal of Biomechanics, 45*(3), 448–454. PubMed ID: 22218284 doi:10.1016/j.jbiomech.2011.12.006

Rowlands, A.V., Fraysse, F., Catt, M., Stiles, V.H., Stanley, R.M., Eston, R.G., & Olds, T.S. (2015). Comparability of measured acceleration from accelerometry-based activity monitors. *Medicine & Science in Sports & Exercise, 47*(1), 201–210. PubMed ID: 24870577 doi:10.1249/MSS.0000000000000394

Signal developers. (2014). *signal: Signal processing*. Retrieved from http://r-forge.r-project.org/projects/signal/

Troiano, R.P., Berrigan, D., Dodd, K.W., Masse, L.C., Tilert, T., & McDowell, M. (2008). Physical activity in the United States measured by accelerometer. *Medicine & Science in Sports & Exercise, 40*(1), 181–188. PubMed ID: 18091006 doi:10.1249/mss.0b013e31815a51b3

van Hees, V.a., Zhou F., Mirkes, E., Heywood, J., Zhao, J.H., Joan, C.P., Sabia, S., & Migueles, J.H. (2019). *GGIR: Raw Accelerometer Data Analysis (Version 1.10-7): Zenodo*.

Van Rossum, G., & Drake, F.L., Jr. (1995). *Python reference manual*. Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.