Package 'imotionsApi'

June 12, 2025

Type Package

Title iMotions R library

Version 2.8.2-0000000000

Date 2025-06-12

Description Provides functions to access your iMotions studies.

Copyright iMotions

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.4.0)

Imports jsonlite (>= 1.8.8),
httr (>= 1.4.7),
data.table (>= 1.15.4),
arrow (>= 15.0.1),
tidyselect (>= 1.2.1),
tidyr (>= 1.3.1),
dplyr (>= 1.1.4),
methods,
purrr (>= 1.0.2),
stringr (>= 1.5.1),
rlang (>= 1.1.3)

Suggests testthat (>= 1.0.2),
mockr (>= 0.1),
lintr (>= 1.0.0),
knitr,
markdown,
rmarkdown,
mockery

URL https://imotions.com,
https://my.imotions.com/#studies

RoxygenNote 7.3.1

VignetteBuilder knitr

Contents

`imotionsApi-package` *iMotions R API package*

Description

A client for accessing data from the iMotions Lab or iMotions Online platform.

Details

Use tokens (found on a remote study's R Analysis page or by right-clicking on an analysis in the software) to start accessing your data.

Author(s)

Maintainer: iMotions <support@imotions.com>

Authors:

- Amandine Grappe

Other contributors:

- Paolo Masulli [contributor]

See Also

Useful links:

- https://imotions.com
- https://my.imotions.com/#studies

Examples

```
myToken <- "xxxxxxxx"
connection <- imotionsApi::imConnection(myToken)
```

`convertRecordingTsToIntervals`

Convert recording's timestamps (relative to data recording start) into stimulus/scene/AOI timestamps (relative to the interval first fragment start). Fragments are concatenated to give new array of timestamps in range [0, concatenated duration of stimulus/scene/AOI].

Description

Timestamps falling between an interval start/end will be kept, others will be discarded.

Usage

```
convertRecordingTsToIntervals(recordingTs, intervals, keepTs = FALSE)
```

Arguments

`recordingTs`	An array of recording's timestamps (relative to data recording start). Scalar, imSignals object as returned by `getSensorData` or data.table with a column Timestamp are also accepted.
`intervals`	An imInterval or imIntervalList object with start/end of a stimulus/scene/AOI as given by `getRespondentIntervals` or `getAOIRespondentData`.
`keepTs`	A boolean (or string) indicating whether timestamps falling outside an interval start/end should be kept unchanged, by default there are discarded. If keepTs is set to "NA", timestamps falling outside the interval will be replaced by NA. This will only work on array or scalar recordingTs.

Value

A new array/scalar/data.table with timestamps in range [0, concatenated duration of stimulus/scene/AOI].

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ])
signals <- imotionsApi::getSensorData(study, sensors[3, ])
```

```
intervals <- imotionsApi::getRespondentIntervals(study, respondents[1, ])

# get a new signal with timestamps in range [0, concatenated duration of stimulus/scene/AOI].
signals <- imotionsApi::convertRecordingTsToIntervals(signals, intervals[1, ])

## End(Not run)
```

---

createExport *Create an export file at a specific location and append metadata to it if provided. Note that a column with the study name will be appended to each export.*

Description

Create an export file at a specific location and append metadata to it if provided. Note that a column with the study name will be appended to each export.

Usage

```
createExport(
  params,
  study,
  data,
  outputDirectory,
  fileName,
  metadata = NULL,
  segment = NULL
)
```

Arguments

params	The list of parameters provided to the script.
study	An imStudy object as returned from imStudy.
data	A data.table containing the export metrics to save.
outputDirectory	The path where the file should be created.
fileName	The name of the file to create (should finish with .csv).
metadata	Optional - a data.table with metadata information. Column names will be converted to metadata headers and there must be a row corresponding to each data column.
segment	Optional - an imSegment object as returned from getSegment to upload the export. In case of a cloud study, this parameter needs to be provided.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
data <- data.frame("Respondent Name" = "Test", "Metric1" = seq(1:100), "Metric2" = rep(0, 100))
createExport(study, data, outputDirectory = "C:/Documents", fileName = "textExport.csv")
```

```
# Adding some metadata to the data
metadata <- data.table("Units" = c("", "ms", ""), "Description" = c("Desc1", "Desc2", "Desc"))
createExport(study, data, outputDirectory = "C:/Documents", fileName = "textExport.csv", metadata)

## End(Not run)
```

`getAOI` *Get a specific AOI from a study.*

Description

Available AOIs can be found with `getAOIs`. In case no AOI is found, return NULL.

Usage

```
getAOI(study, AOIId)
```

Arguments

`study`	An imStudy object as returned from `imStudy`.
`AOIId`	The id of the AOI you would like to retrieve.

Value

An imAOI object (data.table) containing the AOI of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
AOIs <- imotionsApi::getAOIs(study)
AOI <- imotionsApi::getAOI(study, AOIs$id[1])

## End(Not run)
```

`getAOIRespondentData` *Get the inOutGaze information, inOutMouseClick information and AOI's intervals for a specific AOI/respondent combination. Note that imAOI object, by definition, are linked to a specific stimulus.*

Description

The inOutGaze data.table has a IsGazeInAOI column that is TRUE when a gaze was recorded inside the AOI and FALSE if outside (timestamps correspond to the actual gazepoint Timestamp). To reduce the size of the file created, only timestamps where a change of value occur are given. If the AOI was never active, the table is empty.

Usage

```
getAOIRespondentData(study, AOI, respondent)
```

Arguments

study	An imStudy object as returned from imStudy()
AOI	An imAOI object as returned from getAOIs.
respondent	An imRespondent object as returned from getRespondents.

Details

The inOutMouseClick data.table has a IsMouseInAOI column that is TRUE when a click was recorded inside the AOI and FALSE if outside (timestamps correspond to the actual Timestamp of each click). If no click was recorded or if the AOI was never active, the table is empty.

Value

A list with inOutGaze/inOutMouseClick information for the specific AOI/respondent combination and an imIntervalList object (data.table) composed of the start, end, duration, id and name of this AOI.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
AOIs <- imotionsApi::getAOIs(study)
respondents <- imotionsApi::getRespondents(study, AOI = AOIs[1, ])
AOIData <- imotionsApi::getAOIRespondentData(study, AOIs[1, ], respondents[1, ])

# Retrieving list items
inOutData <- AOIData$inOutData
intervals <- AOIData$intervals

## End(Not run)
```

---

getAOIRespondentMetrics

Get the metrics for a specific AOI/respondent combination.

Description

Get the metrics for a specific AOI/respondent combination.

Usage

```
getAOIRespondentMetrics(study, AOI, respondent)
```

Arguments

study	An imStudy object as returned from imStudy()
AOI	An imAOI object as returned from getAOIs.
respondent	An imRespondent object as returned from getRespondents.

Value

A data.table of one row (imMetrics object) with metrics for the AOI /respondent combination of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
AOIs <- imotionsApi::getAOIs(study)
respondents <- imotionsApi::getRespondents(study, AOI = AOIs[1, ])
AOImetrics <- imotionsApi::getAOIRespondentMetrics(study, AOIs[1, ], respondents[1, ])

## End(Not run)
```

---

getAOIs *Get AOIs from a study.*

Description

Generic getAOIs function that takes as parameter a study object and optionally a respondent/stimulus object. In case no AOIs is defined for the combination, return NULL.

Usage

```
getAOIs(study, stimulus = NULL, respondent = NULL, generateInOutFiles = FALSE)
```

Arguments

study	An imStudy object as returned from imStudy.
stimulus	Optional - An imStimulus object as returned from getStimuli.
respondent	Optional - An imRespondent object as returned from getRespondents.
generateInOutFiles	A boolean indicating whether the corresponding InOut files should be generated and linked to each imAOI object.

Details

Important to note: to speed up the computation of gazes falling in/out AOIs at the respondent/stimulus level, an optional parameter generateInOutFiles can be enabled (both stimulus AND respondent arguments need to be provided). When enabled, a file will be generated for each AOI of this stimulus/respondent combination containing information regarding AOI activation/deactivation and gazes/clicks falling in. Filepaths to these newly generated files will be stored in their corresponding AOI object. If available, these filepaths will then directly be used by the getAOIRespondentData

function instead of re-generating the files. This is particularly useful in case multiple AOIs are defined for the same stimulus. For remote studies, InOutFiles are already generated so this parameter doesn't apply.

Value

An imAOIList object (data.table) with all AOIs of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
stimuli <- imotionsApi::getStimuli(study)
respondents <- imotionsApi::getRespondents(study)

## Get all AOIs in the study
AOIs <- imotionsApi::getAOIs(study)

## Get all AOIs defined for a specific stimulus
AOIs <- imotionsApi::getAOIs(study, stimulus = stimuli[1, ])

## Get all AOIs defined for a specific respondent
AOIs <- imotionsApi::getAOIs(study, respondent = respondents[1, ])

## Get all AOIs defined for a specific stimulus/respondent combination
AOIs <- imotionsApi::getAOIs(study, respondent = respondents[1, ], stimulus = stimuli[1, ])

## Get all AOIs defined for a specific stimulus/respondent combination and process their InOut data
AOIs <- imotionsApi::getAOIs(study, respondent = respondents[1, ], stimulus = stimuli[1, ],
                             generateInOutFiles = T)

print(AOIs$fileId) # a field "fileId" should have been added with the path to the InOut file

## End(Not run)
```

`getRespondent` *Get a specific respondent from a study.*

Description

Available respondents can be found with `getRespondents`.

Usage

```
getRespondent(study, respondentId)
```

Arguments

`study`	An imStudy object as returned from `imStudy`.
`respondentId`	The id of the respondent you would like to retrieve.

Value

An imRespondent object (data.table) containing the respondent of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
respondent <- imotionsApi::getRespondent(study, respondents$id[1])

## End(Not run)
```

`getRespondentIntervals` *Get the list of time intervals (imIntervalList) for a given respondent.*

Description

This imIntervalList is composed of stimuli/scenes/annotations intervals. For remote study, only stimuli intervals are supported as of now. Note that AOIs intervals can be retrieved using `getAOIRespondentData`.

Usage

```
getRespondentIntervals(
  study,
  respondent,
  type = c("Stimulus", "Scene", "Annotation")
)
```

Arguments

`study`	An imStudy object as returned from `imStudy`.
`respondent`	An imRespondent object as returned from `getRespondents`.
`type`	The type of intervals to retrieve (can be set to Stimulus, Scene and/or Annotation).

Value

An imIntervalList object (data.table) composed of the start, end, duration, parent stimulus, id and name of each stimulus/scene/annotation. Annotations comments are also included.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
intervals <- imotionsApi::getRespondentIntervals(study, respondents[1, ])
```

```
# Get only the stimuli intervals
intervals <- imotionsApi::getRespondentIntervals(study, respondents[1, ], type = "Stimulus")

## End(Not run)
```

`getRespondents` *Get respondents from a study.*

Description

Generic getRespondents function that takes as parameter a study object and optionally a stimulus/AOI/segment object. As AOIs are linked to a stimulus, it is not possible to provide both of them.

Usage

```
getRespondents(
  study,
  stimulus = NULL,
  AOI = NULL,
  segment = NULL,
  keepRespondentVariables = FALSE
)
```

Arguments

`study`	An imStudy object as returned from `imStudy`.
`stimulus`	Optional - An imStimulus object as returned from `getStimuli`.
`AOI`	Optional - An imAOI object as returned from `getAOIs`.
`segment`	Optional - An imSegment object as returned from `getSegments`.
`keepRespondentVariables`	Optional - A boolean indicating whether respondent variables should be kept (if they are available). If this has the default value of FALSE, only the "group" variable is exposed.

Value

An imRespondentList object (data.table) with all respondents of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
stimuli <- imotionsApi::getStimuli(study)
segments <- imotionsApi::getSegments(study)
AOIs <- imotionsApi::getAOIs(study)

## Get all respondents in the study
respondents <- imotionsApi::getRespondents(study)
```

```
## Get all respondents exposed to a specific stimulus
respondents <- imotionsApi::getRespondents(study, stimulus = stimuli[1, ])

## Get all respondents in a specific segment
respondents <- imotionsApi::getRespondents(study, segment = segments[1, ])

## Get all respondents for whom a specific AOI has been defined
respondents <- imotionsApi::getRespondents(study, AOI = AOIs[1, ])

## Get all respondents in a specific segment exposed to a specific stimulus
respondents <- imotionsApi::getRespondents(study, stimulus = stimuli[1, ], segment = segments[1, ])

## Get all respondents in a specific segment for whom a specific AOI has been defined
respondents <- imotionsApi::getRespondents(study, AOI = AOIs[1, ], segment = segments[1, ])

## Get all respondents in the study and access their available variables
respondents <- imotionsApi::getRespondents(study, keepRespondentVariables = TRUE)

## End(Not run)
```

`getSegment` *Get a specific segment from a study.*

Description

Available segments can be found with `getSegments`.

Usage

```
getSegment(study, segmentId)
```

Arguments

`study`	An imStudy object as returned from `imStudy`.
`segmentId`	The id of the segment you would like to retrieve.

Value

An imSegment object (data.table) containing the segment of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
segments <- imotionsApi::getSegments(study)
segment <- imotionsApi::getSegment(study, segments$id[1])

## End(Not run)
```

getSegments *Get all segments from a study.*

Description

Retrieves detailed information about segments in the study.

Usage

```
getSegments(study)
```

Arguments

study	An imStudy object as returned from imStudy.

Value

An imSegmentList object (data.table) containing all segments from the study.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
segments <- imotionsApi::getSegments(study)

## End(Not run)
```

getSensorData *Download data corresponding to a specific sensor (signals/metrics).*

Description

Available sensors in your study can be listed using the getSensors.

Usage

```
getSensorData(study, sensor, signalsName = NULL, intervals = NULL)
```

Arguments

study	An imStudy object as returned from imStudy.
sensor	An imSensor object as returned from getSensors.
signalsName	Optional - A vector of specific signals name you would like to return.
intervals	Optional - An imInterval or imIntervalList object with start/end of data to subset as given by getRespondentIntervals. In case of segment sensor, this is not possible.

Details

Signals always have a "Timestamp" column and are unique to a given respondent/segment and a given sensor source. Metrics are stored as a special sensor, also specific to a given respondent.

Value

An imData object (data.table) containing the signals (imSignals) or metrics (imMetrics) for the sensor of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ])
data <- imotionsApi::getSensorData(study, sensors[1, ])

## End(Not run)
```

---

getSensors *Get all sensors available for a given respondent/segment.*

Description

Retrieves detailed information about sensors available for a given respondent/segment.

Usage

```
getSensors(study, target, stimulus = NULL)
```

Arguments

study	An imStudy object as returned from imStudy.
target	The target respondent/segment (an imRespondent/imSegment object as returned from getRespondents or getSegments).
stimulus	Optional - an imStimulus object as returned from getStimuli to retrieve sensors specific to this stimulus. In case of a segment target, this parameter needs to be provided.

Value

An imSensorList object (data.table) with all sensors collected for the respondent of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
segments <- imotionsApi::getSegments(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ])

# Get sensors for a specific stimulus
stimuli <- imotionsApi::getStimuli(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ], stimuli[1, ])

# Get sensors for a specific segment/Stimulus
stimuli <- imotionsApi::getStimuli(study)
sensors <- imotionsApi::getSensors(study, segments[1, ], stimuli[1, ])

## End(Not run)
```

---

getSensorsMetadata *Get sensors specific metadata.*

Description

Available sensors in your study can be listed using the getSensors.

Usage

```
getSensorsMetadata(sensors)
```

Arguments

sensors	An imSensorList object as returned from getSensors.

Value

A data.table with sensors metadata (one row by sensor).

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ])
metadata <- imotionsApi::getSensorsMetadata(study, sensors)

## End(Not run)
```

getStimuli *Get all stimuli from a study.*

Description

Retrieves detailed information about stimuli in the study.

Usage

```
getStimuli(study, respondent = NULL, relevant = TRUE)
```

Arguments

study	An imStudy object as returned from imStudy.
respondent	Optional - An imRespondent object as returned from getRespondents.
relevant	A boolean indicating whether only relevant stimuli should be kept, by default non-relevant stimuli are discarded.

Value

An imStimulusList object (data.table) containing all stimuli from the study.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)

## Get all stimuli in the study
stimuli <- imotionsApi::getStimuli(study)

## Get all stimuli for a specific respondent
stimuli <- imotionsApi::getStimuli(study, respondents[1, ])

## End(Not run)
```

getStimulus *Get a specific stimulus from a study.*

Description

Available stimuli can be found with getStimuli.

Usage

```
getStimulus(study, stimulusId)
```

Arguments

study	An imStudy object as returned from imStudy.
stimulusId	The id of the stimulus you would like to retrieve.

Value

An imStimulus object (data.table) containing the stimulus of interest.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
stimuli <- imotionsApi::getStimuli(study)
stimulus <- imotionsApi::getStimulus(study, stimuli$id[1])

## End(Not run)
```

---

imConnection *Create a connection with the iMotions API.*

Description

Tokens can be obtained by right-clicking on an analysis in iMotions and clicking "Get token for R API connection".

Usage

```
imConnection(token, baseUrl = NULL, s3BaseUrl = NULL)
```

Arguments

token	The token to be used for authentication.
baseUrl	Optional - The server to connect to in case of remote connection.
s3BaseUrl	Optional - The server to use to write back data in case of remote connection.

Value

An imConnection object to be passed to other methods.

Examples

```
## Not run:
# Local connection
myToken <- "xxxxxxxx"
connection <- imotionsApi::imConnection(myToken)

# Remote connection
myToken <- "token"
connection <- imotionsApi::imConnection(myToken, "myBaseUrl", "myS3BaseUrl")

## End(Not run)
```

imStudy *Load an iMotions study by id.*

Description

Retrieves detailed information about a study including stimuli, respondents, segments, and more.

Usage

```
imStudy(connection, studyId)
```

Arguments

connection	An imConnection object as returned from imConnection.
studyId	The id of the study you would like to retrieve.

Details

Available studies can be found with listStudies.

Value

An imStudy object to be passed to other methods.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
studyId <- studies$id[1]
study <- imotionsApi::imStudy(connection, studyId)

## End(Not run)
```

listLoadedStudies *List studies that have been loaded in the current session.*

Description

More detailed information about a study can be retrieved using imStudy.

Usage

```
listLoadedStudies()
```

Value

A list of the studies that have been loaded in the current session.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
studyId <- studies$id[1]
study <- imotionsApi::imStudy(connection, studyId)
listLoadedStudies()

## End(Not run)
```

---

listStudies *List available studies.*

Description

More detailed information about a study can be retrieved using imStudy.

Usage

```
listStudies(connection)
```

Arguments

connection	An imConnection object as returned from imConnection.

Value

A data frame containing names and ids of studies.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)

## End(Not run)
```

---

truncateSignalsByIntervals *Truncate signals data based on given intervals.*

Description

Any interval combination can be asked. Timestamps falling between an interval start/end will be kept, others will be discarded. If dropIntervals is set to TRUE, Timestamps falling between an interval start/end will be discarded.

Usage

```
truncateSignalsByIntervals(signals, intervals, dropIntervals = FALSE)
```

Arguments

signals	An imSignals object as returned by getSensorData or a data.table including a "Timestamp" column that needs to be cut.
intervals	An imInterval or imIntervalList object with start/end of data to subset as given by getRespondentIntervals or getAOIRespondentData.
dropIntervals	A boolean indicating whether Timestamps falling between an interval start/end should be discarded, by default there are kept and other Timestamps are removed.

Value

A truncated imSignals object.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
sensors <- imotionsApi::getSensors(study, respondents[1, ])
signals <- imotionsApi::getSensorData(study, sensors[3, ])
intervals <- imotionsApi::getRespondentIntervals(study, respondents[1, ])

# get the 3 first intervals
dataSubset <- imotionsApi::truncateSignalsByIntervals(signals, intervals[1:3, ])

# remove the second interval
dataSubset <- imotionsApi::truncateSignalsByIntervals(signals, intervals[2, ], dropIntervals = TRUE)

## End(Not run)
```

---

unloadStudies *Remove studies that have been loaded in the current session.*

Description

This function can be used when new changes have been made to already loaded studies.

Usage

```
unloadStudies()
```

Value

"Studies successfully removed from the current session" if success.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
studyId <- studies$id[1]
study <- imotionsApi::imStudy(connection, studyId)
unloadStudies()

## End(Not run)
```

`uploadAOIMetadata` *Upload AOI metrics metadata for a specific study. The "Group" and "Group description" fields are mandatory as they are used to group the metadata.*

Description

Upload AOI metrics metadata for a specific study. The "Group" and "Group description" fields are mandatory as they are used to group the metadata.

Usage

```
uploadAOIMetadata(study, metadata)
```

Arguments

`study`	An imStudy object as returned from imStudy()
`metadata`	A data.table with metadata information. Column names will be converted to metadata headers.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
metadata <- data.table("Units" = c("ms", ""), "Show" = c("FALSE", "TRUE"), "Group" = c("1", "2"),
                       "Group description" = c("1", "2"))

uploadAOIMetadata(study, metadata)

## End(Not run)
```

uploadAOIMetrics *Upload metrics for a specific respondent and AOI in a study.*

Description

Upload metrics for a specific respondent and AOI in a study.

Usage

```
uploadAOIMetrics(study, AOI, target, metrics)
```

Arguments

study	An imStudy object as returned from imStudy()
AOI	An imAOI object as returned from getAOIs.
target	The target respondent/segment for the sensor (an imRespondent/imSegment object as returned from getRespondents or getSegments).
metrics	A data.table containing the metrics to upload.

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
AOIs <- imotionsApi::getAOIs(study)
respondents <- imotionsApi::getRespondents(study, AOI = AOIs[1, ])
metrics <- data.frame("metric1" = 2, "metric2" = 234, "metric3" = 1234)
uploadAOIMetrics(study, AOIs[1, ], respondents[1, ], metrics)

## End(Not run)
```

uploadEvents *Create events for a specific respondent in a study.*

Description

Events data.table must be composed of a EventName, Timestamp and Description column. Description will be rendered as tooltip in the software.

Usage

```
uploadEvents(
  params,
  study,
  events,
  target,
  eventsName,
  scriptName,
  metadata = NULL
)
```

Arguments

params	The list of parameters provided to the script - specific parameters value will be stored as metadata.
study	An imStudy object as returned from imStudy.
events	A data.table containing the events to upload (imData object are also accepted).
target	The target respondent for the sensor (an imRespondent object as returned from getRespondents).
eventsName	The name of the new events you would like to create.
scriptName	The name of the script used to produce these signals.
metadata	Optional - a data.table with metadata information. Column names will be converted to metadata headers and there must be a row corresponding to each data column.

Details

Params required field are "iMotionsVersion" and "flowName" (flow name will be used to link events to the original script)

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)

events <- data.table("Timestamp" = seq(1:5), "EventName" = rep("My event names", 5),
                     "Description" = rep("Description of event", 5))

params <- list("iMotionsVersion" = 8, "flowName" = "Test")
uploadEvents(params, study, events, respondents[1, ], eventsName = "New events",
             scriptName = "Example Script")


# Adding some metadata to the events
metadata <- data.table("Units" = c("ms", "", ""), "Show" = c("FALSE", "TRUE", "TRUE"))
uploadEvents(params, study, events, respondents[1, ], eventsName = "New events",
             scriptName = "Example Script", metadata)

## End(Not run)
```

---

uploadMetrics *Create metrics for a specific respondent in a study.*

Description

Metrics data.table must be composed of a StimulusId column, a Timestamp column, and at least one additional column with metrics. The Timestamp column should be filled with recording timestamps falling during the stimulus of interest (i.e. the timestamp of the start of the stimulus it corresponds to).

Usage

```
uploadMetrics(
  params,
  study,
  metrics,
  target,
  metricsName,
  scriptName,
  metadata = NULL
)
```

Arguments

params	The list of parameters provided to the script - specific parameters value will be stored as metadata.
study	An imStudy object as returned from imStudy.
metrics	A data.table containing the metrics to upload (imData object are also accepted).
target	The target respondent for the sensor (an imRespondent object as returned from getRespondents).
metricsName	The name of the new metrics you would like to create.
scriptName	The name of the script used to produce these signals.
metadata	Optional - a data.table with metadata information. Column names will be converted to metadata headers and there must be a row corresponding to each data column.

Details

Params required field are "iMotionsVersion" and "flowName" (flow name will be used to link metrics to the original script)

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)

metrics <- data.table("StimulusId" = c("1000", "1001"), "Timestamp" = c(10, 20),
                      "Metric1" = c(12, 25), "Metrics2" = c(NA_real_, 120))

params <- list("iMotionsVersion" = 8, "flowName" = "Test")
uploadMetrics(params, study, metrics, respondents[1, ], metricsName = "New metrics",
              scriptName = "Example Script")


# Adding some metadata to the data
metadata <- data.table("Units" = c("", "ms", "", "microSiemens"))
uploadMetrics(params, study, metrics, respondents[1, ], metricsName = "New metrics",
              scriptName = "Example Script", metadata)

## End(Not run)
```

`uploadSensorData` *Create a new sensor for a specific respondent/segment in a study.*

Description

Signals data.table (with a Timestamp column) can be uploaded. After processing, the sensor can then be viewed and exported locally through the iMotions Desktop.

Usage

```
uploadSensorData(
  params,
  study,
  data,
  target,
  sensorName,
  scriptName,
  metadata = NULL,
  stimulus = NULL,
  overwrite = TRUE
)
```

Arguments

`params`	The list of parameters provided to the script - specific parameters value will be stored as metadata.
`study`	An imStudy object as returned from `imStudy`.
`data`	A data.table containing the signals to upload (imData object are also accepted).
`target`	The target respondent/segment for the sensor (an imRespondent/imSegment object as returned from `getRespondents` or `getSegments`).
`sensorName`	The name of the new sensor you would like to create.
`scriptName`	The name of the script used to produce these signals.
`metadata`	Optional - a data.table with metadata information. Column names will be converted to metadata headers and there must be a row corresponding to each data column.
`stimulus`	Optional - an imStimulus object as returned from `getStimuli` to upload data specific to this stimulus. In case of a segment target, this parameter needs to be provided.
`overwrite`	Optional - a boolean indicating if the new sensor should overwrite sensors generated for the same flowName. By default, they are overwritten.

Details

Params required field are "iMotionsVersion" and "flowName" (flow name will be use as "instance" of the new sensor)

Examples

```
## Not run:
connection <- imotionsApi::imConnection("xxxxxxxx")
studies <- imotionsApi::listStudies(connection)
study <- imotionsApi::imStudy(connection, studies$id[1])
respondents <- imotionsApi::getRespondents(study)
segments <- imotionsApi::getSegments(study)
data <- data.frame("Timestamp" = seq(1:100), "Thresholded value" = rep(0, 100))
params <- list("iMotionsVersion" = 8, "flowName" = "Test")

uploadSensorData(params, study, data, respondents[1, ], sensorName = "New sensor",
                 scriptName = "Example Script")

# Uploading data to a specific stimulus
stimuli <- imotionsApi::getStimuli(study)
uploadSensorData(params, study, data, respondents[1, ], sensorName = "New sensor",
                 scriptName = "Example Script", stimulus = stimuli[1, ])

# Uploading data to a specific segment/stimulus
stimuli <- imotionsApi::getStimuli(study)
uploadSensorData(params, study, data, segments[1, ], sensorName = "New sensor",
                 scriptName = "Example Script", stimulus = stimuli[1, ])

# Adding some metadata to the data
metadata <- data.table("Units" = c("ms", ""), "Show" = c("FALSE", "TRUE"))
uploadSensorData(params, study, data, respondents[1, ], sensorName = "New sensor",
                 scriptName = "Example Script", metadata)

## End(Not run)
```

Index