Rethomics: an R framework to analyse high-throughput behavioural data

Quentin Geissmann[1*], Luis Garcia Rodriguez[2], Esteban J Beckwith[1], Giorgio F Gilestro[1*]

1 Department of Life Sciences, Imperial College London, London, United Kingdom
2 Affiliation Dept/Program/Center, Institution Name, City, State, Country

* qgeissmann@gmail.com, giorgio@gilest.ro

Abstract

Ethomics, a quantitative and high-throughput approach to animal behaviour, is a new and exciting field. The recent development of automatised methods that can score various behaviours on a large number of animals provides biologists with an unprecedented tools to decipher these complex phenotypes. Analysing ethomics data comes with many challenges that are independent of these acquisition platform. However, there is little effort in providing a generic framework to specifically analyse multiple and long behavioural time series. We developed the `rethomics` framework, a suite of `R` packages that altogether offer utilities to: import, store, visualise and analyse behavioural data. In this article, we describe it and show an example of its application to the blooming field of sleep and circadian rhythm in fruit fly. The `rethomics` framework is available and documented at https://rethomics.github.io.

Introduction

Animal behaviours are complex phenotypical manifestations of the interaction between nervous systems and their external or internal environment. In the last few decades, our ability to record vast quantities of various phenotypical data has tremendously increased. Behaviour scoring is certainly not an exception to this trend. Indeed, many platforms (TODO citations) have been developed in order to allow biologists to continuously record behaviours such as activity, position and feeding of multiple animals over long durations (days or weeks).

The availability of large amounts of data is very exciting as it paves the way for in-depth analyses. Clearly, the multiplicity of model organisms, hypotheses and paradigms should be reflected in diverse range of recording tools. However, when it comes to the subsequent data analysis, there is no unified, programmatic, framework that could be used as a set of building blocks in a pipeline. Instead, tools tend to consist of graphical interfaces with rigid functionalities that only import data from a single platform. There are, at least, three issues with this approach. First of all, state-of-the-art analysis and visualisation requires a level of reproducibility, flexibility and scalability than only a programmatic interface can provide. Secondly, it favours replicated work as developers need to create their independent solution to similar problems. Lastly, it links analysis and visualisation to the target tool, which makes it very difficult to share cross-tool utilities and concepts.

Thankfully, behavioural data is conceptually largely agnostic of the acquisition platform and paradigm. Typically, the behaviour of each individual is described by a

long time series (possibly multivariate and heterogeneous). Importantly, individuals are labelled with arbitrary metadata defined by the experimenter (*e.g.* sex, treatment and genotype). Efficiently combining and manipulating these two types of information, on datasets of hundreds of individuals, each recorded for weeks, is not trivial.

In the article herein, we describe `rethomics`, a framework that unifies analysis of behavioural dataset in an efficient and flexible manner. It offers an elegant computational solution to store, manipulate and visualise a large amount of data. We expect it to fill the gap between behavioural biology and data sciences. `rethomics` comes with a extensive documentation and a set of both practical and theoretical tutorials.

Design and Implementation

`rethomics` is implemented as a collection of small packages linked to one another (Fig 1). This paradigm follows the model of modern frameworks such as the `tidyverse`, which results in increased testability and maintainability. In it, the different tasks of the analysis workflow (*i.e.* data import, manipulation and visualisation) are explicitly handled by different packages. At the core of `rethomics`, the `behavr` package offers a very flexible and efficient solution to store both large amounts data (*e.g.* position and activity) and metadata (*e.g.* treatment, genotype and so on) in a single `data.table`-derived object. Any input package will import experimental data as a `behavr` table which can, in turn, be manipulated and visualised regardless of the original input platform. Results and plots integrate seamlessly within the `R` ecosystem, hence providing users with state-of-the-art visualisation and statistics tools.

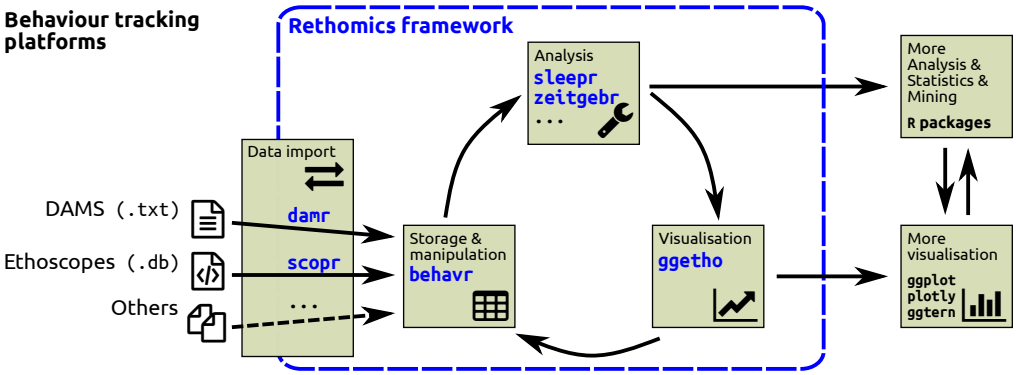

Fig 1. The `rethomics` workflow. Diagram representing, from left to right, the interplay between the raw data, the `rethomics` packages (in blue) and the rest of the `R` ecosystem.

Internal data structure

We created `behavr` (Fig 2), a new data structure, based on the widely adopted `data.table` object, in order to address two challenges that are inherent to manipulating ethomics results.

Firstly, there could be very long (typically $k_i > 10^8, \forall i \in [1, n]$), multivariate (often, $q > 10$), time series for each individual. For instance, each series could represent variables that encode coordinates, orientation, dimensions, activity, colour intensity and so on, sampled several times per second, over multiple days. Therefore, data structure must be computationally efficient – both in term of memory footprint and processing speed.

Secondly, a large amount of individuals are often studied (typically $n > 100$). Each individual (i) is associated with metadata, that is a set of p "metavariables" that describe experimental conditions. For instance, metadata stores information regarding the date and location of the experiment, treatment, genotype, sex, *post hoc* observations and other arbitrary metavariables. It is good practice to record as many metavariables as possible so they can later be used as covariates. Therefore, typically $p > 10$.

`behavr` tables link metadata and data within the same object, extending the syntax of `data.table` to manipulate, join and access metadata. This approach guaranties that any data point can be mapped correctly to its parent metadata. It also allows implicit update of metadata when data is altered. For instance, when is data filtered, only the remaining individuals should be in the new metadata. It is also important that metadata and data can interoperate. For instance, when one wants to update variable according to the value of a metavariable (say, alter the variable x only for animals with the metavariable $sex =$ "$male$").

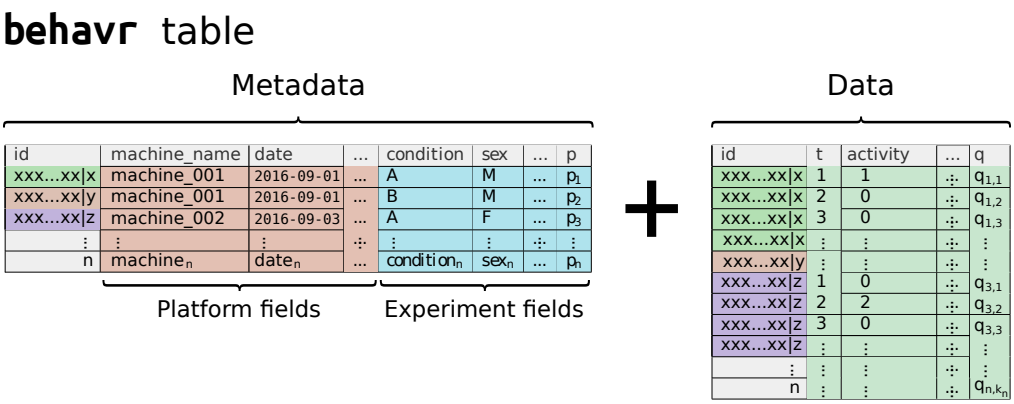

Fig 2. Example of a `behavr` table. The metadata holds a single row for each of the n individuals. Its columns, the p metavariables, are of two kind: either required and defined by the acquisition platform (*i.e.* used to map the data) or user-defined (*i.e.* arbitrary). In the data, each row is a "read" (*i.e.* information about one animal at one time point). It is formed of q variables and is expected to have a very large number of reads k for each individual. Data and metadata are implicitly joined with the `id` field. Notably, the names used for variables and metavariable are only plausible examples that will likely differ in practice.

Data import

Data import package translate results from a recording platform (*e.g.* text files and databases) into a single `behavr` object. Currently, we provide a package to read Drosophila Activity Monitor (DAM2) data and another one for Ethoscope data. Although the structure of the raw results if very different, conceptually, loading data is very similar. In all cases the user is asked to generate a metadata table (one row per individual). In it, there will be both mandatory and optional columns. The mandatory ones are the necessary and sufficient information to fetch data (*e.g.* machine id, region of interest and date). The optional columns are user-defined arbitrary fields that relate to the experiment itself (*e.g.* condition and sex).

In this respect the metadata file is a standardised an comprehensive data frame describing an experiment. Using such a structure comes with multiple advantages. For instance, it simplify collaboration and data exchange as all treatments and individuals are very explicit. Then, it promotes good experimental practices such as interspersion of treatments (indeed, without it, users are tempted to simplify their design, for instance,

confounding device/location and treatment). Furthermore, it streamlines the inclusion and analysis of further replicates in the same workflow. Indeed, additional replicates can simply be added as new rows, and replicate number later user, if needed, as a covariate.

Visualisation

Long time series often need to be preprocessed before visualisation. Typically, users are interested in understanding individual or population trends over time. To integrate visualisation in `rethomics`, we implemented `ggetho`, a package extending the widely adopted `ggplot2` by providing preprocessing tools as well as new layers and scales. Our tools make full use of the internal `behavr` structure to deliver efficient representations of temporal trends. It particularly applies to the visualisation of long experiments, with the ability to, for instance, annotate light and dark phases, wrap time over a circadian day, display "double-plotted actograms" and periodograms. Importantly, `ggetho` is fully compatible with `ggplot`.

Circadian and sleep analysis

The packages `zeitgebr` and `sleepr` provide tools to analyse circadian behaviours and sleep, respectively. Together, they offer a suite of methods to compute periodograms and find their peaks, score sleep from inactivity (*e.g.* using the "five minute rule"), and characterise the architecture of sleep bouts (*e.g.* number, length and latency).

Results

```
library(damr)      # input DAM2 data
library(zeitgebr)  # periodogram computation
library(sleepr)    # sleep analysis
library(ggetho)    # behaviour visualisation
```

Then, the metadata file is read and linked to the `.txt` result files.

```
metadata <- link_dam2_metadata("metadata.csv",".")     # linking
# print(metadata)                                      # check metadata
dt <- load_dam2(metadata)                              # loading
summary(dt)                                            # quick summary

## behavr table with:
##  58 individuals
##  8 metavariables
##  2 variables
##  1.58722e+05 measurements
##  1 key (id)
```

Preprocessing

We notice, from the metadata, that the two replicates do not have the same time in baseline. We would like to express the time relative to the important event: the transition to `LL`. To do so, we subtract the `baseline_days` metavariable from the `t` variable. This gives us an opportunity to illustrate the use `xmv()`, which expands

metavariables as variables. In addition, we use the `data.table` syntax to create, in place, a `moving` variable. It is `TRUE` when and only when `activity` is greater than zero:

```
# baseline subtraction -- note the use of xmv
dt[,t := t - days(xmv(baseline_days))]
dt[, moving :=  activity > 0]
summary(dt)
```

To simplify visualisation, we create our own `label` metavariable, as combination of a number and `genotype`. In the restricted context of this analysis, `label` acts a unique identifier. Importantly, we keep `id` , which is more rigorous and universal.

```
dt[, label := interaction(1:.N, genotype), meta=T]
print(dt)
```

Curation

It is important to visualise an overview of how each individual behaved and, if necessary, alter the data accordingly. For this, we generate a tile plot (Fig 3A).

```
# make a ggplot object with label on the y and moving on the z axis
fig3A <- ggetho(dt, aes(y=label, z=moving)) +
  # show data as a tile plot
  # that is z is a pixel whose intensity maps moving
  stat_tile_etho() +
  # add layers to draw annotations to show L and D phases
  # as white and black, respectivelly
  # the first layer is for the baseline (until t=0)
  stat_ld_annotations(x_limits = c(dt[,min(t)], 0)) +
  # in the 2nd one, we start at 0 and use grey
  # instead of black as we work in LL
  stat_ld_annotations(x_limits = c(0, dt[,max(t)]),
                      ld_colours = c("white", "grey"))
```

Activity of dead or escaped animals is falsely scored as long series of zeros. Our `sleepr` package offer a tool to detect and remove such artefactual data. The updated version can be visualised in Fig 3B.

```
# remove data after death
dt <- sleepr::curate_dead_animals(dt, moving)
# same as above
fig3B <- ggetho(dt, aes(y=label, z=moving)) +
    stat_tile_etho() +
    stat_ld_annotations(x_limits = c(dt[,min(t)], 0)) +
    stat_ld_annotations(x_limits = c(0, dt[,max(t)]),
                        ld_colours = c("white", "grey"))
```

For the purpose of this example, we keep only individuals that have *at least five days in LL*.

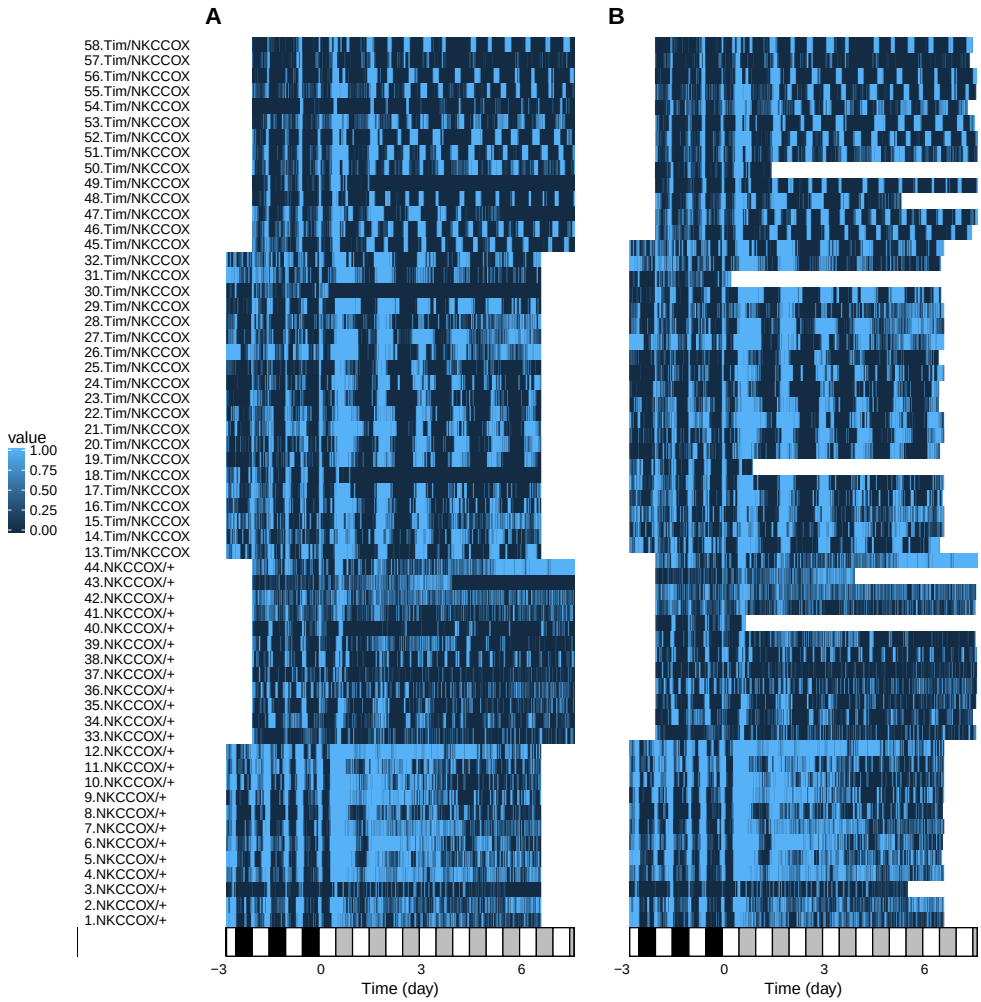

Fig 3. Experiment quality control. Tile plot representing the fraction of time spent moving as a colour intensity. Each individual is represented by a row and time, on the x axis, is binned in 30 minute.

```
# for each id, we check for validity
valid_dt <- dt[ , .(valid = max(t) > days(5)), by=id]
# a vector of all valid ids
valid_ids <- valid_dt[valid == T, id]
# filter dt with the valid ids
dt <- dt[id %in% valid_ids]
summary(dt)

## behavr table with:
##  52 individuals
##  9 metavariables
##  3 variables
##  1.40609e+05 measurements
##  1 key (id)
```

Note that as a result, we now have 52 “valid”individuals.

Double plotted actograms

"Double-plotted actograms" are a common visualisation of periodicity and rhythmicity in circadian experiments. In Fig S1A, we show the double-plotted actograms of each animals:

```
figS1A <- ggetho(dt, aes(z = moving), multiplot = 2) +
            stat_bar_tile_etho() +
            facet_wrap( ~ label, ncol=4) +
            scale_y_discrete(name="Day")
```

Periodograms

Ultimately, in order to quantify periodicity and rhythmicity, we compute periodograms. In several methods are implemented in `zeitbebr`. In this example, we generate χ^2 periodrograms and lay them out in a grid (see S1 Fig).

```
dt_ll <- dt[t > days(1)]
per_dt <- periodogram(moving,
                      dt_ll,
                      resample_rate = 1/mins(10),
                      FUN=chi_sq_periodogram)

per_dt <- find_peaks(per_dt)

figS1B <- ggperio(per_dt, aes(y = power, peak=peak)) +
                  geom_line() +
                  geom_line(aes(y=signif_threshold), colour="red") +
                  geom_peak() +
                  facet_wrap( ~ label, ncol=4)
```

Availability and Future Directions

All packages in the `rethomics` framework are available under the terms of the GPLv3 license and listed at https://github.com/rethomics/. Extensive installation instructions as well as reproducible demos and tutorials are available at https://rethomics.github.io/. All packages are continuously integrated and unit tested on several version of `R` to minimise resent and future issues.

* Other inputs
* Position analysis
* GUI
* ...

Supporting information

S1 Fig. Bold the title sentence. Add descriptive text after the title of the item (optional).

Acknowledgements

TODO:
Han Kim
Maite
Hannah
Patrick Krätschmer

References

1. Conant GC, Wolfe KH. Turning a hobby into a job: how duplicated genes find new functions. Nat Rev Genet. 2008 Dec;9(12):938–950.

2. Ohno S. Evolution by gene duplication. London: George Alien & Unwin Ltd. Berlin, Heidelberg and New York: Springer-Verlag.; 1970.

3. Magwire MM, Bayer F, Webster CL, Cao C, Jiggins FM. Successive increases in the resistance of Drosophila to viral infection through a transposon insertion followed by a Duplication. PLoS Genet. 2011 Oct;7(10):e1002337.