Package 'mpathr'

May 9, 2026

Title Easily Handling Data from the 'm-Path' Platform

Version 1.0.4

Description Provides tools for importing and cleaning Experience Sampling Method (ESM) data collected via the 'm-Path' platform. The goal is to provide with a few utility functions to be able to read and perform some common operations in ESM data collected through the 'm-Path' platform (<https://m-path.io/landing/>). Functions include raw data handling, format standardization, and basic data checks, as well as to calculate the response rate in data from ESM studies.

License GPL (>= 3)

URL https://m-path.io, https://github.com/m-path-io/mpathr

BugReports https://github.com/m-path-io/mpathr/issues

Depends R (>= 4.1.0)

Imports cli, dplyr, ggplot2, jsonlite, lifecycle, lubridate, readr, rlang, tidyr

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Merijn Mestdagh [aut, cre] (ORCID: <https://orcid.org/0000-0001-5077-861X>), Lara Navarrete [aut], Koen Niemeijer [aut] (ORCID: <https://orcid.org/0000-0002-0816-534X>), m-Path Software [cph]

Maintainer Merijn Mestdagh <merijn.mestdagh@m-path.io>

Repository CRAN

Date/Publication 2026-02-05 11:40:02 UTC

Contents

`example_data` *Example m-path data*

Description

Contains the preprocessed example data for an m-path research study.

In the study, 20 participants completed 11 beeps over the course of 10 days. The study consisted of:

- An intake questionnaire, that participants answered at the study's start.
- A main questionnaire (10 times per day), where participants answered questions about their emotions and context at the time.
- An evening questionnaire (once, at the end of the day), about their emotions and activities throughout the day.

Each row corresponds to one beep sent during the study.

Usage

```
example_data
```

Format

A data frame with 1980 rows and 47 columns:

participant Participant identifier.

code Code the participants used to sign up for the study.

questionnaire The questionnaire that participants answered in that beep (it can be the main or the evening questionnaire).

scheduled Time stamp for when the notification was scheduled for, in unix time.

sent Time stamp for when the notification was sent, in unix time.

start Time stamp for when the notification was answered, in unix time. If the notification was never answered, this value is an NA.

stop Time stamp for when the notification was completed, in unix time. If the notification was never answered, this value is an NA.

phone_server_offset The difference between the phone time and the server time.

obs_n Observation number for each participant. Goes from 1 (first observation), to 110 (last observation of the study).

day_n Day number of the study, for the participant. Goes from 1 to 10.

obs_n_day Observation number within the day (for each participant). Goes from 1 to 11.

answered Logical, whether the beep was answered or not.

bpm_day Average heart rate per day. Note that unlike the rest of the variables, this corresponds to simulated data.

gender Participant's gender. 1 means 'Male', 2 means 'Female', 3 'Other'.

gender_string Participant's gender, as a string.

age Participant's age in years.

life_satisfaction Composite variable corresponding to participant's life satisfaction according to the Satisfaction With Life Scale (SWLS).

neuroticism Composite variable corresponding to participant's neuroticism according to the Big Five Inventory (BFI).

slider_happy Participants' self-reported happiness at the time of the beep. From 0 (not happy at all) to 100 (very happy).

slider_sad Participants' self-reported sadness at the time of the beep. From 0 (not sad at all) to 100 (very sad).

slider_angry Participants' self-reported anger at the time of the beep. From 0 (not angry at all) to 100 (very angry).

slider_relaxed Participants' self-reported relaxation at the time of the beep. From 0 (not relaxed at all) to 100 (very relaxed).

slider_anxious Participants' self-reported anxiety at the time of the beep. From 0 (not anxious at all) to 100 (very anxious).

slider_energetic Participants' self-reported energy at the time of the beep. From 0 (not energetic at all) to 100 (very energetic).

slider_tired Participants' self-reported tiredness at the time of the beep. From 0 (not tired at all) to 100 (very tired).

location_index Index corresponding to the participant's answer to the question "Where are you now?", from a list of multiple options.

location_string Text corresponding to the participant's selected location at the time of the beep.

company_index Index corresponding to the participant's answer to the question "With whom are you right now?", from a list of multiple options.

company_string Text corresponding to the participant's selected company at the time of the beep.

activity_index Index corresponding to the participant's answer to the question "What are you doing now?", from a list of multiple options.

activity_string Text corresponding to the participant's selected activity at the time of the beep.

step_count Step count between the previous answered beep and the current beep

evening_slider_happy Participants' happiness during the day, from 0 (not happy at all) to 100 (very happy).

evening_slider_sad Participants' sadness during the day, from 0 (not sad at all) to 100 (very sad).

evening_slider_angry Participants' anger during the day, from 0 (not angry at all) to 100 (very angry).

evening_slider_relaxed Participants' relaxation during the day, from 0 (not relaxed at all) to 100 (very relaxed).

evening_slider_anxious Participants' anxiety during the day, from 0 (not anxious at all) to 100 (very anxious).

evening_slider_energetic Participants' energy during the day, from 0 (not energetic at all) to 100 (very energetic).

evening_slider_tired Participants' tiredness during the day, from 0 (not tired at all) to 100 (very tired).

evening_stressful Participant's answer to whether something stressful had happened during the day. 1 means 'yes', 0 means 'no'.

evening_positive Participant's answer to whether something positive had happened during the day. 1 means 'yes', 0 means 'no'.

positive_description Explanation of the positive event (if participants responded 'yes' to the previous question).

stressful_description Explanation of the stressful event (if participants responded 'yes' to the previous question).

evening_activity_index Index corresponding to the participant's answer(s) to the question "What activities did you do today?", from a list of multiple options.

evening_activity_string Text corresponding to the participant's selected activities during the day.

delay_start_min Delay in minutes between the scheduled beep and the time the participants started the beep.

delay_end_min Time in minutes the participants took to fill in the beep (difference between the columns start and stop).

`extract_app_usage` *Extract App Usage from Paired Name/Value Columns*

Description

[Experimental]

Parses app names and usage values into structured usage data, with start and end timestamps and usage durations for both "Far" and "Near" windows.

The input can be formatted in two ways:

- If the data is in its raw form (e.g. imported from CSV via `read.csv()`), both `app_names` and `app_values` should be character vectors where each element is a comma-separated string.
- If the data was imported via `read_mpath()`, then `app_names` should be a list of character vectors, and `app_values` should be a list of integer vectors.

The function expects that each app is associated with exactly six values: `startTimeFar`, `endTimeFar`, `usageFar`, `startTimeNear`, `endTimeNear`, `usageNear`.

Usage

```
extract_app_usage(app_names, app_values)
```

Arguments

app_names	Either a character vector (comma-separated strings) or a list of character vectors, one per row.
app_values	Either a character vector (comma-separated strings) or a list of numeric vectors, one per row. Each block of 6 values corresponds to one app's usage record.

Value

A list of tibbles (one per input row). Each tibble contains one or more rows:

- app: App name
- startTimeFar, endTimeFar: POSIXct timestamps (UTC)
- usageFar: Integer usage during the far window
- startTimeNear, endTimeNear: POSIXct timestamps (UTC)
- usageNear: Integer usage during the near window

Time windows

Each measurement of app usage includes two time windows: a "near: window that captures recent app activity (typically ending around the time of the ESM beep), and a "far" window that covers the 24 hours prior to the near window. For both windows, Android automatically provides a start time, an end time, and the total usage in seconds during that period. These time ranges are determined by the operating system and may vary across apps and across measurements. Because the start and end times of these app usage windows rarely align exactly with the time between ESM beeps, interpreting the values requires caution as the window may include usage that occurred before the last beep To draw meaningful conclusions about app use between two beeps, it is important to consider which time windows and how much each window overlaps with that interval. Differences in the length and timing of these windows can affect your interpretation and should be accounted for in your analysis.

Examples

```
# Using character input (e.g., raw from CSV)
app_names <- c("foo", "foo,bar")
app_values <- c(
  "1000,2000,1,3000,4000,2",
  "4000,5000,3,6000,7000,4,8000,9000,5,10000,11000,6"
)
extract_app_usage(app_names, app_values)

# Using list-column input (e.g., from read_mpath())
app_names <- list("foo", c("foo", "bar"))
app_values <- list(
  c(1000,2000,1,3000,4000,2),
  c(4000,5000,3,6000,7000,4,8000,9000,5,10000,11000,6)
```

```
)
extract_app_usage(app_names, app_values)

# You can also use this function within a tidyverse pipeline:
library(dplyr)
tibble(app_name = app_names, app_value = app_values) |>
  mutate(usage = extract_app_usage(app_name, app_value))
```

`mpath_example` *Get path to m-Path example data*

Description

This function provides an easy way to access the m-Path example files.

Usage

```
mpath_example(file = NULL)
```

Arguments

`file`	the name of the file to be accessed. If `NULL`, the function will return a list of all the example files.

Value

a character string with the path to the m-Path example data

Examples

```
# Example 1: access 'example_basic.csv' data

mpath_example('example_basic.csv') # returns the full path to the file
'example_basic.csv'

# Example 2: list all the example files

mpath_example() # returns the example files as a vector
```

`plot_response_rate` *Plots response rate per day (and per participant)*

Description

This function returns a ggplot object with the response rate per day (x axis) and participant (color). Note that instead of using calendar dates, the function returns a plot grouped by the day inside the study for the participant.

Usage

```
plot_response_rate(data, valid_col, participant_col, time_col)
```

Arguments

`data`	data frame with data
`valid_col`	name of the column that stores whether the beep was answered or not
`participant_col`	name of the column that stores the participant id (or equivalent)
`time_col`	name of the column that stores the time of the beep

Value

a ggplot object with the response rate per day (x axis) and participant (color)

Examples

```
# load data
data(example_data)

# make plot with plot_response_rate
plot_response_rate(data = example_data,
time_col = sent,
participant_col = participant,
valid_col = answered)
# The resulting ggplot object can be formatted using ggplot2 functions (see ggplot2
# documentation).
```

read_mpath *Read m-Path data*

Description

[Stable]

This function reads an m-Path CSV file into a tibble, an extension of a `data.frame`.

Usage

```
read_mpath(file, meta_data, warn_changed_columns = TRUE)
```

Arguments

`file`	A string with the path to the m-Path file.
`meta_data`	A string with the path to the meta data file.
`warn_changed_columns`	Warn if the question text, type of question, or type of answer has changed during the study. Default is `TRUE` and may print up to 50 warnings.

Details

Note that this function has been tested with the meta data version v.1.1, so it is advised to use that version of the meta data. In the m-Path dashboard, change the version in 'Export data' > "export version".

Value

A tibble with the m-Path data.

See Also

`write_mpath()` for saving the data back to a CSV file.

Examples

```
# We can use the function mpath_examples to get the path to the example data
basic_path <- mpath_example(file ="example_basic.csv")
meta_path <- mpath_example("example_meta.csv")

data <- read_mpath(file = basic_path,
                   meta_data = meta_path)
```

response_rate *Calculate response rate*

Description

Calculate response rate

Usage

```
response_rate(
  data,
  valid_col,
  participant_col,
  time_col = NULL,
  period_start = NULL,
  period_end = NULL
)
```

Arguments

data	data frame with data
valid_col	name of the column that stores whether the beep was answered or not
participant_col	name of the column that stores the participant id (or equivalent)
time_col	optional: name of the column that stores the time of the beep, as a 'POSIXct' object.
period_start	string representing the starting date to calculate response rates (optional). Accepts dates in the following formats: yyyy-mm-dd oryyyy/mm/dd.
period_end	period end to calculate response rates (optional).

Value

a data frame with the response rate for each participant, and the number of beeps used to calculate the response rate

Examples

```
# Example 1: calculate response rates for the whole study
# Get example data
data(example_data)

# Calculate response rate for each participant

# We don't specify time_col, period_start or period_end.
# Response rates will be based on all the participant's data
response_rate <- response_rate(data = example_data,
                               valid_col = answered,
```

```
                                participant_col = participant)

# Example 2: calculate response rates for a specific time period
data(example_data)

# Calculate response rate for each participant between dates
response_rate <- response_rate(data = example_data,
                                valid_col = answered,
                                participant_col = participant,
                                time_col = sent,
                                period_start = '2024-05-15',
                                period_end = '2024-05-31')

# Get participants with a response rate below 0.5
response_rate[response_rate$response_rate < 0.5,]
```

`timestamps_to_datetime` *Convert m-Path timestamps to a date time format*

Description

[Stable]

m-Path timestamps are based on the participant's local time zone, and when converted to R datetime format, they are interpreted as being in Coordinated Universal Time (UTC), previously known Greenwich Mean Time (GMT). This function allows for the conversion of m-Path timestamps to datetime, and optionally allows for the specification of a UTC offset or a forced time zone.

Usage

```
timestamps_to_datetime(x, tz_offset = NULL, force_tz = NULL)
```

Arguments

`x`	A vector of timestamps to be transformed to datetime.
`tz_offset`	A numeric value to be added to the timestamps before transforming to datetime. This is typically derived from the `timeZoneOffset` column from m-Path data. This is only useful when you want to compare timestamps in an absolute manner or link it to external data sources.
`force_tz`	A string specifying the time zone to force the timestamps to. This is useful when the data is to be compared to other data sources that are in a different time zone. Note that this will not change the actual time of the timestamp, but only the time zone that is displayed. A list of time zones can be used in `OlsonNames()`.

Details

This function has three use cases:

1. The most common use case: You have only ESM data and want to work in each participant's local time zone. In this case, the `tz_offset` and `force_tz` should be left empty. This is likely the right use case for you.
2. You have ESM data and external data (e.g. sensing data or data from a multi-lab study) that you want to match based on their time stamp. The external data is likely in UTC while m-Path data is in the participant's local time zone. In this case, you should specify the `tz_offset` argument to convert the local time stamps to true UTC time. However, this will change the time stamp to UTC so you will **lose the ability to work in the local time zone**.
3. This is a more specialised version of use case 2, namely when you are certain that every participant lives in the same time zone and there not been any changes in daylight savings time. In this case, you can specify the `force_tz` argument to set the same time zone for all participants. This will not change the displayed time (11AM will stay 11AM) but will change the underlying time zone.

Value

A vector of `POSIXct` objects representing the timestamps in the UTC time zone. The time zone may differ if `force_tz` is specified.

Background

Timestamps in m-Path, like those in `timeStampScheduled` and `timeStampStart`, are a variation on UNIX timestamps, defined as the number of seconds since January 1, 1970, at 00:00:00. However, unlike standard UNIX timestamps (which use UTC), m-Path timestamps are based on the participant's local time zone. This is because we are generally interested in time from the participant's perspective and not in an absolute sense compared to other participants. Unfortunately, having multiple time zones in a single column is a not possible in R, which is why all time zones are (incorrectly) displayed as UTC.

When converted to `R` datetime format, they may display as UTC, which could lead to confusion. This typically isn't an issue when analyzing ESM data within the participant's local context, but it can affect comparisons with other data sources. For accurate cross-referencing with other data, consider specifying the UTC offset to correctly adjust for the participant's local time. Alternatively, you can force the timestamps to display in a specific time zone using the `force_tz` argument.

Examples

```
data <- read_mpath(
  mpath_example("example_basic.csv"),
  mpath_example("example_meta.csv")
)[1:10,]

# The most common use case for this function: Convert
# `timeStampStart` to datetime. Remember that these are in the
# local time zone, but R displays them as being in UTC.
timestamps_to_datetime(data$timeStampStart)
```

```
# Convert `timeStampStop` to datetime, but as being the correct
# value in UTC.
timestamps_to_datetime(
  x = data$timeStampStop,
  tz_offset = data$timeZoneOffset
)

# Let's convert `timeStampSent` to datetime, but this time we want to
# force the time zone to be in "America/New_York" as we know all
# participants were in this time zone and so we can link with other
# data that is also in New York's time zone.
timestamps_to_datetime(
  x = data$timeStampSent,
  force_tz = "America/New_York"
)
```

`write_mpath` *Write m-Path data to a CSV file*

Description

[Experimental]

Save a data frame or tibble to a CSV file in the same format as the downloaded data from the m-Path website. This function is useful when you have made modifications to the original data and would like to save it in the same format. Note that reading back the data using `read_mpath()` may not always work, as the data may no longer be in line with the meta data of the original data file.

Usage

```
write_mpath(x, file, .progress = TRUE)
```

Arguments

`x`	A data frame or tibble to write to disk.
`file`	File or connection to write to.
`.progress`	Logical indicating whether to show a progress bar. Default is `TRUE`.

Details

Even though saving a data frame to a CSV file may seem trivial, there are several issues that need to be addressed when saving m-Path data. The main issue is that m-Path data contains list columns that need to be "collapsed" to a single string before they can be saved to a CSV file. This function collapses most list columns to a single string using `paste()` with commas as a delimiter of the values. However, for columns that contain strings, this is not possible as the strings themselves may contains commas as well. To address this, the function converts all character columns to JSON strings using `jsonlite::toJSON()` before saving them to disk.

While `write_mpath()` aims to provide a similar CSV file as the m-Path dashboard, we cannot provide any guarantees that the data can be read back using `read_mpath()`, especially when the

data has been modified. If you want to save the data to use it at a later point in R (even when transferring it to another computer), we recommend using `saveRDS()` or `save()` instead.

Note that the resulting data file may not exactly be equal to the original, even if it was not modified after reading it with `read_mpath()`. The main reason is that CSV files from the m-Path dashboard do not contain all necessary file delimiters corresponding to the number of rows in the data. This function, however, does contain the correct number of file delimiters which makes the files slightly bigger compared to the original file.

Value

Returns `x` invisibly.

See Also

`read_mpath()` to read m-Path data into R.

Examples

```
data <- read_mpath(
  mpath_example("example_basic.csv"),
  mpath_example("example_meta.csv")
)

write_mpath(data, "data.csv")
```

Index