Analysis Guide

Amy R Hodgson

22/07/2019

This guide describes how to analyse data from the GENEActiv accelerometers in R. GGIR is an R package created by Dr Vincent van Hees to process raw accelerometry data. Van Hees provides a shell function, which can provide a number of variables for activity and sleep research, requiring only the specification of input and output folder locations, as well as a few other small adjustments. The signal processing includes automatic calibration, detection of abnormal values, detection of non-wear and calculation of average acceleration. GGIR then uses this information estimate physical activity, inactivity and sleep. The result is a number of .csv and .pdf files containing relevant variables.

This package has been used by a number of research groups around the world, and a list of publications using/related to GGIR is available here. Also see here https://github.com/wadpac/GGIR/issues for a place where questions about GGIR can be asked. Questions are usually answered by the author or another user of GGIR. If you run into a problem it's a good idea to check here, as someone else may have had the same issue and found a solution.

Step 1: Organise your folders

I would recommend creating one folder to contain all parts of your analysis, named in the following way:

[date]_[your-name]_[keyword-for-your-study]

So my folder name would be:

2019-08-02_Amy-Hodgson_example

Within this folder create a folder for your scripts, one for your raw data, and another for the results that will be produced. In this guide, the folder containing the raw data will be named *Data*, the folder containing the results will be called *Output*, and the folder containing the sleep diary is called *Logs*.

Step 2: Get your data

remember about IDs when getting data from GENEActiv software

Download the raw .bin files produced by the GENEActiv software, and save all of these files in your *Data* folder. Make sure you have specified appropriate unique IDs. If you are using a sleep diary, remember that the ID in the diary must exactly match the data file ID. It is also important to note that GGIR will not work with files converted to epochs with the GENEActiv software, so it is better to simply use the .bin files.

Steps 3 & 4 relate to setting up the sleep diaries for use with GGIR. If you are not using a sleep diary, feel free to skip these steps.

Step 3: Set up your sleep diary

GGIR can use basic infomation from a sleep diary to improve estimates of sleep parameters. Your sleep log should be set up so that the first column contains the participant IDs and the subsequent columns contain alternating sleep onset and wake times. See the GGIR vignette here for an example of how this should look. Each row should contain a different participant.

Step 4: Sleep diary checklist

In my experience, integrating the sleep diary is the part of the script that is most likely to cause problems. However, if you follow the items in this checklist, the risk of errors or other problems will be minimised.

1. Do you have the same number of onset and wake columns in your sleep log as you have days in your actigraphy data? If you have some days missing in the sleep log, still create the right number of columns, but leave them blank. The sleep log that can be entered into the function has only one 'sleep' time, so you must account for whether you have entered bed time or sleep onset when looking at the results.
2. Do the participant IDs from the data file match the IDs in the sleep log? Note: if the `idloc` argument is set to `1`, the program will look for the participant ID in the file header, and if it is set to `2`, it will look for the ID in the letters or numbers before the underscore in the file name.
3. Is your sleep log file saved as a .csv file?

If you are having problems, check the *I keep getting this error* guide, which described a few possible problems and solutions.

Step 5: Set up for GGIR shell function

First, create a new R script or RMarkdown file to use for your analysis. The GGIR shell function, which is shown in Step 6, requires some input that is easier to specify beforehand.

The `datadir` argument should specify a file path to the data you want to analyse. In most cases you will want to analyse multiple files, and so you will specify a folder rather than a single file. Using the `list.files()` function creates a list of files within the folder you specify, which should be the *Data* folder you created earlier to store all of your .bin files from the GeneActiv software. Make sure that the `full.names` argument is set to `TRUE`, so that the full file paths are listed rather than the relative file paths.

The `outputdir` argument should specify the path to the folder where you want the analysed data to be stored. In this case it is the folder called *Output*.

`studyname` specifies the name of the study, which will determine the name of the folder containing the analysed data (within the *Output* folder).

`loglocation` should be the path to the location of your sleep log. Don't forget to include the file extension, which should be `.csv`, as the sleep log file should be saved in this format.

Remember about file paths- can you shorten them by setting wd and using ~?

```
datadir <- list.files("C:/Users/amyho/OneDrive/Documents/GENEActiv/Data",
                      full.names =T)
outputdir <- "C:/Users/amyho/OneDrive/Documents/GENEActiv/Output"
studyname <- "Date_Name_Keyword"
loglocation <- "C:/Users/amyho/OneDrive/Documents/GENEActiv/Logs/Date_name_Sleeplogs.csv"

nnights <- 5 # The number of nights in your data
f0 <- 1 # File to start with (default = 1)
f1 <- 4 # File to finish with (default = number of files available)
```

If you have problems due to the file paths, there is some advice for potential solutions in the error guide.

Secondly, load in the packages you will need to run your script.

```
library(GGIR)
library(zoo)
```

Step 6: The shell function

This function may seem very long, but if you have specified the variables above, there should be very few changes that need to be made in most circumstances. If you are not using a sleep log, remove this argument from the function. If your study involves assessing physical activity and you want to alter the thresholds, or you want more information about any of these arguments, each is described in more detail in either the vignette or the package documentation.

A couple of arguments to be aware of are `includenightcrit`, which refers to the minimum number of valid hours per night (set by default to 16), and `def.noc.sleep` allows you to specify the time window in which sustained activity will be interpreted as sleep, only used if there is no sleep log.

Copy the shell function below your setup code, and you should be ready to run your analysis. Simply run the code and it should begin processing your data. Depending on how many participants you have, and the computer you are using, this may take quite a long time- perhaps even several hours. Do not be concerned by this, the script will stop and produce an error message if there is a problem. However if your computer crashes, you may need to find a more powerful computer to run your analysis.

```
g.shell.GGIR(
             mode=c(1,2,3,4,5),
             datadir = datadir,
             outputdir = outputdir,
             idloc = 2,
             f0=f0, f1=f1,
             studyname = studyname,
             #---------------------------------
             # Part 1:
             #---------------------------------
             do.enmo = TRUE,            do.anglez=TRUE,
             chunksize=1,                printsummary=TRUE,
             #---------------------------------
             # Part 2:
             #---------------------------------
             strategy = 1,               ndayswindow=7,
             hrs.del.start = 0,          hrs.del.end = 0,
             maxdur = 9,                 includedaycrit = 16,
             winhr = c(5,10),
             qlevels = c(c(1380/1440),c(1410/1440)),
             qwindow=c(0,24),
             ilevels = c(seq(0,400,by=50),8000),
             mvpathreshold =c(100,120),
             bout.metric = 4,
             closedbout=FALSE,
             #---------------------------------
             # Part 3:
             #---------------------------------
             timethreshold= c(5),        anglethreshold=5,
             ignorenonwear = TRUE,
             #---------------------------------
             # Part 4:
             #---------------------------------
             excludefirstlast = FALSE,
             includenightcrit = 16,
             def.noc.sleep = c(),
             loglocation= loglocation,
```

```
            outliers.only = FALSE,
            criterror = 4,
            relyonsleeplog = FALSE, sleeplogidnum = FALSE,
            colid=1, coln1=2,
            do.visual = TRUE,
            nnights = nnights,
            #---------------------------------
            # Part 5:
            #---------------------------------
            threshold.lig = c(30), threshold.mod = c(100),  threshold.vig = c(400),
            boutcriter = 0.8,      boutcriter.in = 0.9,     boutcriter.lig = 0.8,
            boutcriter.mvpa = 0.8, boutdur.in = c(1,10,30), boutdur.lig = c(1,10),
            boutdur.mvpa = c(1),   timewindow = c("WW"),
            #---------------------------------
            # Report generation
            #---------------------------------
            do.report=c(2,4,5), visualreport=TRUE,
            dofirstpage = TRUE, viewingwindow=1
            )
```

Step 7: Check your data

Once the function has run successfully, it is a good idea to check your data for any mistakes or problems. Firstly, if you have used a sleep log, the 'sleeplog_used' column of the part 4 output should say TRUE. If it is FALSE, this means that there has been a problem- you may have made a mistake specifying the location of the sleep log, or there may have been an issue with matching the IDs.

Secondly, there is an option to run the script and visualise only outliers. By default `outliers.only` is set to `FALSE`, but if you change this to `TRUE`, only nights with a difference in onset or waking time larger than `criterror` will be visualised, allowing you should be able to see any potentially abnormal values in your output. **Check this works as expected**

GGIR produces a large number of variables in the output. There is a dictionary of variables on the GGIR vignette page, but I have also made a shorter guide describing the key sleep variables and how they are calculated.