Introduction to the **ActivityIndex** package in R

Created: October 15, 2015;   *Revised*: October 22, 2015

The **ActivityIndex** package contains functions to 1) read raw accelerometry data and 2) compute "Activity Index" (AI) using the raw data. This introduction provides step-by-step instructions on how to read data from `.csv` files and then compute AI.

1 Data description

The sample data were collected by accelerometer GT3X+ (ActiGraph,), downloaded from `https://help.theactigraph.com/entries/21688392-GT3X-ActiSleep-Sample-Data`. The data are available in the **ActivityIndex** package and their paths can be acquired using command:

```
system.file("extdata", "sample_GT3X+.csv", package = "ActivityIndex")
system.file("extdata", "sample_table.csv", package = "ActivityIndex")
```

`sample_GT3X+.csv` is the standard output of GT3X+ accelerometer, with a 10-line header containing the basic information of the data collection, followed by 3-column raw acceleration data. `sample_table.csv` contains the same 3-column acceleration data without the 10-line header. The first 15 lines of `sample_GT3X+.csv` are shown below:

```
## ------------ Data File Created By ActiGraph GT3X+ ActiLife v6.7.1 Firmware v2.5.0 date f
## Serial Number: NEO1DXXXXXXXX
## Start Time 10:54:00
## Start Date 6/27/2012
## Epoch Period (hh:mm:ss) 00:00:00
## Download Time 16:25:52
## Download Date 6/28/2012
## Current Memory Address: 0
## Current Battery Voltage: 4.22     Mode = 12
## --------------------------------------------------
## 0,0,0
## 0,0,0
```

```
## 0,0,0
## 0,0,0
## 0,0,0
```

while the first 5 lines of `sample_table.csv` are

```
## 0,0,0
## 0,0,0
## 0,0,0
## 0,0,0
## 0,0,0
```

Users should follow the same format while preparing their own data.

2 Read the data

`ReadGT3XPlus` and `ReadTable` functions read the GT3X+ `.csv` file and the 3-column acceleration table, respectively. To read the data, use the following code

```
sampleGT3XPlus = ReadGT3XPlus(system.file("extdata",
    "sample_GT3X+.csv", package = "ActivityIndex"))
sampleTable = ReadTable(system.file("extdata", "sample_table.csv",
    package = "ActivityIndex"))
```

Now that object `sampleGT3XPlus` has class `GT3XPlus`, which contains the raw data and header information. Function `ReadGT3XPlus` automatically applies time stamps to the acceleration time series using the information from the header. For example, our sample data look like this

```
str(sampleGT3XPlus)

## List of 8
##  $ SN          : chr "NEO1DXXXXXXXX"
##  $ StartTime   : chr "10:54:00"
##  $ StartDate   : chr "2012-06-27"
##  $ Epoch       : chr "00:00:00"
##  $ DownloadTime: chr "16:25:52"
##  $ DownloadDate: chr "6/28/2012"
##  $ Hertz       : num 30
##  $ Raw         :Classes 'data.table' and 'data.frame': 1006080 obs. of  5 variables:
```

```
##   ..$ Date: chr [1:1006080] "2012-06-27" "2012-06-27" "2012-06-27" "2012-06-27" ...
##   ..$ Time: chr [1:1006080] "10:54:00" "10:54:00" "10:54:00" "10:54:00" ...
##   ..$ X   : num [1:1006080] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ Y   : num [1:1006080] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ Z   : num [1:1006080] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "class")= chr "GT3XPlus"
```

However, `sampleTable` is much simpler, since limited information was given. The first 6 lines of it look like this

```
head(sampleTable, n = 6)
```

```
##    Index X Y Z
## 1:     1 0 0 0
## 2:     2 0 0 0
## 3:     3 0 0 0
## 4:     4 0 0 0
## 5:     5 0 0 0
## 6:     6 0 0 0
```

3 Compute AI

AI is a metric to reflect the variability of the raw acceleration signals after removing systematic noise of the signals. Formally, its definition (a one-second AI) is

$$\mathrm{AI}_i^{\mathrm{new}}(t;H) = \sqrt{\max\left(\frac{1}{3}\left\{\sum_{m=1}^{3}\frac{\sigma_{im}^2(t;H)-\bar{\sigma}_i^2}{\bar{\sigma}_i^2}\right\},0\right)}, \tag{1}$$

where $\sigma_{im}^2(t;H)$ $(m = 1,2,3)$ is axis-m's moving variance during the window starting from time t (of size H), and $\bar{\sigma}_i$ is the systematic noise of the signal when the device is placed steady.

Function `computeActivityIndex` is used to compute AI. The syntax of the function is

```
computeActivityIndex(x, x_sigma0 = NULL, sigma0 = NULL,
    epoch = 1, hertz)
```

`x` is the data used to compute AI. It can either be a `GT3XPlus` object, or a 4-column data frame (tri-axial acceleration time series with an index column). Either `x_sigma0` or `sigma0` are used to determine the systematic noise $\bar{\sigma}_i$. More detailed example will follow to illustrate how to use them. `epoch` is the epoch length (in second) of the AI. For example, the default `epoch=1` yields to 1-second AI, while minute-by-minute AI is given by `epoch=60`. `hertz` specifies the sample rate (in Hertz), which is usually 10, 30 or 80, etc.

We will continue our example of computing AI using our data `sampleGT3XPlus` and `sampleTable`.

3.1 Find $\bar{\sigma}_i$ on-the-fly

According to the definition of the systematic noise $\bar{\sigma}_i$, it changes with subject i. Therefore, strictly speaking, we are to compute $\bar{\sigma}_i$ every time we compute AI for a new subject i. Argument `x_sigma0` can be used to specify a 4-column data frame (one column for indices and three columns for accceleration) which is used to calculate $\bar{\sigma}_i$. The 4-column data frame should contain the raw accelerometry data collected while the accelerometer is not worn or kept steady. For example, if we say a segment of our sample data (`sampleTable[1004700:1005600,]`) meets such requirement, we could compute AI using the following code

```
AI_sampleTable_x = computeActivityIndex(sampleTable,
    x_sigma0 = sampleTable[1004700:1005600, ], epoch = 1,
    hertz = 30)
AI_sampleGT3XPlus_x = computeActivityIndex(sampleGT3XPlus,
    x_sigma0 = sampleTable[1004700:1005600, ], epoch = 1,
    hertz = 30)
```

3.2 Find $\bar{\sigma}_i$ beforehand

Sometimes we do not want to calculate $\bar{\sigma}_i$ whenever computing AI. For example, if 10 accelerometers were used to collect data over 100 subjects, there is no reason to calculate $\bar{\sigma}_i$ for 100 times. One $\bar{\sigma}_i$ is only needed for one accelerometer. Furthermore, if we could verify the $\bar{\sigma}_i$'s of the 10 accelerometers are close to each others, we could combine them into a single $\bar{\sigma} = \sum_{i=1}^{10} \bar{\sigma}_i/10$. In this case, $\bar{\sigma}$ will be used for all subjects in that study, which is crucial for fast processing of data collected by large studies.

This can be achieved by using the argument `x_sigma0` to specify a pre-determined $\bar{\sigma}_i$. Still using the same segment of data (`sampleTable[1004700:1005600,]`) as an example, we calculate a `sample_sigma0` beforehand with code

```
sample_sigma0 = Sigma0(sampleTable[1004700:1005600, ],
    hertz = 30)
```

Then we could use this `sample_sigma0`=0.00218 to compute AI with code

```
AI_sampleTable = computeActivityIndex(sampleTable, sigma0 = sample_sigma0,
    epoch = 1, hertz = 30)
AI_sampleGT3XPlus = computeActivityIndex(sampleGT3XPlus,
    sigma0 = sample_sigma0, epoch = 1, hertz = 30)
```

4 Explore AI

Using either method to compute AI yield to the same result. The output of function `computeActivityIndex` has two columns: `RecordNo` saves the indices and `AI` stores AI. The first 10 lines of `AI_sampleGT3XPlus` is as follow

```
head(AI_sampleGT3XPlus, n = 10)

##    RecordNo      AI
## 1  10:54:00   0.000
## 2  10:54:01   0.000
## 3  10:54:02   0.000
## 4  10:54:03   0.000
## 5  10:54:04 133.708
## 6  10:54:05  34.837
## 7  10:54:06  62.947
## 8  10:54:07  54.207
## 9  10:54:08 124.708
## 10 10:54:09 147.842
```

We could also compute AI in different epoch. Say we want minute-by-minute AI, then we could use the following code

```
AI_sampleTable_min = computeActivityIndex(sampleTable,
    sigma0 = sample_sigma0, epoch = 60, hertz = 30)
AI_sampleGT3XPlus_min = computeActivityIndex(sampleGT3XPlus,
    sigma0 = sample_sigma0, epoch = 60, hertz = 30)
```

And according to the definition of AI, the minute-by-minute AI's are simply the sum of all 1-second AI within each minute. The AI during the first 6 minutes are

```
head(AI_sampleGT3XPlus_min)

##   RecordNo       AI
## 1 10:54:00 3002.460
## 2 10:55:00  392.185
## 3 10:56:00  655.593
## 4 10:57:00   89.337
## 5 10:58:00  499.150
## 6 10:59:00  238.130
```