Package 'forceplate'

July 22, 2025

Title Processing Force-Plate Data

Version 1.1-4

Description Process raw force-plate data (txt-files) by segmenting them into trials and, if needed, calculating (user-defined) descriptive statistics of variables for user-defined time bins (relative to trigger onsets) for each trial. When segmenting the data a baseline correction, a filter, and a data imputation can be applied if needed. Experimental data can also be processed and combined with the segmented force-plate data. This procedure is suggested by Johannsen et al. (2023) <doi:10.6084/m9.figshare.22190155> and some of the options (e.g., choice of low-pass filter) are also suggested by Winter (2009) <doi:10.1002/9780470549148>.

Imports data.table, signal, stats, stringi

Suggests curl

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/RaphaelHartmann/forceplate

BugReports https://github.com/RaphaelHartmann/forceplate/issues

RoxygenNote 7.3.2

NeedsCompilation no

Author Raphael Hartmann [aut, cre] (ORCID: <https://orcid.org/0000-0003-4686-9329>), Anton Koger [aut, ctb] (ORCID: <https://orcid.org/0009-0004-6906-5184>), Leif Johannsen [ctb]

Maintainer Raphael Hartmann <raphael.hartmann@protonmail.com>

Repository CRAN

Date/Publication 2025-03-19 15:00:05 UTC

Contents

`combine_data` *Combine Data Tables*

Description

Combine two `data.tables`, either two force-plate data, two exeperimental data, or one force-plate and one experimental data.

Usage

```
combine_data(
  dt1,
  dt2,
  by = list(subj = "subj", block = "block", trial = "trial"),
  continuous = FALSE
)
```

Arguments

`dt1`	A `data.table` of the class `fp.segm`, `fp.tl`, or `exp.prep`.
`dt2`	A `data.table` of the class `fp.segm`, `fp.tl`, or `exp.prep`. Make sure the two data.table have either the same number of rows or the same columns.
`by`	A list of three variable names in the experimental data that reflect the subj (subject number), block (block number), and trial (trial number) in the force-plate data. This argument is only necessary for combining experimental data with force-plate data.
`continuous`	A logical value. Default is `FALSE`, meaning the variable for the trials in the used experimental data.table counts for each block separately, that is in each block it counts from 1 to the number of trials in that block. If `TRUE` it is assumed that the trials are counted from 1 to the total number of trials of a subject.

Value

A `data.table` either of the same class as `dt1` and `dt2`, if they share the same class, or of the class `dt.comb`.

Author(s)

Raphael Hartmann & Anton Koger

prep_exp_data *Prepare Experimental Data*

Description

Processing the experimental data by removing unnecessary variables and removing rows in the data that are not trials. The output is a `data.table`.

Usage

```
prep_exp_data(
  filenames,
  na.strings = c(",,", "[]", "None"),
  excl.vars = NULL,
  blacklist.vars = NULL,
  whitelist.vars = NULL,
  sort = TRUE
)
```

Arguments

`filenames`	A (vector of) character(s) providing the raw experimental data file name(s). Files can be .txt, .csv, or any common type.
`na.strings`	A (vector of) character(s) naming the strings that should be treated as NA.
`excl.vars`	A (vector of) number(s) or character(s) providing the column number(s) or name(s) of the data which will be used for spotting rows that are not trials, that is, rows that are NA in each of the columns `excl.vars`.
`blacklist.vars`	A (vector of) number(s) or character(s) providing the column number(s) or name(s) of variables to be deleted from the data. NULL means no variable will be deleted.
`whitelist.vars`	A (vector of) number(s) or character(s) providing the column number(s) or name(s) of variables to be kept in the data. All others will be deleted. NULL means all variables will be kept.
`sort`	TRUE or FALSE. If TRUE the data will be sorted by subject number and block number.

Value

A `data.table` of the class `exp.prep`.

Author(s)

Raphael Hartmann & Anton Koger

Examples

```
# Using example data from github which requires internet
if (curl::has_internet()) {
  url <- paste0("https://raw.githubusercontent.com/RaphaelHartmann/forceplate/",
                "main/data/subj13_exp_data.csv")

  # Safe download, handling potential errors
  tryCatch({
   filenames <- tempfile(pattern = c("subj13_exp_data"), tmpdir = tempdir(), fileext = ".csv")
    download.file(url, filenames)

    # prepare experimental data
    exp.dt <- prep_exp_data(filenames = filenames, excl.vars = 2:5)

    # Clean up
    unlink(filenames)
  }, error = function(e) {
    message("Failed to download data: ", e$message)
  })
}
```

`segment_fp_data` *Segmentation to Data per Trial*

Description

Processing force-plate data by segmenting the data in trials, baseline correct each trial (optional), applying a low-pass 4th order Butterworth filter (optional), labeling stimuli and response onsets in each trial, labeling conditions in each trial, and some more (see below). The output is a `data.table`.

Usage

```
segment_fp_data(
  filenames,
  n.trials,
  start.trigger,
  baseline.trigger,
  baseline.intv,
  stimulus.trigger.list,
  response.trigger.list,
  cond.trigger.list,
  skip = 19,
  sampling.freq = 1000,
  cutoff.freq = 10,
  control = NULL
)
```

Arguments

`filenames` A (vector of) character(s) providing the raw force-plate file name(s). Files should be in tab-delimited .txt-format.

`n.trials` A (vector of) number(s) providing the number of trial (per filename).

`start.trigger` A (vector of) number(s) providing the trigger(s) marking the beginning of a trial.

`baseline.trigger`
A (vector of) number(s) providing the trigger number(s) providing the reference for the interval for the baseline correction. For example, if set to 1 the onset of event with trigger 1 is used as zero point for the next argument (`baseline.intv`). Use 0 to indicate that you wish to use no baseline correction.

`baseline.intv` A vector of length 2 providing the lower and upper bounds of the interval that will be used as baseline interval (in milliseconds). For each measurement variable, the mean of the data points that fall into this interval will be subtracted from all data points within a trial.

`stimulus.trigger.list`
If a trial contains one task only, then a vector providing the trigger(s) marking the onset of the stimulus. If a trial contains more than one task, then a named list of vectors providing the trigger(s) marking the onset of stimuli. For example, `visual = c(4, 5), auditory = c(16, 17)`.

`response.trigger.list`
Same as `stimulus.trigger.list` but with trigger(s) marking the onset of responses. For example, `auditory = c(32, 33, 34, 36), visual = c(128, 129, 130, 132)`.

`cond.trigger.list`
A named list of vectors providing the trigger(s) marking the conditions.

`skip` A number giving the number of lines in the raw force-plate data to skip. In BioWare this is 19. The real data starts at line 20. Therefore the default value is set to 19.

`sampling.freq` A number giving the sampling frequency. Typically 1000 Hz.

`cutoff.freq` A number giving the cut-off frequency used for the low-pass 4th order Butterworth filter. If set to 0, no low-pass filter will be applied. Default is 10 Hz.

`control` List of additional options:

- `az0` Thickness parameter of the force plate in millimeter and negative. If this value (e.g., -41 for the Kistler force plate type 9260AA) is not 0 then the center of pressure in the x- and y-direction is calculated (like in Johannsen et al., 2023) using this value.
- `prepend.ms`: A number giving the number of milliseconds to prepend before the `start.trigger`. If this is not 0 then each trial will have additional `prepend.ms` milliseconds added at the beginning of each trial (potentially taken from the previous trial).
- `prepend.event`: Overwrite the events of the prepended frames with a (new) event number (integer). If set to `NULL` (default), the event numbers are kept as they are.

- `prepend.data`: Overwrite the data of the prepended frames with NA? If set to `FALSE` (default), the data is kept as it was (i.e., you might have data from the previous trial).
- `append.ms`: A number giving the number of milliseconds to append after each trial. If this is not 0 then each trial will have additional `append.ms` milliseconds added at the end of each trial (potentially taken from the next trial).
- `append.event`: Overwrite the events of the appended frames with a (new) event number (integer). If set to `NULL` (default), the event numbers are kept as they are. Note: this does not affect the last trial in each file, since this not followed directly by a trial.
- `append.data`: Overwrite the data of the appended frames with NA? If set to `FALSE` (default), the data is kept as it was (i.e., you might have data from the next trial). Note: this does not affect the last trial in each file, since this not followed directly by a trial.
- `sort` TRUE or FALSE. If TRUE (default) the data will be sorted by subject number and block number.
- `imputation` If you expect any NaNs in your raw force-plate data you might use this argument. Use either of the following options: "fmm", "periodic", "natural", "monoH.FC", or "hyman". These are method options in the `stats::spline()` function. Usually this option is not needed and the default (NULL) can be used.
- `variable.names` If used (i.e., not NULL), a named list of names. This will rename the variables of the force-plate data. There are three cases to consider:
  - the time variable: if your force-plate data does not contain a variable with the string "time" in it or you want to rename the time variable in the force-plate data, you can specify `time = "fp_time_name"` in the `variable.names` list. The left hand side must be an expression that contains the string "time". The right hand side must be the actual variable name in your raw force-plate data you want to replace.
  - the parallel-port pin variable: if your force-plate data does not contain variables with the string "pin" in it or you want to rename the pin variables in the force-plate data, you can specify `pin1 = "fp_pin1_name"`, `pin2 = "fp_pin2_name"`, `pin3 = "fp_pin3_name"`, and so on, in the `variable.names` list. The left hand side must be the string "pin" followed by a number. The right hand side must be the actual variable name in the force-plate data you want to replace.
  - measurement variables: if you wish to rename some measurement variables in your force-plate data you can do so. The only restriction being that the right hand side does not contain the strings "time" nor "pin". For example `y_Force = "Fy"` is allowed. But we recommend sticking with the six basic measurement variable names "Fx", "Fy", "Fz", "Mx", "My", and "Mz".

Value

A `data.table` of the class `fp.segm`. The following variables are included in the `data.table`:

- `subj`: subject number,
- `block`: block number,
- `trial`: trial number,
- `forceplate`: force-plate data of each trial as `data.table`. Use, for example, `fp.dt$forceplate[[1]]` to open the force-plate data of the first trial, first block, and first subject.

Author(s)

Raphael Hartmann & Anton Koger

References

Johannsen, L., Stephan, D. N., Straub, E., Döhring, F., Kiesel, A., Koch, I., & Müller, H. (2023). Assessing the influence of cognitive response conflict on balance control: An event-related approach using response-aligned force-plate time series data. *Psychological Research, 87*, 2297–2315.

Winter, D. A. (2009). *Biomechanics and Motor Control of Human Movement*.

Examples

```
# Using example data from GitHub which requires internet
 # takes longer than 5 seconds
if (curl::has_internet()) {
  url <- paste0("https://raw.githubusercontent.com/RaphaelHartmann/forceplate/",
                "main/data/subj013_block001.txt")

  # Safe download, handling potential errors
  tryCatch({
    filenames <- tempfile(pattern = c("subj013_block001_"),
                          tmpdir = tempdir(), fileext = ".txt")
    download.file(url, filenames)

    # segment raw text file from Bioware
   fp.dt <- segment_fp_data(filenames = filenames, n.trials = 80, baseline.trigger = 128,
                            baseline.intv = c(0, 215), start.trigger = 128,
                            stimulus.trigger.list = c(1, 2, 4, 8),
                            response.trigger.list = c(32, 64),
                            cond.trigger.list = list(stimulus = c(1, 2, 4, 8),
                                                     correctness = c(32, 64)),
                            control = list(prepend.ms = 0))

    # Clean up
    unlink(filenames)
  }, error = function(e) {
    message("Failed to download data: ", e$message)
  })
}
```

time_lock_stats *Calculate Statistics for Time Locked Bins*

Description

Processing segmented force-plate data by calculating descriptive statistics like mean, standard deviation, and range for defined time bins around time-locked events, such as stimulus onset or response onset etc.

Usage

```
time_lock_stats(
  fp.dt,
  vars,
  time.lock.trigger,
  bins,
  bin.width = NULL,
  n.bins = NULL,
  FUN = list(mean = mean, sd = sd, range = function(x) diff(range(x)))
)
```

Arguments

fp.dt	A data.table of the class "fp.segm" produced by segment_fp_data().
vars	A (vector of) character(s) giving the variable names in fp.dt$forceplate, for which the statistics (see FUN below) should be calculated for each bin (see arguments below). For example Fx, Fy, Mx, My etc.
time.lock.trigger	A (vector of) number(s) containing the trigger(s) marking the onset of the time locking (the event of interest like stimulus onset or response onset). The onset of this trigger will be treated as reference (point zero) for the bins to be defined in the next argument(s).
bins	Either a vector of length 2 or a list of vectors of length 2 providing the lower and upper boundaries of the bins (in milliseconds). If only one vector is used either one of the next two arguments can be used to make (equaly sized) bins. If a list is used the next two arguments are ignored.
bin.width	If bins is a vector of 2 then this argument can be used to divide the bin into smaller bins of the size bin.width in milliseconds.
n.bins	If bins is a vector of 2 then this argument can be used to divide the bin into n.bins number of bins of equal size. If bin.width is provided as well, n.bins will be ignored.
FUN	A list of functions. These functions should be statistics that take as input a vector and return a scalar. See usage for an example (mean, standard deviation, range).

Value

A `data.table` of the class `fp.tl`. The following variables are included in the `data.table`:

- `subj`: subject number,
- `block`: block number,
- `trial`: trial number,
- `forceplate`: force-plate data of each trial as `data.table`. Use, for example, `fp.dt$forceplate[[1]]` to open the force-plate data of the first trial, first block, and first subject (if `sort` in the `segment_fp_data` was set to `TRUE`.
- For each combination of variable `vars` and `bin` a new variable is created by the function(s) provided by `FUN`.

Author(s)

Raphael Hartmann & Anton Koger

References

Johannsen, L., Stephan, D. N., Straub, E., Döhring, F., Kiesel, A., Koch, I., & Müller, H. (2023). Assessing the influence of cognitive response conflict on balance control: An event-related approach using response-aligned force-plate time series data. *Psychological Research, 87*, 2297–2315.

Examples

```
# Using example data from github which requires internet
 # takes longer than 5 seconds
if (curl::has_internet()) {
  url <- paste0("https://raw.githubusercontent.com/RaphaelHartmann/forceplate/",
                "main/data/subj013_block001.txt")

  # Safe download, handling potential errors
  tryCatch({
    filenames <- tempfile(pattern = c("subj013_block001_"),
                          tmpdir = tempdir(), fileext = ".txt")
    download.file(url, filenames)
   fp.dt <- segment_fp_data(filenames = filenames, n.trials = 80, baseline.trigger = 128,
                       baseline.intv = c(0, 215), start.trigger = 128, start.prepend = 0,
                            stimulus.trigger.list = c(1, 2, 4, 8),
                            response.trigger.list = c(32, 64),
                            cond.trigger.list = list(stimulus = c(1, 2, 4, 8),
                                                     correctness = c(32, 64)))

   # Response-locking with 2 bins before and 2 bins after response onset. Each bin is 100 ms.
    tl.dt <- time_lock_stats(fp.dt = fp.dt, vars = c("Mx", "My"),
                         time.lock.trigger = c(1,2,4,8), bins = c(-150, 150), n.bins = 2,
                     FUN = list(mean = mean, sd = sd, range = function(x) diff(range(x))))

    # Clean up
    unlink(filenames)
  }, error = function(e) {
```

```
    message("Failed to download data: ", e$message)
  })
}
```

Index