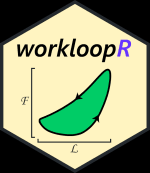
Studies of muscle physiology often rely on closed-source, proprietary software for not only recording data but also for data wrangling and analyses. Although specialized software might be necessary to record data from highly-specialized equipment, data wrangling and analyses should be free from this constraint. It’s becoming more common for researchers to provide code along with published papers (but usually as Matlab scripts…ugh), but it is still typical for most analyses to be performed with code that stays in-house. Even worse is when some of the steps are carried out in a non-reproducible way, like needing to click and drag sliders across a screen (*by hand!!*) to select a data range of interest.



To give muscle physiologists a set of tools to help perform reproducible research, we present our new R package workloopR.

**What it does**

workloopR (pronounced “work looper”) provides a variety of features that we hope will help the typical muscle physiology researcher’s workflow. These include:

* Data import from .ddf files, like those produced by Aurora Scientific, with retention of important metadata like file creation times and stimulus protocols.
* Automatic cycle selection within data, with three options for how cycles are defined; see our select\_cycles() function
* Gear ratio correction and other forms of transformation; see Data transformation functions
* Work loop analyses, which integrate muscle force and length change to determine mechanical work output (and power output)
* Analyses of twitch and tetanic data to determine the time course of force production

**Some perks of using workloopR**

**Metadata**

One thing we think is helpful to researchers is how workloopR automatically retains metadata when reading a .ddf file. We’ll demonstrate with the example work loop file provided with the package; first, let’s load the package and import some data.

library(workloopR)

## import the workloop.ddf file included within workloopR

wl\_dat <-read\_ddf(system.file("extdata", "workloop.ddf",

package = 'workloopR'),

phase\_from\_peak = TRUE)

Important metadata from the file are stored as attributes of the object. Here’s what’s stored for the work loop file:

names(attributes(wl\_dat))

## [1] "names" "class" "row.names"

## [4] "stimulus\_frequency" "cycle\_frequency" "total\_cycles"

## [7] "cycle\_def" "amplitude" "phase"

## [10] "position\_inverted" "units" "sample\_frequency"

## [13] "header" "units\_table" "protocol\_table"

## [16] "stim\_table" "stimulus\_pulses" "stimulus\_offset"

## [19] "stimulus\_width" "gear\_ratio" "file\_id"

## [22] "mtime"

Specific metadata can be called via either attributes() or attr(). For example, we’ll take a look at the stimulus protocol, which shows the specific pattern of electrical stimulation that was delivered to the muscle during the experiment:

attr(wl\_dat,"stim\_table")

## offset frequency width pulses cycle\_frequency

## 1 0.012 300 0.2 4 28

## 2 0.000 0 0.0 0 0

There’s lots of useful info here: offset refers to the span of time (usually in secs) that elapsed before the stimulation began; frequency and width give info on how often (in Hz) and for how long (ms) the muscle was stimulated; pulses shows the total number of stimulus pulses that were delivered; and cycle\_frequency shows the frequency (Hz) of muscle length oscillation for further context.

**Visualization**

Although our functions do not generate plots, each is designed to be friendly to either base R or ggplot2 (and other Tidyverse packages). For example, we’ll show a work loop plotted in two ways. Because mechanical work is the product of force (y-axis) and distance (“Position” here, x-axis), the area within the loops of the following plots corresponds to the work that is performed by the muscle.

library(ggplot2)

## select cycles 3 through 5 using a l0-to-l0 definition

wl\_selected <- select\_cycles(wl\_dat, cycle\_def = "lo", keep\_cycles = 3:5)

## apply a gear ratio correction, run the analysis function,

## and then get the full object

wl\_analyzed <- analyze\_workloop(wl\_selected, GR = 2)

## base R work loop plot for the second retained cycle (cycle "b")

plot(wl\_analyzed$cycle\_b$Position,

wl\_analyzed$cycle\_b$Force,

xlab = "Position (mm)",

ylab = "Force (mN)",

main = "Work loop plot via base R",

bty = "n", tck = 0.02)

## now via ggplot

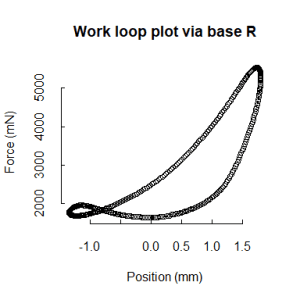
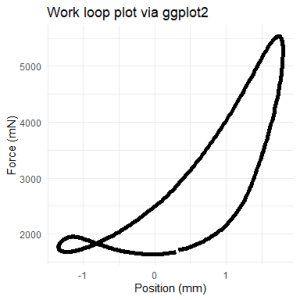
ggplot(wl\_analyzed$cycle\_b, aes(x = Position, y = Force)) +

geom\_path(lwd = 2) +

labs(y = "Force (mN)", x = "Position (mm)") +

ggtitle("Work loop plot via ggplot2") +

theme\_minimal()

**Summarizing a set of trials**

The batch processing capabilities of workloopR should also be handy for efficiently analyzing all files within a specific folder, e.g., a set of trials from a single experiment. Here’s an example:

## batch read and analyze a set of work loop trials stored within one directory

wl\_batch\_analyzed <- read\_analyze\_wl\_dir(

system.file("extdata/wl\_duration\_trials",

package = 'workloopR'),

sort\_by = 'file\_id',

phase\_from\_peak = TRUE,

cycle\_def = 'lo',

GR = 2,

keep\_cycles = 3

)

## now create a summary of the trials

wl\_batch\_summarized <- summarize\_wl\_trials(wl\_batch\_analyzed)

wl\_batch\_summarized

## File\_ID Cycle\_Frequency Amplitude Phase Stimulus\_Pulses

## 1 01\_4pulse.ddf 28 1.575 -24.36 4

## 2 02\_2pulse.ddf 28 1.575 -24.64 2

## 3 03\_6pulse.ddf 28 1.575 -24.92 6

## 4 04\_4pulse.ddf 28 1.575 -24.64 4

## Stimulus\_Frequency mtime Mean\_Work Mean\_Power

## 1 300 1572459771 0.0027387056 0.078427135

## 2 300 1572459771 0.0009849216 0.027832717

## 3 300 1572459771 -0.0002192395 0.004323004

## 4 300 1572459771 0.0022793831 0.065468837

**How to get workloopR**

We are not (yet) on CRAN but the package is available through rOpenSci’s GitHub:

#install.packages("devtools") # if devtools is not installed

devtools::install\_github("ropensci/workloopR")

## To build vignettes as well:

devtools::install\_github("ropensci/workloopR", build\_vignettes = TRUE)

**Feel free to make suggestions or requests**

workloopR, as the name implies, was originally designed to handle data from work loop experiments as well as experiments that are complementary to work loops like twitch and tetanic trials.

We are happy to expand the scope of the package to incorporate even more types of analyses of muscle physiology or biomechanics. We’re presently eager to add data import functions for non-ddf file types. We’re also interested in integrating support for electromyographic data recorded directly from muscles, but that may take a little longer to develop.