An Accelerometer-Based Intake-Balance Method for Assessing Energy Intake in Time Restricted Eating Trials

Paul R. Hibbing

This vignette will walk you through sample code to execute our accelerometer-based intake balance method. It's interspersed with commentary that will hopefully make it easier to digest. You can also view a web-based analog of this vignette where there are further instructions for getting help etc.

Prerequisites

Before moving forward with any other code, you need to (once only) take care of the following to get set up:

1. Install R (required) and RStudio (optional)
2. Windows users, install Rtools
3. Make sure you have the necessary R packages installed. To do that, open up R, paste the following code into the console, and hit **enter** to execute. Fair warning, this could take awhile to finish running if you're using R for the first time or otherwise don't have all of these packages installed.

```
dependencies <- c(
  "data.table", "dplyr", "equivalence", "ggplot2", "gsignal",
  "lazyeval", "lubridate", "magrittr", "PhysicalActivity", "R.utils",
  "Rcpp", "remotes", "reshape2", "rlang", "tools", "zoo"
)

sapply(
  dependencies,
  function(x) if (!x %in% installed.packages()) install.packages(x)
)

if (!"read.gt3x" %in% installed.packages()) remotes::install_github(
  "THLfi/read.gt3x", dependencies = FALSE
)

if (!"agcounts" %in% installed.packages()) remotes::install_github(
  "paulhibbing/agcounts", dependencies = FALSE
)

if (!"PAutilities" %in% installed.packages()) remotes::install_github(
  "paulhibbing/PAutilities", dependencies = FALSE
)
```

Now that you're set up

The rest of this vignette provides step-by-step code to implement the method. It is designed to let you follow along with the example of processing one file. In practice, you would likely want to set this up for batch processing multiple files. That's beyond our scope here, but not by much. The main thing you would have to do in that case is move this code into a *function* or a *for loop*, depending on your preference. For some helpful starting points, check out some information on list.files, for loops, and saveRDS.

Step 1: Reading in your data

The first step of the method requires you to read in some data, in both "raw acceleration" and "activity count" format. Below, we'll use a built in sample file (part of the read.gt3x package) to show how this can be done.

```
# Attach the magrittr package to gain
# access to `%>%` and similar operators
suppressPackageStartupMessages(library(magrittr))

# This will retrieve the existing sample file
sample_file <- system.file(
  "extdata/TAS1H30182785_2019-09-17.gt3x",
  package = "read.gt3x"
)

# Your own file might look like this:
# my_file <- "C:/users/myusername/Desktop/myfile.gt3x"

# Read the raw acceleration data (30+ Hz) and make sure timestamps
# are in UTC timezone -- Store this in an object called `accel`
accel <-
  read.gt3x::read.gt3x(sample_file, FALSE, TRUE, TRUE) %>%
  dplyr::mutate(time = lubridate::force_tz(time, "UTC"))

# Convert to activity counts (60-s epochs) and make sure timestamps
# are in UTC timezone; store this in a separate object called `AG`
# (The call to `slice` is needed because `calculate_counts` adds zeroes to the
# end of the file based on when the monitor was downlodaed, whereas `read.gt3x`
# does not)
AG <-
  agcounts::calculate_counts(accel, 60, tz = "UTC") %>%
  dplyr::slice(which(time <= dplyr::last(accel$time))) %>%
  dplyr::mutate(filename = basename(sample_file)) %>%
  dplyr::relocate(filename)
```

Step 2: Calculate minute-by-minute energy expenditure

This step breaks down into the following:

1. Calculate Euclidian Norm Minus One (ENMO) for each raw acceleration sample, rounding values < 0 up to 0
2. Calculate mean ENMO each second

3. Apply the Hildebrand non-linear oxygen consumption (VO2) equation presented by Ellingson et al. (2017)
4. Calculate the mean VO2 each minute
5. Convert the VO2 estimates (ml/kg/min) to caloric expenditure (kcal/kg/min)

Here's how we can accomplish all that:

```
accel %<>%
  # Calculate ENMO, in milli-G
    dplyr::mutate(
      ENMO = {sqrt(X^2 + Y^2 + Z^2) - 1} %>% pmax(0) %>% {. * 1000}
    ) %>%
  # Average each second
    dplyr::group_by(time = lubridate::floor_date(time, "1 second")) %>%
    dplyr::summarise(dplyr::across(.fns = mean), .groups = "drop") %>%
  # Calculate VO2
    dplyr::mutate(VO2 = {ENMO ^ .534} %>% {0.901 * .} %>% pmax(3, .)) %>%
  # Average each minute
    dplyr::group_by(time = lubridate::floor_date(time, "1 minute")) %>%
    dplyr::summarise(dplyr::across(.fns = mean), .groups = "drop") %>%
  # Convert to kcal/kg/min
    dplyr::mutate(
      kcal_kg_min =
        (VO2 / 1000) *
        PAutilities::get_kcal_vo2_conversion(RER = 0.85, kcal_table = "Lusk")
    ) %>%
  # Retain only the relevant variables
    dplyr::select(time, kcal_kg_min)
```

Step 3: Calculate minute-by-minute non-wear

This is where the previously-calculated activity count data come into play. Here's how we calculate non-wear using the **PhysicalActivity** package (Choi et al. non-wear method).

```
AG %<>%
  PhysicalActivity::wearingMarking(
    TS = "time", cts = "Axis1", perMinuteCts = 1, tz = "UTC"
  ) %>%
  dplyr::mutate(is_nonwear = !wearing %in% "w") %>%
  dplyr::select(-c(wearing, weekday, days))
```

Step 4: Combine the energy expenditure and non-wear data

This step can be accomplished with a pretty straightforward merge.

```
AG %<>% merge(accel)
rm(accel)
```

Step 5: Calculate daily totals

Now it is time to calculate total energy expenditure (and acceleration metrics) for each day of data. When summing these variables, it's crucial to exclude data from non-wear periods. To do that, we can replace the

values with 0 during non-wear, so that the sum comes out correctly. We will also summarize the total number of non-wear minutes, both for screening purposes (e.g., throwing out days with 14+ hours of non-wear) and for imputation of basal metabolic rate for all the non-wear minutes.

```
AG %<>%
  dplyr::group_by(filename, time = as.Date(time)) %>%
  dplyr::mutate(dplyr::across(
    !dplyr::all_of("is_nonwear"),
    .fns = ~ ifelse(is_nonwear | is.na(.x), 0, .x)
  )) %>%
  dplyr::summarise(
    dplyr::across(.fns = sum),
    total_mins = dplyr::n(),
    .groups = "drop"
  ) %>%
  dplyr::rename(nonwear_mins = is_nonwear, date = time, kcal_kg = kcal_kg_min) %>%
  dplyr::relocate(filename, date, total_mins, nonwear_mins)
```

Step 6: Apply compliance checks and filter out non-compliant data

In accelerometry, a rule of thumb for assessing compliance is to filter out any day with < 10 hours of wear time, and to filter out any participant who does not have at least four days meeting that criterion. However, these decisions are study-specific, and for this illustration we are only dealing with a small amount of data. So, here we will require at least one day of data with < 10 minutes of non-wear. The code will produce an empty result for non-compliant participants. For batch processing, you would want to check if the result is empty and, if so, cut off the processing from there (e.g., by **return(AG)** inside a function, or **next** inside a for loop).

```
maximum_nonwear_mins <- 10
minimum_days <- 1

AG %<>%
  dplyr::group_by(filename) %>%
  dplyr::mutate(
    day_compliant = nonwear_mins < maximum_nonwear_mins,
    participant_compliant = sum(day_compliant) >= minimum_days
  ) %>%
  dplyr::filter(day_compliant & participant_compliant)
```

Step 7: Determine total daily energy expenditure

Here we need to incorporate the participant's body mass (to convert kcal/kg to kcal), along with other anthropometric/demographic variables to determine estimated basal metabolic rate (BMR) from Schofield's equations. Then we need to impute energy expenditure for non-wear periods. This can all be done as follows:

```
## Suppose demographic information is available in a data frame like this
demo <- data.frame(
  filename = basename(sample_file), sex = "M",
  ht_m = 1.80, wt_kg = 75, age = 30
)
```

```r
## Then we can predict BMR like this:
demo$basal_kcal_day <- PAutilities::get_ree(
  df = demo, method = "schofield_wt_ht", sex = "sex", age_yr = "age",
  wt_kg = "wt_kg", ht_m = "ht_m", output = "kcal_day", RER = 0.85,
  kcal_table = "Lusk"
)

## Now we can pull all the pieces together into
## a final estimate of energy expenditure
EE <-
  # Start by merging the demographic and accelerometer data
  merge(demo, AG) %>%
  # Then determine how many kcal to impute during non-wear minutes
  # (NB: The number of minutes in a full day is 24*60 = 1440)
  dplyr::mutate(
    basal_kcal_day = round(basal_kcal_day),
    nonwear_kcal = basal_kcal_day*(nonwear_mins / 1440),
    wear_kcal = kcal_kg * wt_kg,
    total_kcal = nonwear_kcal + wear_kcal,
    kcal_kg = NULL
  ) %>%
  # Remove the activity count totals for simplicity
  dplyr::select(!dplyr::matches("^[AV][xe]")) %>%
  # Reorder the variables
  dplyr::relocate(participant_compliant, day_compliant, .after = basal_kcal_day)
```

Step 8: Calculate energy intake

For this final step, we will need to determine the average daily energy expenditure, then cross-reference it against daily change in energy storage to determine energy intake. The energy storage data will likely come from repeated dual-energy X-ray absorptiometry scans. For the purpose of this vignette, we will assume those data have already been processed to determine daily change in energy storage.

```r
# Suppose the energy storage (ES) data look like this
# (reflecting a 100 kcal/day surplus):
ES <- data.frame(filename = basename(sample_file), ES = 100)

## Then we can calculate energy intake (EI) like so:
intake <-
  merge(EE, ES) %>%
  dplyr::rename(EE = total_kcal) %>%
  dplyr::mutate(
    ES = round(ES),
    EE = round(EE),
    EI = ES + EE
  ) %>%
  dplyr::select(
    dplyr::all_of(names(demo)),
    total_mins, nonwear_mins, ES, EE, EI
  )
```

```
## Print the output:
print(intake)
```

```
##                        filename sex ht_m wt_kg age basal_kcal_day total_mins
## 1 TAS1H30182785_2019-09-17.gt3x   M  1.8    75  30           1817         41
##   nonwear_mins  ES EE  EI
## 1            5 100 69 169
```

With this example, it's important to remember we aren't looking at a full day of data. Furthermore, we have simulated some data that may not be physiologically realistic. But all that aside, let's look at the output and see what it's telling us. The output shows that the participant burned an estimated 69 kcal during the 41 minutes in question, and that their energy storage increased by 100 kcal. Thus, the participant would have needed to consume 169 kcal in the same time period to achieve the observed outcome.

Conclusion

This accelerometer-based method entails a long but highly programmable process. With a few tweaks to the code above, users can potentially process data from many participants with a single keystroke. This is a major advantage compared to the labor- and cost-intensiveness of doubly labeled water. However, there is a learning curve associated with using this R-based method, and accelerometry itself is not yet a gold standard method for free-living assessments. Furthermore, there are other methods (such as the NIDDK Body Weight Planner) that should also be considered. Thus, study-specific decisions are necessary when selecting an assessment method, based on the strengths and weaknesses of each method and how they relate to the research question being asked. The accelerometer-based method outlined above is meant to show proof-of-concept for a technique that presents many opportunities for refinement going forward. Future studies are needed to assess the criterion validity of accelerometer-based methods, as well as to calibrate new predictive models with improved group-level and individual-level accuracy. The code above provides an important first step in that process. Please let us know if there is anything we can clarify. Thanks for your interest, and good luck in your research!