Package 'eyelinkReader'

September 5, 2022

Title Import Gaze Data for EyeLink Eye Tracker

Version 1.0.0

Description Import gaze data from edf files generated by the SR Research <https://www.sr-research.com/> EyeLink eye tracker. Gaze data, both recorded events and samples, is imported per trial. The package allows to extract events of interest, such as saccades, blinks, etc. as well as recorded variables and custom events (areas of interest, triggers) into separate tables. The package requires EDF API library that can be obtained at <https://www.sr-support.com/>.

License GPL (>= 3)

URL https://github.com/alexander-pastukhov/eyelinkReader/, https://alexander-pastukhov.github.io/eyelinkReader/

BugReports https://github.com/alexander-pastukhov/eyelinkReader/issues

Depends R (>= 4.1.0), RcppProgress, rlang

Encoding UTF-8

NeedsCompilation yes

VignetteBuilder knitr

LazyData true

LinkingTo Rcpp, RcppProgress

Imports dplyr, fs, Rcpp, stringr, tidyr, methods, ggplot2

RoxygenNote 7.2.1

SystemRequirements GNU make

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Author Alexander Pastukhov [aut, cre] (<https://orcid.org/0000-0002-8738-8591>)

Maintainer Alexander Pastukhov <pastukhov.alexander@gmail.com>

Repository CRAN

Date/Publication 2022-09-05 08:10:05 UTC

R topics documented:

adjust_message_time . . . 2
compiled_library_status . . . 3
compute_cyclopean_samples . . . 4
convert_NAs . . . 5
extract_AOIs . . . 5
extract_blinks . . . 6
extract_display_coords . . . 7
extract_fixations . . . 8
extract_saccades . . . 9
extract_triggers . . . 10
extract_variables . . . 11
eyelinkReader . . . 12
eyelinkRecording-class . . . 12
gaze . . . 17
plot.eyelinkRecording . . . 18
print.eyelinkRecording . . . 19
read_edf . . . 20
read_preamble . . . 22

Index **23**

`adjust_message_time` *Adjusts message time based on embedded text offset*

Description

Uses text in the message to adjust its time. E.g., for a message `"-50 TARGET_ONSET"` that was sent at `105600` the actual onset occurred 50 milliseconds earlier (`-50`). The function adjusts the event timing and removes the timing offset information from the message. I.e., the example message becomes `"TARGET_ONSET"` and its time become `105550`.

Usage

```
adjust_message_time(object, prefix)

## S3 method for class 'data.frame'
adjust_message_time(object, prefix = "^[-+]?[:digit:]+[:space:]+")

## S3 method for class 'eyelinkRecording'
adjust_message_time(object, prefix = "^[-+]?[:digit:]+[:space:]+")
```

Arguments

object	An eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.
prefix	String with a regular expression that defines the offset. Defaults to "^[-+]?[:digit:]+[:space:]+" (a string starts with a positive or negative integer offset followed by a white space and the rest of the message).

Value

Object of the same time as input, i.e., either a eyelinkRecording object with *modified* events slot or a data.frame with offset-adjusted events.

Examples

```
data(gaze)

# by passing events table
adjusted_events <- adjust_message_time(gaze$events)

# by passing the recording
gaze <- adjust_message_time(gaze)
```

---

compiled_library_status

Status of compiled library

Description

Return status of compiled library

Usage

```
compiled_library_status()
```

Value

logical

Examples

```
compiled_library_status()
```

compute_cyclopean_samples

Computes cyclopean samples by averaging over binocular data

Description

Computes cyclopean samples by averaging over binocular recorded properties such as pxL/pxR, pyL/pyR, hxL/hxR, etc. Uses function specified via fun parameter to compute the average with na.rm = TRUE option. In case of a monocular recording or when the information from one eye missing, uses information from one eye only, ignoring the other column. In both binocular and monocular recording cases, simplifies column names so that pxL and/or pxR are replaced with a single column px, pyL/pyR with py, etc.

Usage

```
compute_cyclopean_samples(object, fun = mean)

## S3 method for class 'data.frame'
compute_cyclopean_samples(object, fun = mean)

## S3 method for class 'eyelinkRecording'
compute_cyclopean_samples(object, fun = mean)
```

Arguments

object	Either an eyelinkRecording object or data.frame with samples, i.e., samples slot of the eyelinkRecording object.
fun	Function used to average across eyes, defaults to mean.

Value

Object of the same time as input, i.e., either a eyelinkRecording object with *modified* samples slot or a data.frame with cyclopean samples.

Examples

```
data(gaze)

# by passing samples table
cyclopean_samples <- compute_cyclopean_samples(gaze$samples)

# storing cyclopean samples as a separate table in recording
gaze$cyclopean_samples <- compute_cyclopean_samples(gaze$samples)

# by passing the recording, cyclopean samples replace original ones
gaze <- compute_cyclopean_samples(gaze)
```

convert_NAs *Convert -32767 (missing info) to NA*

Description

Converts all -32767 (smallest INT16 value indicating missing info) to NA. You don't need to call this function directly, as it is automatically evoked within `read_edf` function.

Usage

```
convert_NAs(original_frame)
```

Arguments

`original_frame`	data.frame to be processed

Value

processed data.frame

Examples

```
data(gaze)
gaze$samples <- convert_NAs(gaze$samples)
```

extract_AOIs *Extracts rectangular areas of interest (AOI)*

Description

Extracts rectangular areas of interest (AOI), as defined by "`!V IAREA RECTANGLE`" command. Specifically, we expect it to be in format `!V IAREA RECTANGLE <index> <left> <top> <right> <bottom> <label>`, where `<label>` is a string label and all other variables are integer. Please note that due to a non-standard nature of this function **is not** called during the `read_edf` call and you need to call it separately.

Usage

```
extract_AOIs(object)

## S3 method for class 'data.frame'
extract_AOIs(object)

## S3 method for class 'eyelinkRecording'
extract_AOIs(object)
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.

Value

Object of the same time as input, i.e., either a eyelinkRecording object with an additional AOIs slot or a data.frame with AOIs' information. See eyelinkRecording for details.

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_AOIs(gaze)

# by passing events table
AOIs <- extract_AOIs(gaze$events)
```

---

extract_blinks *Extract blinks*

Description

Extracts blinks from the events table of the eyelinkRecording object.. Normally, you don't need to call this function yourself, as it is called during the read_edf with default settings (*e.g.*, import_blinks = TRUE).

Usage

```
extract_blinks(object)

## S3 method for class 'data.frame'
extract_blinks(object)

## S3 method for class 'eyelinkRecording'
extract_blinks(object)
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.

Value

Object of the same time as input, i.e., either a eyelinkRecording object with an additional blinks slot or a data.frame with blinks' information. See eyelinkRecording for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_blinks(gaze)

# by passing events table
blinks <- extract_blinks(gaze$events)
```

`extract_display_coords` *Extract display coordinates from an event message*

Description

Extracts display coordinates from a message that adheres to a `<message_prefix> <label>` format. Please note that this function called during the `read_edf` call with `silent = TRUE`. If `display_coords` are missing from the `eyelinkRecording`, run this method to see the warnings.

Usage

```
extract_display_coords(
  object,
  message_prefix = "DISPLAY_COORDS",
  silent = FALSE
)

## S3 method for class 'data.frame'
extract_display_coords(
  object,
  message_prefix = "DISPLAY_COORDS",
  silent = FALSE
)

## S3 method for class 'eyelinkRecording'
extract_display_coords(
  object,
  message_prefix = "DISPLAY_COORDS",
  silent = FALSE
)
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.
message_prefix	Beginning of the message string that identifies the DISPLAY_COORDS message. Defaults to "DISPLAY_COORDS".
silent	Whether to suppress a warning when DISPLAY_COORDS message is missing. Default to FALSE.

Value

A eyelinkRecording object with an additional display_coords slot (if that was object type), Either a four element numeric vector with display coordinates, or NULL if object was an events table of eyelinkRecording object. See eyelinkRecording for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_display_coords(gaze)

# by passing events table
display_coords <- extract_display_coords(gaze$events)
```

---

extract_fixations *Extract fixations*

Description

Extracts fixations from the events table of the eyelinkRecording object. Normally, you don't need to call this function yourself, as it is called during the read_edf with default settings (*e.g.*, import_fixations = TRUE).

Usage

```
extract_fixations(object)

## S3 method for class 'data.frame'
extract_fixations(object)

## S3 method for class 'eyelinkRecording'
extract_fixations(object)
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.

Value

Object of the same time as input, i.e., either a eyelinkRecording object with an additional fixations slot or a data.frame with fixations' information. See eyelinkRecording for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_fixations(gaze)

# by passing events table
fixations <- extract_fixations(gaze$events)
```

---

extract_saccades *Extract saccades from recorded events*

Description

Extract saccades from the events table of the eyelinkRecording object. Normally, you don't need to call this function yourself, as it is called during the read_edf with default settings (*e.g.*, import_saccades = TRUE).

Usage

```
extract_saccades(object)

## S3 method for class 'data.frame'
extract_saccades(object)

## S3 method for class 'eyelinkRecording'
extract_saccades(object)
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.

Value

Object of the same time as input, i.e., either a eyelinkRecording object with an additional saccades slot or a data.frame with saccades' information. See eyelinkRecording for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_saccades(gaze)

# by passing events table
saccades <- extract_saccades(gaze$events)
```

---

extract_triggers *Extract triggers, a custom message type*

Description

Extracts trigger events, messages that adhere to a <message_prefix> <label> format. Their purpose is to identify the time instance of specific interest. Please note that due to a non-standard nature of this function **is not** called during the read_edf call and you need to call it separately.

Usage

```
extract_triggers(object, message_prefix = "TRIGGER")

## S3 method for class 'data.frame'
extract_triggers(object, message_prefix = "TRIGGER")

## S3 method for class 'eyelinkRecording'
extract_triggers(object, message_prefix = "TRIGGER")
```

Arguments

object	Either an eyelinkRecording object or data.frame with events, i.e., events slot of the eyelinkRecording object.
message_prefix	Beginning of the message string that identifies trigger messages. Defaults to "TRIGGER".

Value

Object of the same time as input, i.e., either a eyelinkRecording object with an additional triggers slot or a data.frame with triggers' information. See eyelinkRecording for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_triggers(gaze)

# by passing events table
triggers <- extract_triggers(gaze$events)

# with an explicit message prefix
triggers <- extract_triggers(gaze$events, "TRIGGER")
```

---

extract_variables *Extract variables*

Description

Extracts variables from the `events` table of the `eyelinkRecording` object. Normally, you don't need to call this function yourself, as it is called during the `read_edf` with default settings (*e.g.*, `import_variables = TRUE`).

Usage

```
extract_variables(object)

## S3 method for class 'data.frame'
extract_variables(object)

## S3 method for class 'eyelinkRecording'
extract_variables(object)
```

Arguments

`object`	Either an `eyelinkRecording` object or data.frame with events, i.e., `events` slot of the `eyelinkRecording` object.

Value

Object of the same time as input, i.e., either a `eyelinkRecording` object with an additional `variables` slot or a data.frame with variables' information. See `eyelinkRecording` for details.

See Also

read_edf, eyelinkRecording

Examples

```
data(gaze)

# by passing the recording
gaze <- extract_variables(gaze)

# by passing events table
variables <- extract_variables(gaze$events)
```

`eyelinkReader` *eyelinkReader*

Description

Imports gaze data recorded by a SR Research EyeLink eye tracker from an EDF file. Includes options to import events and/or recorded samples and extract individual events such as saccades, fixations, blinks, and recorded variables.

`eyelinkRecording-class`
Class `eyelinkRecording`.

Description

S3 class containing information imported from an edf-file.

Details

See `methods(class = "eyelinkRecording")` for an overview of available methods.

Slots

`preamble` A preamble of the recording, see also `read_preamble`.

`events` Events table which is a collection of all `FEVENT` imported from the EDF file. See description below.

`samples` Samples table which is a collection of all `FSAMPLE` imported from the EDF file. See description below.

`headers` Headers of the individual trials, see description below.

`recordings` Individual recording start/end information, see description below.

`display_coords` Recorded screen coordinates (if recorded), see `extract_display_coords`.

`saccades` Saccades extracted from `events`, see description below and `extract_saccades`.

`fixations` Fixations extracted from `events`, see description below and `extract_fixations`.

`blinks` Blinks extracted from `events`, see description below and `extract_blinks`.

`variables` Recorded variables extracted from `events`, see description below and `extract_variables`.

`triggers` Events messages that adhere to a `TRIGGER <label>` format. This is a **non-standard message** that the package author uses to mark events like onsets or offsets, similar to how it is done in M/EEG. See description below and `extract_triggers`.

`AOIs` Areas of interest events. See description below and `extract_AOIs`.

Events

Events table which is a collection of all `FEVENT` imported from the EDF file. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. Additional non-standard fields are marked in bold.

- `trial` Trial index, starts at 1.
- `time` Time of event.
- `type` Event type.
- `read` Flags which items were included.
- `sttime` Start time of the event.
- `entime` End time of the event.
- `sttime_rel` Start time of the event, relative to the start time of the trial.
- `entime_rel` End time of the event, relative to the start time of the trial.
- `hstx`, `hsty` Head reference starting points.
- `gstx`, `gsty` Gaze starting points.
- `sta` Pupil size at start.
- `henx`, `heny` Headref ending points.
- `genx`, `geny` Gaze ending points.
- `ena` Pupil size at end.
- `havx`, `havy` Headref averages.
- `gavx`, `gavy` Gaze averages.
- `ava` Average pupil size.
- `avel` Accumulated average velocity.
- `pvel` Accumulated peak velocity.
- `svel` Start velocity.
- `evel` End velocity.
- `supd_x`, `supd_y` Start units-per-degree.
- `eupd_x`, `eupd_y` End units-per-degree.
- `eye` Either `'LEFT'` (0) or `'RIGHT'` (1).
- `status` Error, warning flags.
- `flags` Flags to indicate contents.
- `input` Extra (input word).
- `buttons` Button state and changes.
- `parsedby` 7 bits of flags, PARSEDBY codes.
- `message` Any message string.

Samples

Samples table which is a collection of all FSAMPLE imported from the EDF file. Please note that read_edf parameters determines which fields are imported. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. Suffixes L and R denote left and right eye. Non-standard additional fields are marked in bold.

- trial Trial index, starts at 1.
- eye 'LEFT' (0), 'RIGHT' (1), or 'BINOCULAR' (2).
- time Time of sample.
- time_rel Time relative to the start of the trial.
- pxL, pxR, pyL, pyR Pupil coordinates.
- hxL, hxR, hyL, hyR Headref coordinates.
- paL, paR Pupil size or area.
- gxL, gxR, gyL, gyR Screen gaze coordinates.
- rx, ry Screen pixels per degree.
- gxvelL, gxvelR, gyvelL, gyvelR Gaze velocity.
- hxvelL, hxvelR, hyvelL, hyvelR Headref velocity.
- rxvelL, rxvelR, ryvelL, ryvelR Raw velocity.
- fgxvelL, fgxvelR, fgyvelL, fgyvelR Fast gaze velocity.
- fhxvelL, fhxvelR, fhyvelL, fhyvelR Fast headref velocity.
- frxvelL, frxvelR, fryvelL, fryvelR Fast raw velocity.
- hdata_1 -hdata_8 Head-tracker data (not pre-scaled). Each column correspond to a single element of the INT16 FSAMPLE::hdata[8].
- flags Flags to indicate contents.
- input Extra (input word).
- buttons Button state & changes.
- htype Head-tracker data type (0=none).
- errors Process error flags.

Headers

Trial headers table which is a collection of all TRIAL structures imported from the EDF file. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. All fields of the RECORDINGS structure are prefixed with rec_. Non-standard additional fields are marked in bold.

- trial Trial index.
- duration Duration of the trial.
- starttime Start time of the trial.
- endtime End time of the trial.
- rec_time Start time or end time.

- rec_sample rate Sample rate in Hz: 250, 500, 1000 or 2000.
- rec_eflags Extra information about events.
- rec_sflags Extra information about samples.
- rec_state 'START' (2) or 'END' (1).
- rec_record_type 'SAMPLES' (1), 'EVENTS' (2), or 'SAMPLES and EVENTS' (3).
- rec_pupil_type 'AREA' (0) or 'DIAMETER' (1).
- rec_recording_mode 'PUPIL' (0) or 'CR' (1).
- rec_filter_type 1, 2, or 3.
- rec_pos_type Should be 'GAZE' (0), 'HREF' (1) or 'RAW', but currently this column is kept as numeric, since observed values do not match the declared constants.
- rec_eye 'LEFT' (1), 'RIGHT' (2) or 'LEFT and RIGHT' (3).

Recordings

Recordings table which is a collection of all RECORDING structures imported from the EDF file. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. Non-standard additional fields are marked in bold.

- trial Trial index.
- time Start time or end time.
- sample rate Sample rate in Hz: 250, 500, 1000 or 2000.
- eflags Extra information about events.
- sflags Extra information about samples.
- state 'START' (2) or 'END' (1).
- record_type 'SAMPLES' (1), 'EVENTS' (2), or 'SAMPLES and EVENTS' (3).
- pupil_type 'AREA' (0) or 'DIAMETER' (1).
- recording_mode 'PUPIL' (0) or 'CR' (1).
- filter_type 1, 2, or 3.
- pos_type Should be 'GAZE' (0), 'HREF' (1) or 'RAW', but currently this column is kept as numeric, since observed values do not match the declared constants.
- eye 'LEFT' (1), 'RIGHT' (2) or 'LEFT and RIGHT' (3).

Saccades and Fixations

Saccades and fixations extracted from the events, tables have the same structure. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. Non-standard additional fields are marked in bold.

- trial Trial index.
- sttime Start time.
- entime End time.
- sttime_rel Start time, relative to the start time of the trial.

- entime_rel End time, relative to the start time of the trial.
- duration Duration.
- hstx, hsty Head reference starting points.
- gstx, gsty Gaze starting points.
- sta Pupil size at start.
- henx, heny Headref ending points.
- genx, geny Gaze ending points.
- ena Pupil size at end.
- havx, havy Headref averages.
- gavx, gavy Gaze averages.
- ava Average pupil size.
- avel Accumulated average velocity.
- pvel Accumulated peak velocity.
- svel Start velocity.
- evel End velocity.
- supd_x, supd_y Start units-per-degree.
- eupd_x, eupd_y End units-per-degree.
- eye Either 'LEFT' (1) or 'RIGHT' (2).

Blinks

Blinks extracted from the events table. Column descriptions were copied directly from the *EDF access C API manual*. Please refer to that manual for further details. Non-standard additional fields are marked in bold.

- trial Trial index.
- sttime Start time.
- entime End time.
- sttime_rel Start time, relative to the start time of the trial.
- entime_rel End time, relative to the start time of the trial.
- duration Duration.
- eye Either 'LEFT' (1) or 'RIGHT' (2).

Variables

User recorded variables extracted from message events with a 'TRIAL_VAR' prefix. Original format can be either 'TRIAL_VAR <name> <value>' or 'TRIAL_VAR <name>=<value>'. The <name> cannot contain spaces or '=' sign. White spaces are trimmed for both <name> and <value>.

- trial Trial index.
- sttime Start time.
- sttime_rel Start time, relative to the start time of the trial.
- variable Variable name, the <name> part of the event message.
- value Variable value, the <value> part of the event message.

Trigger events

Events messages that adhere to a TRIGGER <label> format. This is a **non-standard message** that the package author uses to mark events like onsets or offsets, similar to how it is done in M/EEG.

- `trial` Trial index.
- `sttime` Start time.
- `sttime_rel` Start time, relative to the start time of the trial.
- `label` *label* part of the message, can contain white spaces.

AOIs

Rectangular areas of interest (AOI), as defined by "!V IAREA RECTANGLE" command. Specifically, they are expected to be in format `!V IAREA RECTANGLE <index> <left> <top> <right> <bottom> <label>`. where `<label>` is a string label and all other variables are integer.

- `trial` Trial index.
- `sttime` Start time.
- `sttime_rel` Start time, relative to the start time of the trial.
- `index` AOI index.
- `left`, `top`, `right`, `bottom` AOI coordinates.
- `label` AOI label.

See Also

`read_edf`, `extract_saccades`, `extract_fixations`, `extract_blinks`, `extract_triggers`, `extract_display_coords`, `extract_AOIs`

---

gaze *Imported example.edf, events and samples*

Description

An `eyelinkRecording` for *example.edf* via `read_edf(system.file("extdata", "example.edf", package = "eyelinkReader"), import_samples = TRUE)`. Contains all extracted events including triggers, areas of interested, and display coordinates. The original recording consist of ten trials with a participant fixating on a target that jumps to a new location after one second and stays on the screen for another second. Includes all relevant events.

Usage

```
gaze
```

Format

An object of class `eyelinkRecording` of length 12.

Details

See eyelinkRecording for details.

See Also

eyelinkRecording, read_edf

---

plot.eyelinkRecording *Plot fixations and saccades for a set of trials*

Description

This is only a basic plotting utility intended primarily for a quick visual check. Please refer to companion vignette on plotting for details about geoms and implementing your own custom plotting routine.

Usage

```
## S3 method for class 'eyelinkRecording'
plot(
  x,
  trial = 1,
  show_fixations = TRUE,
  fixation_size_property = "duration",
 size_legend = ifelse(fixation_size_property == "duration", "Fixation duration [ms]",
   NA),
  show_saccades = TRUE,
  saccade_color_property = "sttime_rel",
 color_legend = ifelse(saccade_color_property == "sttime_rel", "Saccade onset [ms]", NA),
  ...
)
```

Arguments

x	eyelinkRecording object
trial	Trials to be plotted, could be a scalar index, a vector of indexes, or NULL (all trials). Defaults to 1.
show_fixations	logical, whether to draw fixation as circles. Defaults to TRUE.
fixation_size_property	Which fixation property is used as circle aesthetics. Defaults to "duration".
size_legend	An optional legend title, defaults to "Fixation duration [ms]" if fixation_size_property is "duration" and to NA otherwise. In the latter case, the legend title is unmodified (i.e., determined by ggplot).
show_saccades	logical, whether to draw saccades as line segments. Defaults to TRUE.

saccade_color_property
: Which saccade property is used as color aesthetics. Defaults to "sttime_rel" (onset time relative to the trial start).

color_legend
: An optional legend title, defaults to "Saccade onset [ms]" if saccade_color_property is "sttime_rel" and to NA otherwise. In the latter case, the legend title is unmodified (i.e., determined by ggplot).

...
: Addition parameters (unused)

Value

ggplot object

Examples

```
data(gaze)

# fixations and saccades for the first trial
plot(gaze)

# fixations for the all trials
plot(gaze, trial = NULL, show_saccades = FALSE)

# saccades for the first two trials
plot(gaze, trial = 1:2, show_fixations = FALSE)

# color codes duration of a saccade
plot(gaze, saccade_color_property = "duration")
```

---

print.eyelinkRecording

Print info about eyelinkRecording

Description

Print info about eyelinkRecording

Usage

```
## S3 method for class 'eyelinkRecording'
print(x, ...)
```

Arguments

x
: eyelinkRecording object

...
: Addition parameters (unused)

Value

No return value, called for printing to console.

Examples

```
if (eyelinkReader::compiled_library_status()) {
 recording <- read_edf(system.file("extdata", "example.edf", package = "eyelinkReader"))
  print(recording)
}
```

`read_edf` *Read EDF file with gaze data recorded by SR Research EyeLink eye tracker*

Description

Reads EDF file with gaze data recorded by SR Research EyeLink eye tracker and returns an `eyelinkRecording` object that contains events, samples, and recordings, as well as specific events such as saccades, fixations, blinks, etc.

Usage

```
read_edf(
  file,
  consistency = "check consistency and report",
  import_events = TRUE,
  import_recordings = TRUE,
  import_samples = FALSE,
  sample_attributes = NULL,
  start_marker = "TRIALID",
  end_marker = "TRIAL_RESULT",
  import_saccades = TRUE,
  import_blinks = TRUE,
  import_fixations = TRUE,
  import_variables = TRUE,
  verbose = TRUE,
  fail_loudly = TRUE
)
```

Arguments

`file`	full name of the EDF file
`consistency`	consistency check control for the time stamps of the start and end events, etc. Could be `'no consistency check'`, `'check consistency and report'` (default), `'check consistency and fix'`.
`import_events`	logical, whether to import events, defaults to codeTRUE
`import_recordings`	logical, whether to import information about start/end of the recording, defaults to codeTRUE

`import_samples` logical, whether to import samples, defaults to `FALSE`. Please note that specifying `sample_attributes` automatically sets it to `TRUE`.

`sample_attributes`
: a character vector that lists sample attributes to be imported. By default, all attributes are imported (default). For the complete list of sample attributes please refer to `eyelinkRecording` or EDF API documentation.

`start_marker` event string that marks the beginning of the trial. Defaults to `"TRIALID"`.

`end_marker` event string that marks the end of the trial. Defaults to `"TRIAL_RESULT"`. Please note that an **empty** string `''` means that a trial lasts from one `start_marker` till the next one.

`import_saccades`
: logical, whether to extract saccade events into a separate table for convenience. Defaults to `TRUE`.

`import_blinks` logical, whether to extract blink events into a separate table for convenience. Defaults to `TRUE`.

`import_fixations`
: logical, whether to extract fixation events into a separate table for convenience. Defaults to `TRUE`.

`import_variables`
: logical, whether to extract stored variables into a separate table for convenience. Defaults to `TRUE`.

`verbose` logical, whether the number of trials and the progress are shown in the console. Defaults to `TRUE`.

`fail_loudly` logical, whether lack of compiled library means error (`TRUE`, default) or just warning (`FALSE`).

Value

an `eyelinkRecording` object that contains events, samples, and recordings, as well as specific events such as saccades, fixations, blinks, etc.

Examples

```
if (eyelinkReader::compiled_library_status()) {
  # Import only events and recordings information
 recording <- read_edf(system.file("extdata", "example.edf", package = "eyelinkReader"))

  # Import events and samples (only time and  screen gaze coordinates)
 recording <- read_edf(system.file("extdata", "example.edf", package = "eyelinkReader"),
                       sample_attributes = c('time', 'gx', 'gy'))

  # Import events and samples (all attributes)
 recording <- read_edf(system.file("extdata", "example.edf", package = "eyelinkReader"),
                       import_samples= TRUE)
}
```

`read_preamble` *Reads edf-file preamble*

Description

Read the preamble of the EDF file and parses it into an reading-friendly format

Usage

```
read_preamble(file, fail_loudly = TRUE)
```

Arguments

`file`	name of the EDF file
`fail_loudly`	logical, whether lack of compiled library means error (`TRUE`, default) or just warning (`FALSE`).

Value

a character vector but with added class `eyelinkPreamble` to simplify printing.

Examples

```
 if (eyelinkReader::compiled_library_status()) {
  read_preamble(system.file("extdata", "example.edf", package = "eyelinkReader"))
}
```

Index

∗ **datasets**
gaze, 17

adjust_message_time, 2

compiled_library_status, 3
compute_cyclopean_samples, 4
convert_NAs, 5

extract_AOIs, 5, *13*, *17*
extract_blinks, 6, *12*, *17*
extract_display_coords, 7, *12*, *17*
extract_fixations, 8, *12*, *17*
extract_saccades, 9, *12*, *17*
extract_triggers, 10, *13*, *17*
extract_variables, 11, *13*
eyelinkReader, 12
eyelinkRecording, *3, 4*, *6–11*, *17–21*
eyelinkRecording *(*eyelinkRecording-class*)*, 12
eyelinkRecording-class, 12

gaze, 17

mean, *4*

plot.eyelinkRecording, 18
print.eyelinkRecording, 19

read_edf, *5–11*, *14*, *17, 18*, 20
read_preamble, *12*, 22