Package ‘workloopR’

August 5, 2019

Type Package

Title Analysis of Work Loops and Other Data from Muscle Physiology Experiments

Version 1.0.1

Author Vikram B. Baliga [cre, aut] (<https://orcid.org/0000-0002-9367-8974>),
Shreeram Senthivasan [aut] (<https://orcid.org/0000-0002-7118-9547>)

Maintainer Vikram B. Baliga <vbaliga87@gmail.com>

Description Functions for the import, transformation, and analysis of data from muscle physiology experiments. The work loop technique is used to evaluate the mechanical work and power output of muscle. Josephson (1985) <https://jeb.biologists.org/content/114/1/493> modernized the technique for application in comparative biomechanics. Although our initial motivation was to provide functions to analyze work loop experiment data, as we developed the package we incorporated the ability to analyze data from experiments that are often complementary to work loops. There are currently three supported experiment types: work loops, simple twitches, and tetanus trials. Data can be imported directly from .ddf files or via an object constructor function. Through either method, data can then be cleaned or transformed via methods typically used in studies of muscle physiology. Data can then be analyzed to determine the timing and magnitude of force development and relaxation (for isometric trials) or the magnitude of work, net power, and instantaneous power among other things (for work loops). Although we do not provide plotting functions, all resultant objects are designed to be friendly to visualization via either base-R plotting or 'tidyverse' functions.

URL https://github.com/vbaliga/workloopR

BugReports https://github.com/vbaliga/workloopR/issues

Imports pracma (>= 2.0.7),
signal (>= 0.7-6)

Suggests testthat (>= 2.1.1),
knitr,
rmarkdown

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1.9000

VignetteBuilder knitr

R topics documented:

`analyze_workloop` *Analyze work loop object to compute work and power output*

Description

Compute work and power output from a work loop experiment on a per-cycle basis.

Usage

```
analyze_workloop(x, simplify = FALSE, GR = 1, M = -1,
  vel_bf = 0.05, ...)
```

Arguments

x	A `workloop` object of class `muscle_stim` that has been passed through `select_cycles`. See Details.
simplify	Logical. If `FALSE`, the full analyzed workloop object is returned. If `TRUE` a simpler table of net work and power (by cycle) is returned.
GR	Gear ratio, set to 1 by default
M	Velocity multiplier, set adjust the sign of velocity. This parameter should generally be either -1 (the default) or 1.
vel_bf	Critical frequency (scalar) for low-pass filtering of velocity via `signal::butter()`
...	Additional arguments potentially passed down from `read_analyze_wl()` or `read_analyze_wl_dir()`

Details

Please note that select_cycles() must be run on data prior to using this function. This function relies on the input muscle_stim object being organized by cycle number.

The muscle_stim object (x) must be a workloop, preferably read in by one of our data import functions. Please see documentation for as_muscle_stim() if you need to manually construct a muscle_stim object from a non .ddf source.

The gear ratio (GR) and velocity multiplier (M) parameters can help correct for issues related to the magnitude and sign of data collection. By default, they are set to apply no gear ratio adjustment and to positivize velocity. Instantaneous velocity is often noisy and the vel_bf parameter allows for low-pass filtering of velocity data. See signal::butter() and signal::filtfilt() for details of how filtering is achieved.

Please also be careful with units! Se Warning section below.

Value

The function returns a list of class analyzed_workloop that provides instantaneous velocity, a smoothed velocity, and computes work, instantaneous power, and net power from a work loop experiment. All data are organized by the cycle number and important metadata are stored as Attributes.

Within the list, each entry is labeled by cycle and includes:

- Time: Time, in sec
- Position: Length change of the muscle, corrected for gear ratio, in mm
- Force: Force, corrected for gear ratio, in mN
- Stim: When stimulation occurs, on a binary scale
- Cycle: Cycle ID, as a letter
- Inst_velocity: Instantaneous velocity, computed from Position change, reported in meters/sec
- Filt_velocity: Instantaneous velocity, after low-pass filtering, again in meter/sec
- Inst_Power: Instantaneous power, a product of Force and Filt_velocity, reported in J
- Percent_of_Cycle: The percent of that particular cycle which has elapsed

In addition, the following information is stored in the analyzed_workloop object's attributes:

- stimulus_frequency: Frequency at which stimulus pulses occurred
- cycle_frequency: Frequency of oscillations (assuming sine wave trajectory)
- total_cycles: Total number of oscillatory cycles (assuming sine wave trajectory) that the muscle experienced.
- cycle_def: Specifies what part of the cycle is understood as the beginning and end. There are currently three options: 'lo' for L0-to-L0; 'p2p' for peak-to-peak; and 't2t' for trough-to-trough
- amplitude: Amplitude of length change (assuming sine wave trajectory)
- phase: Phase of the oscillatory cycle (in percent) at which stimulation occurred. Somewhat experimental, please use with caution
- position_inverted: Logical; whether position inversion has been applied)

`units`	The units of measurement for each column in the object after running this function. See Warning
`sample_frequency`	Frequency at which samples were collected
`header`	Additional information from the header
`units_table`	Units from each Channel of the original ddf file
`protocol_table`	Protocol in tabular format; taken from the original ddf file
`stim_table`	Specific info on stimulus protocol; taken from the original ddf file
`stimulus_pulses`	Number of sequential pulses within a stimulation train
`stimulus_offset`	Timing offset at which stimulus began
`gear_ratio`	Gear ratio applied by this function
`file_id`	Filename
`mtime`	Time at which file was last modified
`retained_cycles`	Which cycles were retained, as numerics
`summary`	Simple table showing work (in J) and net power (in W) for each cycle

Warning

Most systems we have encountered record Position data in millimeters and Force in millinewtons, and therefore this function assumes data are recorded in those units. Through a series of internal conversions, this function computes velocity in meters/sec, work in Joules, and power in Watts. If your raw data do not originate in millimeters and millinewtons, please transform your data accordingly and ignore what you see in the attribute `units`.

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

References

Josephson RK. 1985. Mechanical Power output from Striated Muscle during Cyclic Contraction. Journal of Experimental Biology 114: 493-512.

See Also

`read_ddf`, `read_analyze_wl`, `select_cycles`

Other data analyses: `isometric_timing`, `read_analyze_wl_dir`, `read_analyze_wl`

Other workloop functions: `fix_GR`, `get_wl_metadata`, `invert_position`, `read_analyze_wl_dir`, `read_analyze_wl`, `select_cycles`, `summarize_wl_trials`, `time_correct`

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf",
```

```
                              package = 'workloopR'),
                  phase_from_peak = TRUE)

# select cycles 3 through 5 via the peak-to-peak definition
wl_selected <- select_cycles(wl_dat, cycle_def = "p2p", keep_cycles = 3:5)

# run the analysis function and get the full object
wl_analyzed <- analyze_workloop(wl_selected, GR = 2)

# print methods give a short summary
print(wl_analyzed)

# summary provides a bit more detail
summary(wl_analyzed)

# run the analysis but get the simplified version
wl_analyzed_simple <- analyze_workloop(wl_selected, simplify = TRUE, GR = 2)
```

`as_muscle_stim` *Create your own muscle_stim object*

Description

For use when data are not stored in .ddf format and you would like to create a `muscle_stim` object that can be used by other workloopR functions.

Usage

```
as_muscle_stim(x, type, sample_frequency, ...)
```

Arguments

- `x` — A `data.frame`. See Details for how it should be organized.
- `type` — Experiment type; must be one of: "workloop", "tetanus", or "twitch."
- `sample_frequency` — Numeric value of the frequency at which samples were recorded; must be in Hz. Please format as numeric, e.g. `10000` works but `10000 Hz` does not
- `...` — Additional arguments that can be passed in as attributes. See Details.

Details

`muscle_stim` objects, which are required by (nearly) all workloopR functions, are automatically created via `read_ddf()`. Should you have data that are stored in a format other than .ddf, use this function to create your own object of class `muscle_stim`.

The input `x` must be a `data.frame` that contains time series of numeric data collected from an experiment. Each row must correspond to a sample, and these columns (exact title matches) must be included:
"Time" - time, recorded in seconds
"Position" - instantaneous position of the muscle, preferably in millimeters
"Force" - force, preferably in millinewtons

"Stim" - whether stimulation has occurred. All entries must be either 0 (no stimulus) or 1 (stimulus occurrence).

Additional arguments can be provided via .... For all experiment types, the following attributes are appropriate:
"units","header", "units_table", "protocol_table", "stim_table", "stimulus_pulses", "stimulus_offset", "stimulus_width", "gear_ratio", "file_id", or "mtime".

Please ensure that further attributes are appropriate to your experiment type.

For workloops, these include: "stimulus_frequency", "cycle_frequency", "total_cycles", "cycle_def", "amplitude", "phase", and "position_inverted"

For twitches or tetanic trials: "stimulus_frequency", and "stimulus_length"

Value

An object of class `workloop`, `twitch`, or `tetanus`, all of which inherit class `muscle_stim`. These objects behave like `data.frames` in most situations but also store metadata from the ddf as attributes.

The `muscle_stim` object's columns contain:

`Time`	Time
`Position`	Length change of the muscle, uncorrected for gear ratio
`Force`	Force, uncorrected for gear ratio
`Stim`	When stimulation occurs, on a binary scale

In addition, the following information is stored in the `data.frame`'s attributes:

`sample_frequency`	Frequency at which samples were collected
`pulses`	Number of sequential pulses within a stimulation train
`total_cycles_lo`	Total number of oscillatory cycles (assuming sine wave trajectory) that the muscle experienced. Cycles are defined with respect to initial muscle length (L0-to-L0 as opposed to peak-to-peak).
`amplitude`	amplitude of length change (again, assuming sine wave trajectory)
`cycle_frequency`	Frequency of oscillations (again, assuming sine wave trajectory)
`units`	The units of measurement for each column in the `data.frame`. This might be the most important attribute so please check that it makes sense!

Author(s)

Shreeram Senthivasan

See Also

`read_ddf`

Other data import functions: `get_wl_metadata`, `read_analyze_wl_dir`, `read_analyze_wl`, `read_ddf_dir`, `read_ddf`

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf", package = 'workloopR'))

# see how this object is organized - this will give you a sense
# of how your inputs to `as_muscle_stim()` should be arranged:
#head(wl_dat)
#str(wl_dat)
# formatting of attributes:
#names(attributes(wl_dat))
#str(attributes(wl_dat))
```

`fix_GR` *Adjust for the gear ratio of a motor arm*

Description

Fix a discrepancy between the gear ratio of the motor arm used and the gear ratio recorded by software.

Usage

```
fix_GR(x, GR = 1)
```

Arguments

`x`	A `muscle_stim` object
`GR`	Gear ratio, set to 1 by default

Details

The `muscle_stim` object can be of any type, including `workloop`, `twitch`, or `tetanus`.

If you have manually constructed the object via `as_muscle_stim()`, the `muscle_stim` object should have columns as follows:
`Position`: length change of the muscle;
`Force`: force

Value

An object of the same class(es) as the input (`x`). The function will multiply `Position` by (1/GR) and multiply `Force` by GR, returning an object with new values in `$Position` and `$Force`. Other columns and attributes are welcome and will simply be passed on unchanged into the resulting object.

Author(s)

Vikram B. Baliga

See Also

analyze_workloop, read_analyze_wl, read_analyze_wl_dir

Other data transformations: invert_position, select_cycles

Other workloop functions: analyze_workloop, get_wl_metadata, invert_position, read_analyze_wl_dir, read_analyze_wl, select_cycles, summarize_wl_trials, time_correct

Other twitch functions: invert_position, isometric_timing

Other tetanus functions: invert_position

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf",
                              package = 'workloopR'),
                  phase_from_peak = TRUE)

# apply a gear ratio correction of 2
# this will multiply Force by 2 and divide Position by 2
wl_fixed <- fix_GR(wl_dat, GR = 2)

# quick check:
max(wl_fixed$Force)/max(wl_dat$Force)       #5592.578 / 2796.289 = 2
max(wl_fixed$Position)/max(wl_dat$Position) #1.832262 / 3.664524 = 0.5
```

---

get_wl_metadata *Get file info for a sequence of experiment files*

Description

Grab metadata from files stored in the same folder (e.g. a sequence of trials in an experiment).

Usage

```
get_wl_metadata(filepath, pattern = "*.ddf")
```

Arguments

filepath	Path where files are stored. Should be in the same folder.
pattern	Regex pattern for identifying relevant files in the filepath.

Details

If several files (e.g. successive trials from one experiment) are stored in one folder, use this function to obtain metadata in a list format. Runs `file.info` from base R to extract info from files.

This function is not truly considered to be part of the batch analysis pipeline; see `read_analyze_wl_dir()` for a similar function that not only grabs metadata but also imports & analyzes files. Instead, `get_wl_metadata()` is meant to be a handy function to investigate metadata issues that arise if running `read_analyze_wl_dir()` goes awry.

Unlike read_analyze_wl_dir(), this function does not necessarily need files to all be work loops. Any filetype is welcome (as long as the Regex pattern argument makes sense).

Author(s)

Vikram B. Baliga

See Also

summarize_wl_trials

Other data import functions: as_muscle_stim, read_analyze_wl_dir, read_analyze_wl, read_ddf_dir, read_ddf

Other workloop functions: analyze_workloop, fix_GR, invert_position, read_analyze_wl_dir, read_analyze_wl, select_cycles, summarize_wl_trials, time_correct

Other batch analyses: read_analyze_wl_dir, summarize_wl_trials, time_correct

Examples

```
library(workloopR)

# get file info for files included with workloopR
wl_meta <- get_wl_metadata(system.file("extdata/wl_duration_trials",
                                       package = 'workloopR'))

# or on your own directory
#my_meta <- get_wl_metadata("./my/file/path/")
```

---

invert_position   *Invert the position data*

Description

Multiply instantaneous position by -1.

Usage

```
invert_position(x)
```

Arguments

x	A muscle_stim object

Details

The muscle_stim object can be of any type, including workloop, twitch, or tetanus.

If you have manually constructed the object via as_muscle_stim(), the muscle_stim object should have a column entitled Position. Other columns and attributes are welcome and will be passed along unchanged.

Value

A workloop object with inverted position. The position_inverted attribute is set to TRUE and all others are retained.

Author(s)

Vikram B. Baliga

See Also

Other data transformations: fix_GR, select_cycles

Other workloop functions: analyze_workloop, fix_GR, get_wl_metadata, read_analyze_wl_dir, read_analyze_wl, select_cycles, summarize_wl_trials, time_correct

Other twitch functions: fix_GR, isometric_timing

Other tetanus functions: fix_GR

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf",
                              package = 'workloopR'),
                  phase_from_peak = TRUE)

# invert the sign of Position
wl_fixed <- invert_position(wl_dat)

# quick check:
max(wl_fixed$Position)/min(wl_dat$Position) # -1
```

---

isometric_timing — *Compute timing and magnitude of force in isometric trials*

Description

Calculate timing and magnitude of force at stimulation, peak force, and various parts of the rising (force development) and relaxation (falling) phases of the twitch.

Usage

```
isometric_timing(x, rising = c(10, 90), relaxing = c(90, 50))
```

Arguments

x	A muscle_stim object that contains data from an isometric twitch trial, ideally created via read_ddf.
rising	Set points of the rising phase to be described. By default: 10% and 90%.
relaxing	Set points of the relaxation phase to be described. By default: 90% and 50%.

Details

The `data.frame` (x) must have time series data organized in columns. Generally, it is preferred that you use a `muscle_stim` object imported by `read_ddf()`.

The `rising` and `relaxing` arguments allow for the user to supply numeric vectors of any length. By default, these arguments are `rising = c(10,90)` and `relaxing = c(90,50)`. Numbers in each of these correspond to percent values and capture time and force at that percent of the corresponding curve. These values can be replaced by those that the user specifies and do not necessarily need to have length = 2. But please note that 0 and 100 should not be used, e.g. `rising = seq(10,90,5)` works, but `rising = seq(0,100,5)` does not.

Value

A `data.frame` with the following metrics as columns:

`file_ID`	File ID
`time_stim`	Time between beginning of data collection and when stimulation occurs
`force_stim`	Magnitude of force at the onset of stimulation
`time_peak`	Absolute time of peak force, i.e. time between beginning of data collection and when peak force occurs
`force_peak`	Magnitude of peak force
`time_rising_X`	Time between beginning of data collection and X% of force development
`force_rising_X`	Magnitude of force at X% of force development
`time_relaxing_X`	Time between beginning of data collection and X% of force relaxation
`force_relaxing_X`	Magnitude of force at X% of relaxation

Author(s)

Vikram B. Baliga

References

Ahn AN, and Full RJ. 2002. A motor and a brake: two leg extensor muscles acting at the same joint manage energy differently in a running insect. Journal of Experimental Biology 205, 379-389.

See Also

Other data analyses: `analyze_workloop`, `read_analyze_wl_dir`, `read_analyze_wl`

Other twitch functions: `fix_GR`, `invert_position`

Examples

```
library(workloopR)

# import the twitch.ddf file included in workloopR
twitch_dat <-read_ddf(system.file("extdata", "twitch.ddf",
                                  package = 'workloopR'))

# run isometric_timing() to get info on twitch kinetics
```

```
# we'll use different set points than the defaults
analyze_twitch <- isometric_timing(twitch_dat,
                                rising = c(25, 50, 75),
                                relaxing = c(75, 50, 25))

# see the results
analyze_twitch
```

`read_analyze_wl` *All-in-one import function for work loop files*

Description

`read_analyze_wl()` is an all-in-one function to read in a work loop file, select cycles, and compute work and power output.

Usage

```
read_analyze_wl(filename, ...)
```

Arguments

`filename`	A .ddf file that contains data from a single workloop experiment
`...`	Additional arguments to be passed to `read_ddf()`, `select_cycles()`, or `analyze_workloop()`.

Details

Please be careful with units! See Warnings below. This function combines `read_ddf()` with `select_cycles()` and then ultimately `analyze_workloop()` into one handy function.

As detailed in these three functions, possible arguments include:
`cycle_def` - used to specify which part of the cycle is understood as the beginning and end. There are currently three options: 'lo' for L0-to-L0; 'p2p' for peak-to-peak; and 't2t' for trough-to-trough
`bworth_order` - Filter order for low-pass filtering of `Position` via `signal::butter` prior to finding peak lengths. Default: 2.
`bworth_freq` - Critical frequency (scalar) for low-pass filtering of `Position` via `signal::butter` prior to finding peak lengths. Default: 0.05.
`keep_cycles` - Which cycles should be retained. Default: 4:6.
`GR` - Gear ratio. Default: 1.
`M` - Velocity multiplier used to positivize velocity; should be either -1 or 1. Default: -1.
`vel_bf` - Critical frequency (scalar) for low-pass filtering of velocity via `signal::butter`. Default: 0.05.

The gear ratio (GR) and velocity multiplier (M) parameters can help correct for issues related to the magnitude and sign of data collection. By default, they are set to apply no gear ratio adjustment and to positivize velocity. Instantaneous velocity is often noisy and the `vel_bf` parameter allows for low-pass filtering of velocity data. See `signal::butter()` and `signal::filtfilt()` for details of how filtering is achieved.

Value

The function returns a `list` of class `analyzed_workloop` that provides instantaneous velocity, a smoothed velocity, and computes work, instantaneous power, and net power from a work loop experiment. All data are organized by the cycle number and important metadata are stored as Attributes.

Within the `list`, each entry is labeled by cycle and includes:

- `Time` Time, in sec
- `Position` Length change of the muscle, corrected for gear ratio, in mm
- `Force` Force, corrected for gear ratio, in mN
- `Stim` When stimulation occurs, on a binary scale
- `Cycle` Cycle ID, as a letter
- `Inst_velocity` Instantaneous velocity, computed from `Position` change, reported in meters/sec
- `Filt_velocity` Instantaneous velocity, after low-pass filtering, again in meter/sec
- `Inst_Power` Instantaneous power, a product of `Force` and `Filt_velocity`, reported in J
- `Percent_of_Cycle` The percent of that particular cycle which has elapsed

In addition, the following information is stored in the `analyzed_workloop` object's attributes:

- `stimulus_frequency` Frequency at which stimulus pulses occurred
- `cycle_frequency` Frequency of oscillations (assuming sine wave trajectory)
- `total_cycles` Total number of oscillatory cycles (assuming sine wave trajectory) that the muscle experienced.
- `cycle_def` Specifies what part of the cycle is understood as the beginning and end. There are currently three options: 'lo' for L0-to-L0; 'p2p' for peak-to-peak; and 't2t' for trough-to-trough
- `amplitude` Amplitude of length change (assuming sine wave trajectory)
- `phase` Phase of the oscillatory cycle (in percent) at which stimulation occurred. Somewhat experimental, please use with caution
- `position_inverted` Logical; whether position inversion has been applied)
- `units` The units of measurement for each column in the object after running this function. See Warning
- `sample_frequency` Frequency at which samples were collected
- `header` Additional information from the header
- `units_table` Units from each Channel of the original ddf file
- `protocol_table` Protocol in tabular format; taken from the original ddf file
- `stim_table` Specific info on stimulus protocol; taken from the original ddf file
- `stimulus_pulses` Number of sequential pulses within a stimulation train
- `stimulus_offset` Timing offset at which stimulus began

- `gear_ratio` Gear ratio applied by this function
- `file_id` Filename
- `mtime` Time at which file was last modified
- `retained_cycles` Which cycles were retained, as numerics
- `summary` Simple table showing work (in J) and net power (in W) for each cycle

Warning

Most systems we have encountered record Position data in millimeters and Force in millinewtons, and therefore this function assumes data are recorded in those units. Through a series of internal conversions, this function computes velocity in meters/sec, work in Joules, and power in Watts. If your raw data do not originate in millimeters and millinewtons, please transform your data accordingly and ignore what you see in the attribute `units`.

Author(s)

Vikram B. Baliga

References

Josephson RK. 1985. Mechanical Power output from Striated Muscle during Cyclic Contraction. Journal of Experimental Biology 114: 493-512.

See Also

`read_ddf`, `select_cycles` `analyze_workloop`

Other data analyses: `analyze_workloop`, `isometric_timing`, `read_analyze_wl_dir`

Other data import functions: `as_muscle_stim`, `get_wl_metadata`, `read_analyze_wl_dir`, `read_ddf_dir`, `read_ddf`

Other workloop functions: `analyze_workloop`, `fix_GR`, `get_wl_metadata`, `invert_position`, `read_analyze_wl_dir`, `select_cycles`, `summarize_wl_trials`, `time_correct`

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR and analyze with
# a gear ratio correction of 2 and cycle definition of peak-to-peak
wl_dat <- read_analyze_wl(system.file("extdata", "workloop.ddf",
                                      package = 'workloopR'),
                          phase_from_peak = TRUE,
                          GR = 2, cycle_def = "p2p")
```

`read_analyze_wl_dir` *Read and analyze work loop files from a directory*

Description

All-in-one function to import multiple workloop .ddf files from a directory, sort them by mtime, analyze them, and store the resulting objects in an ordered list.

Usage

```
read_analyze_wl_dir(filepath, pattern = "*.ddf", sort_by = "mtime",
  ...)
```

Arguments

`filepath`	Directory in which files are located
`pattern`	Regular expression used to specify files of interest. Defaults to all .ddf files within filepath.
`sort_by`	Metadata by which files should be sorted to be in the correct run order. Defaults to `mtime`, which is time of last modification of files.
`...`	Additional arguments to be passed to `read_analyze_wl()`, `analyze_workloop()`, `select_cycles()`, or `read_ddf()`.

Details

Work loop data files will be imported and then arranged in the order in which they were run (assuming run order is reflected in `mtime`). Chiefly used in conjunction with `summarize_wl_trials()` and `time_correct()` if time correction is desired.

Value

A list containing `analyzed_workloop` objects, one for each file that is imported and subsequently analyzed. The list is sorted according to the `sort_by` parameter, which by default uses the time of last modification of each file's contents (mtime).

Warning

Most systems we have encountered record Position data in millimeters and Force in millinewtons, and therefore this function assumes data are recorded in those units. Through a series of internal conversions, this function computes velocity in meters/sec, work in Joules, and power in Watts. If your raw data do not originate in millimeters and millinewtons, please transform your data accordingly and ignore what you see in the attribute `units`.

Author(s)

Shreeram Senthivasan

References

Josephson RK. 1985. Mechanical Power output from Striated Muscle during Cyclic Contraction. Journal of Experimental Biology 114: 493-512.

See Also

read_analyze_wl, get_wl_metadata, summarize_wl_trials, time_correct

Other data analyses: analyze_workloop, isometric_timing, read_analyze_wl

Other data import functions: as_muscle_stim, get_wl_metadata, read_analyze_wl, read_ddf_dir, read_ddf

Other workloop functions: analyze_workloop, fix_GR, get_wl_metadata, invert_position, read_analyze_wl, select_cycles, summarize_wl_trials, time_correct

Other batch analyses: get_wl_metadata, summarize_wl_trials, time_correct

Examples

```
library(workloopR)

# batch read and analyze files included with workloopR
analyzed_wls <- read_analyze_wl_dir(system.file("extdata/wl_duration_trials",
                                                package = 'workloopR'),
                                    phase_from_peak = TRUE,
                                    cycle_def = "p2p", keep_cycles = 2:4)

# or on your own directory
#my_analyzed_wls <- read_analyze_wl_dir("./my/file/path/")
```

---

read_ddf — *Import work loop or isometric data from .ddf files*

Description

read_ddf reads in workloop, twitch, or tetanus experiment data from .ddf files.

Usage

```
read_ddf(filename, file_id = NA, rename_cols = list(c(2, 3),
  c("Position", "Force")), skip_cols = 4:11, phase_from_peak = FALSE,
  ...)
```

Arguments

filename	A .ddf file that contains data from a single workloop, twitch, or tetanus experiment
file_id	A string identifying the experiment. The filename is used by default.
rename_cols	List consisting of a vector of indices of columns to rename and a vector of new column names. See Details.
skip_cols	Numeric vector of column indices to skip. See Details.
phase_from_peak	Logical, indicating whether percent phase of stimulation should be recorded relative to peak length or relative to L0 (default)
...	Additional arguments passed to/from other functions that work with read_ddf()

Details

Read in a .ddf file that contains data from an experiment. If position and force do not correspond to columns 2 and 3 (respectively), replace "2" and "3" within `rename_cols` accordingly. Similarly, `skip_cols = 4:11` should be adjusted if more than 11 columns are present and/or columns 4:11 contain important data.

Please note that there is no correction for gear ratio or further manipulation of data. See `fix_GR` to adjust gear ratio. Gear ratio can also be adjusted prior to analyses within the `analyze_workloop()` function, the data import all-in-one function `read_analyze_wl()`, or the batch analysis all-in-one `read_analyze_wl_dir()`.

Please also note that organization of data within the .ddf file is assumed to conform to that used by Aurora Scientific's Dynamic Muscle Control and Analysis Software. YMMV if using a .ddf file from another source. The `as_muscle_stim()` function can be used to generate `muscle_stim` objects if data are imported via another function. Please feel free to contact us with any issues or requests.

Value

An object of class `workloop`, `twitch`, or `tetanus`, all of which inherit class `muscle_stim`. These objects behave like `data.frames` in most situations but also store metadata from the ddf as attributes.

The `muscle_stim` object's columns contain:

- `Time` Time
- `Position` Length change of the muscle, uncorrected for gear ratio
- `Force` Force, uncorrected for gear ratio
- `Stim` When stimulation occurs, on a binary scale

In addition, the following information is stored in the `data.frame`'s attributes:

- `sample_frequency` Frequency at which samples were collected
- `pulses` Number of sequential pulses within a stimulation train
- `total_cycles_lo` Total number of oscillatory cycles (assuming sine wave trajectory) that the muscle experienced. Cycles are defined with respect to initial muscle length (L0-to-L0 as opposed to peak-to-peak).
- `amplitude` amplitude of length change (again, assuming sine wave trajectory)
- `cycle_frequency` Frequency of oscillations (again, assuming sine wave trajectory)
- `units` The units of measurement for each column in the `data.frame`. This might be the most important attribute so please check that it makes sense!

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

See Also

Other data import functions: `as_muscle_stim`, `get_wl_metadata`, `read_analyze_wl_dir`, `read_analyze_wl`, `read_ddf_dir`

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf",
                              package = 'workloopR'),
                  phase_from_peak = TRUE)

# or import your own file
#my_dat <- read_ddf("./my/file/path/myfile.ddf")
```

---

read_ddf_dir *Import a batch of work loop or isometric data files from a directory*

Description

Uses `read_ddf()` to read in workloop, twitch, or tetanus experiment data from multiple .ddf files.

Usage

```
read_ddf_dir(filepath, pattern = "*.ddf", sort_by = "mtime", ...)
```

Arguments

`filepath`	Path where files are stored. Should be in the same folder.
`pattern`	Regex pattern for identifying relevant files in the filepath.
`sort_by`	Metadata by which files should be sorted to be in the correct run order. Defaults to `mtime`, which is time of last modification of files.
`...`	Additional arguments to be passed to `read_ddf()`.

Details

Read in a .ddf file that contains data from an experiment. If position and force do not correspond to columns 2 and 3 (respectively), replace "2" and "3" within `rename_cols` accordingly. Similarly, `skip_cols = 4:11` should be adjusted if more than 11 columns are present and/or columns 4:11 contain important data.

Please note that there is no correction for gear ratio or further manipulation of data. See `fix_GR` to adjust gear ratio. Gear ratio can also be adjusted prior to analyses within the `analyze_workloop()` function, the data import all-in-one function `read_analyze_wl()`, or the batch analysis all-in-one `read_analyze_wl_dir()`.

Please also note that organization of data within the .ddf file is assumed to conform to that used by Aurora Scientific's Dynamic Muscle Control and Analysis Software. YMMV if using a .ddf file from another source. The `as_muscle_stim()` function can be used to generate `muscle_stim` objects if data are imported via another function. Please feel free to contact us with any issues or requests.

Value

A list of objects of class `workloop`, `twitch`, or `tetanus`, all of which inherit class `muscle_stim`. These objects behave like `data.frames` in most situations but also store metadata from the ddf as attributes.

Each `muscle_stim` object's columns contain:

- `Time` Time
- `Position` Length change of the muscle, uncorrected for gear ratio
- `Force` Force, uncorrected for gear ratio
- `Stim` When stimulation occurs, on a binary scale

In addition, the following information is stored in each `data.frame`'s attributes:

- `sample_frequency` Frequency at which samples were collected
- `pulses` Number of sequential pulses within a stimulation train
- `total_cycles_lo` Total number of oscillatory cycles (assuming sine wave trajectory) that the muscle experienced. Cycles are defined with respect to initial muscle length (L0-to-L0 as opposed to peak-to-peak).
- `amplitude` amplitude of length change (again, assuming sine wave trajectory)
- `cycle_frequency` Frequency of oscillations (again, assuming sine wave trajectory)
- `units` The units of measurement for each column in the `data.frame`. This might be the most important attribute so please check that it makes sense!

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

See Also

Other data import functions: `as_muscle_stim`, `get_wl_metadata`, `read_analyze_wl_dir`, `read_analyze_wl`, `read_ddf`

Examples

```
library(workloopR)

# import a set of twitch .ddf files included in workloopR
workloop_dat <-read_ddf_dir(system.file("extdata/wl_duration_trials",
                 package = 'workloopR'))

# or import your own file
#my_dat <- read_ddf_dir("./my/file/path/")
```

`select_cycles` *Select cycles from a work loop object*

Description

Retain data from a work loop experiment based on position cycle

Usage

```
select_cycles(x, cycle_def, keep_cycles = 4:6, bworth_order = 2,
  bworth_freq = 0.05, ...)
```

Arguments

`x`	A `workloop` object (see Details for how it should be organized)
`cycle_def`	A string specifying how cycles should be defined; one of: "lo", "p2p", or "t2t". See Details more info
`keep_cycles`	The indices of the cycles to keep. Include 0 to keep data identified as being outside complete cycles
`bworth_order`	Filter order for low-pass filtering of `Position` via `signal::butter()` prior to finding L0
`bworth_freq`	Critical frequency (scalar) for low-pass filtering of `Position` via `signal::butter()` prior to finding L0
`...`	Additional arguments passed to/from other functions that make use of `select_cycles()`

Details

`select_cycles()` subsets data from a workloop trial by position cycle. The `cycle_def` argument is used to specify which part of the cycle is understood as the beginning and end. There are currently three options:
'lo' for L0-to-L0;
'p2p' for peak-to-peak; and
't2t' for trough-to-trough

Peaks are identified using `pracma::findpeaks()`. L0 points on the rising edge are found by finding the midpoints between troughs and the following peak. However the first and last extrema and L0 points may be misidentified by this method. Please plot your `Position` cycles to ensure the edge cases are identified correctly.

The `keep_cycles` argument is used to determine which cycles (as defined by `cycle_def` should be retained in the final dataset. Zero is the index assigned to all data points that are determined to be outside a complete cycle.

The `muscle_stim` object (`x`) must be a `workloop`, preferably read in by one of our data import functions. Please see documentation for `as_muscle_stim()` if you need to manually construct a `muscle_stim` object from another source.

Value

A `workloop` object with rows subsetted by the chosen position cycles. A `Cycle` column is appended to denote which cycle each time point is associated with. Finally, all attributes from the input `workloop` object are retained and one new attribute is added to record which cycles from the original data were retained.

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

See Also

analyze_workloop, read_analyze_wl, read_analyze_wl_dir

Other data transformations: fix_GR, invert_position

Other workloop functions: analyze_workloop, fix_GR, get_wl_metadata, invert_position, read_analyze_wl_dir, read_analyze_wl, summarize_wl_trials, time_correct

Examples

```
library(workloopR)

# import the workloop.ddf file included in workloopR
wl_dat <-read_ddf(system.file("extdata", "workloop.ddf",
                              package = 'workloopR'),
                  phase_from_peak = TRUE)

# select cycles 3 through 5 via the peak-to-peak definition
wl_selected <- select_cycles(wl_dat, cycle_def = "p2p", keep_cycles = 3:5)

# are only three cycles present, running peak-to-peak?
#plot(wl_selected$Position)

# are the cycles of (approximately) the same length?
summary(as.factor(wl_selected$Cycle))
```

---

summarize_wl_trials *Summarize work loop files*

Description

Summarize important info from work loop files stored in the same folder (e.g. a sequence of trials in an experiment) including experimental parameters, run order, and mtime.

Usage

```
summarize_wl_trials(wl_list)
```

Arguments

wl_list	List of analyzed_workloop objects, preferably one created by read_analyze_wl_dir().

Details

If several files (e.g. successive trials from one experiment) are stored in one folder, use this function to obtain summary stats and metadata and other parameters. This function requires a list of analyze_workloop objects, which can be readily obtained by first running read_analyze_wl_dir() on a specified directory.

Value

A `data.frame` of information about the collection of workloop files. Columns include:

- `File_ID` Name of the file
- `Cycle_Frequency` Frequency of Position change
- `Amplitude` amplitude of Position change
- `Phase` Phase of the oscillatory cycle (in percent) at which stimulation occurred. Somewhat experimental, please use with caution
- `Stimulus_Pulses` Number of stimulation pulses
- `mtime` Time at which file's contents were last changed (`mtime`)
- `Mean_Work` Mean work output from the selected cycles
- `Mean_Power` Net power output from the selected cycles

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

References

Josephson RK. 1985. Mechanical Power output from Striated Muscle during Cyclic Contraction. Journal of Experimental Biology 114: 493-512.

See Also

`read_analyze_wl_dir`, `get_wl_metadata`, `time_correct`

Other workloop functions: `analyze_workloop`, `fix_GR`, `get_wl_metadata`, `invert_position`, `read_analyze_wl_dir`, `read_analyze_wl`, `select_cycles`, `time_correct`

Other batch analyses: `get_wl_metadata`, `read_analyze_wl_dir`, `time_correct`

Examples

```
library(workloopR)

# batch read and analyze files included with workloopR
analyzed_wls <- read_analyze_wl_dir(system.file("extdata/wl_duration_trials",
                                                package = 'workloopR'),
                                    phase_from_peak = TRUE,
                                    cycle_def = "p2p",
                                    keep_cycles = 2:4
                                    )

# now summarize
summarized_wls <- summarize_wl_trials(analyzed_wls)

# or on your own directory
#my_meta <- read_analyze_wl_dir("./my/file/path/")
#my_summaries <- summarize_wl_trials(my_meta)
```

time_correct *Time correction for work loop experiments*

Description

Correct for potential degradation of muscle over time.

Usage

```
time_correct(x)
```

Arguments

x A data.frame with summary data, e.g. an object created by summarize_wl_trials().

Details

This function assumes that across a batch of successive trials, the stimulation parameters for the first and final trials are identical. If not, DO NOT USE. Decline in power output is therefore assumed to be a linear function of time. Accordingly, the difference between the final and first trial's (absolute) power output is used to 'correct' trials that occur in between, with explicit consideration of run order and time elapsed (via mtime). A similar correction procedure is applied to work.

Value

A data.frame that additionally contains:

Time_Corrected_Work
: Time corrected work output, transformed from $Mean_Work

Time_Corrected_Power
: Time corrected net power output, transformed from $Mean_Power

And new attributes:

power_difference
: Difference in mass-specific net power output between the final and first trials.

time_difference
: Difference in mtime between the final and first trials.

time_correction_rate
: Overall rate; power_difference divided by time_difference.

Author(s)

Vikram B. Baliga and Shreeram Senthivasan

See Also

summarize_wl_trials

Other workloop functions: analyze_workloop, fix_GR, get_wl_metadata, invert_position, read_analyze_wl_dir, read_analyze_wl, select_cycles, summarize_wl_trials

Other batch analyses: get_wl_metadata, read_analyze_wl_dir, summarize_wl_trials

Examples

```
library(workloopR)

# batch read and analyze files included with workloopR
analyzed_wls <- read_analyze_wl_dir(system.file("extdata/wl_duration_trials",
                                                package = 'workloopR'),
                                    phase_from_peak = TRUE,
                                    cycle_def = "p2p", keep_cycles = 2:4)

# now summarize
summarized_wls <- summarize_wl_trials(analyzed_wls)


# mtimes within the package are not accurate, so we'll supply
# our own vector of mtimes
summarized_wls$mtime <- read.csv(
                          system.file(
                            "extdata/wl_duration_trials/ddfmtimes.csv",
                            package="workloopR"))$mtime

# now time correct
timecor_wls <- time_correct(summarized_wls)
timecor_wls

# or on your own directory
#my_meta <- read_analyze_wl_dir("./my/file/path/")
#my_summaries <- summarize_wl_trials(my_meta)
#my_timecors <- time_correct(my_summaries)
```

`trapezoidal_integration` *Approximate the definite integral via the trapezoidal rule*

Description

Mostly meant for internal use in our analysis functions, but made available for other use cases. Accordingly, it does not strictly rely on objects of class `muscle_stim`.

Usage

```
trapezoidal_integration(x, f)
```

Arguments

x	a variable, e.g. vector of positions
f	integrand, e.g. vector of forces

Details

In the functions `analyze_workloop()`, `read_analyze_wl()` , and `read_analyze_wl_dir()`, work is calculated as the difference between the integral of the upper curve and the integral of the lower curve of a work loop.

Author(s)

Vikram B. Baliga

References

Atkinson, Kendall E. (1989), An Introduction to Numerical Analysis (2nd ed.), New York: John Wiley & Sons

See Also

analyze_workloop, read_analyze_wl, read_analyze_wl_dir

Examples

```
# create a circle centered at (x = 10, y = 20) with radius 2
t <- seq(0, 2*pi, length = 1000)
coords <- t(rbind(10 + sin(t)*2, 20 + cos(t)*2))

# sanity check: does it look like a circle?
#plot(coords, asp = 1)

# use the function to get the area
trapezoidal_integration(coords[,1],coords[,2])

# does it match (pi * r^2)?
3.14159265358 * (2^2) # very close
```

Index