Package: silverback (via r-universe)

November 17, 2026

Type Package

Title Tidy 'Gorilla' Experiment Builder Exports

Version 0.1.0

Description Reads the data files downloaded from the 'Gorilla' Experiment Builder <https://gorilla.sc/> and turns them into tidy tibbles. Files from Task Builder 1 and 2 and from Questionnaire Builder 1 and 2 can be read together, whether they were downloaded as one file per node or as combined files, as CSV or Excel files, and from several versions of an experiment. Task data are reduced to one row per response with the same columns for both task builders. Questionnaire answers are returned with one row per participant, and participant information such as status, device, randomiser outcomes and stored values is collected in one table. Recordings of the eye tracking and mouse tracking components are read with each sample assigned to a zone of the screen. A summary and a set of checks point out incomplete or repeated participation and unusual response patterns before the analysis. The tables can be saved as CSV files for use in other software.

License MIT + file LICENSE

URL https://github.com/mcanzi/silverback, https://mcanzi.de/silverback/

BugReports https://github.com/mcanzi/silverback/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports dplyr (>= 1.1.0), purrr (>= 1.0.0), readr (>= 2.0.0), readxl (>= 1.4.0), stats, stringr (>= 1.5.0), tibble, tidyr (>= 1.3.0), utils

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

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/pak/sysreqs libicu-dev libx11-dev

Repository https://mcanzi.r-universe.dev

Date/Publication 2026-11-17 07:29:35 UTC

RemoteUrl https://github.com/mcanzi/silverback

RemoteRef HEAD

RemoteSha eaabbc2f39e3d58894bb066ab63f025c414beb73

Contents

blind_ids 2
gorilla_checks 3
gorilla_example 4
gorilla_participants 5
gorilla_trials 6
gorilla_wide 7
read_eyetracking 8
read_gorilla 9
read_mousetracking 11
summary.gorilla_data 12
write_gorilla 13

Index **15**

`blind_ids` *Replace participant identifiers*

Description

Replaces the public identifiers that can link participants to a recruitment platform, such as Prolific or SONA IDs, with codes such as `P001`. The same participant gets the same code in all tables. The private identifiers assigned by 'Gorilla' are kept, because they do not identify anyone outside 'Gorilla'. Answers that participants typed in, such as an ID entered in a questionnaire, are not changed.

Usage

```
blind_ids(
  x,
  columns = c("participant_public_id", "participant_external_session_id",
    "participant_completion_code")
)
```

Arguments

x	A gorilla_data object from read_gorilla().
columns	Columns to replace. By default the public ID, the external session ID and the completion code.

Value

x with the identifiers replaced. Participants without a value keep NA.

Examples

```
g <- read_gorilla(gorilla_example())
gorilla_participants(blind_ids(g))
```

---

gorilla_checks *Check Gorilla data for common problems*

Description

Lists participants and tasks that may need a closer look before the analysis. Each row describes one finding. The function removes nothing, so you decide what to exclude.

Usage

```
gorilla_checks(
  x,
  fast = 200,
  max_fast = 0.1,
  max_same = 0.9,
  min_responses = 10,
  balance = 1.5
)
```

Arguments

x	A gorilla_data object from read_gorilla().
fast	Reaction time in milliseconds below which a response counts as fast.
max_fast	Largest acceptable proportion of fast responses.
max_same	Largest acceptable proportion of identical responses.
min_responses	Fewest responses in a task for the same_response check to apply.
balance	Largest acceptable ratio between the largest and the smallest group of a randomiser node.

Details

The checks are

status Participants whose status is not `complete`, such as live or rejected participants included by hand. Skipped if the export has no status column.

versions Data from more than one version of the experiment.

duplicate The same public ID under several private IDs, which usually means that someone took part more than once.

missing_task Participants who did not do a task that most participants with the same outcomes of randomiser, branch, order, counterbalance and allocator nodes did.

unbalanced Randomiser nodes whose groups differ in size by more than the ratio `balance`.

device Participants who used a different type of device from most participants.

fast Participants with more than the proportion `max_fast` of responses in a task faster than `fast` milliseconds.

same_response Participants who gave the same response in more than the proportion `max_same` of at least `min_responses` responses in a task where other responses were possible. For components that recorded no response values, the object clicked or chosen counts as the response.

Value

A tibble with columns `check`, `participant_private_id`, `task_name` and `detail`, with no rows if nothing was found.

See Also

`summary.gorilla_data()` for an overview of each task.

Examples

```
g <- read_gorilla(gorilla_example(), quiet = TRUE)
gorilla_checks(g)
```

---

gorilla_example *Path to the example exports*

Description

The package includes a small invented experiment downloaded in the one-file-per-node format: a lexical decision task made with Task Builder 2, a rhyme judgement task made with Task Builder 1, a consent form made with Questionnaire Builder 1 and a background questionnaire made with Questionnaire Builder 2. One participant stopped during the lexical decision task, and one node was never reached, so its file is empty. The data were simulated and the identifiers are made up.

Usage

```
gorilla_example(file = NULL)
```

Arguments

file	Name of one file. If NULL, the path to the folder.

Value

A path.

Examples

```
gorilla_example()
list.files(gorilla_example())
```

---

gorilla_participants *Participants*

Description

One row per participant, with the identifiers, the status, the device, operating system and browser, and the times of the first and last recorded event. tasks counts the tasks and questionnaires the participant has data for. The outcome of each randomiser, branch and checkpoint node follows, and then the last value of each store field. Columns that are empty for all participants are dropped.

Usage

```
gorilla_participants(x, status = NULL)
```

Arguments

x	A gorilla_data object from read_gorilla().
status	Keep only participants with these values of participant_status, such as "complete". All by default.

Value

A tibble. start and end are in UTC; minutes is the time between them.

Examples

```
g <- read_gorilla(gorilla_example())
gorilla_participants(g)
```

gorilla_trials *Responses in task data*

Description

Keeps the rows of the task data that hold a participant's response and gives them the same columns whether the task was made with Task Builder 1 or 2. Rows with instructions, continue buttons, fixation crosses, media events and other information are dropped.

Usage

```
gorilla_trials(x, task = NULL, all_rows = FALSE, quiet = FALSE)
```

Arguments

x	A gorilla_data object from read_gorilla().
task	Names of the tasks to include, as in the task_name column. All by default.
all_rows	Keep all rows of the trials, not only the responses. A column response_type then says what each row is, such as "response", "info" or a zone type.
quiet	If TRUE, no message about components without response values is shown.

Details

In Task Builder 2 a response is a row with response_type "response". In Task Builder 1 it is a row from a response zone (zone_type starting with response_).

Some components can record a response without a value, such as a Click Response whose objects have no response set up in 'Gorilla'. The clicked object is then only in object. When every response of a component in a task is empty, a message names the task and component. response is left empty, so use dplyr::coalesce(response, object) if the object names are the answers.

Value

A tibble with one row per response and columns

participant_private_id, experiment_version, tree_node_key, repeat_key, task_name, task_version Who did which task, in which version and node of the experiment, with Gorilla's Repeat Key.

spreadsheet Name of the spreadsheet used.

trial Trial number.

display, screen Display and screen of the response. Task Builder 1 has no displays.

component, object Component and object that recorded the response in Task Builder 2; zone type and zone name in Task Builder 1.

response The response, as text.

rt Reaction time in milliseconds.

correct Whether the response was scored as correct.

time Time of the response, in UTC.

followed by the timing columns of the export, such as `reaction_onset` in Task Builder 1 or `onset_time`, `absolute_reaction_time` and `response_duration` in Task Builder 2, and by the spreadsheet (`spreadsheet_`) and store (`store_`) columns. Only columns with values for the selected tasks are kept. All columns of the export remain in `x$tasks`.

Examples

```
g <- read_gorilla(gorilla_example())
gorilla_trials(g)
gorilla_trials(g, task = "Lexical decision")
```

---

gorilla_wide *Questionnaire answers with one row per participant*

Description

Turns the questionnaire answers of `read_gorilla()` into a wide table with one row per participant and one column per item. Items with a quantised value, such as the position of the chosen option in a dropdown or a Likert scale, get a second column ending in `_quantised`.

Usage

```
gorilla_wide(x, task = NULL, quantised = TRUE)
```

Arguments

`x`	A `gorilla_data` object from `read_gorilla()`.
`task`	Names of the questionnaires to include, as in the `task_name` column. All by default.
`quantised`	Add the `_quantised` columns.

Details

Questionnaires from Questionnaire Builder 1 and 2, downloaded in long or short format, give the same columns. An item that appears in more than one questionnaire gets the questionnaire name as a prefix. Objects of Questionnaire Builder 2 that share a name within a questionnaire, such as several objects left with the default name `Rating Scale`, are different questions and get their object ID as a suffix, as in `rating_scale_object_29`. If a participant answered the same item more than once, for example after repeating a node or going back to a page, the later answers go into columns ending in `__2`, `__3` and so on. Cleaned item names never contain a double underscore, so these columns cannot be confused with items whose names end in a number.

Value

A tibble with `participant_private_id` and one column per item. Column names are converted to snake case.

Examples

```
g <- read_gorilla(gorilla_example())
gorilla_wide(g)
```

`read_eyetracking` *Read eye tracking data*

Description

Reads the files that the Eye Tracking (Webgazer) component of Task Builder 2 saves for each trial. 'Gorilla' does not put these recordings in the main export but in files of their own, named like `123456-3-9100037-task-ab12-50000037-collection-10-2.csv` (gaze) and `...-calibration-1-2.csv` (calibration).

Usage

```
read_eyetracking(path, x = NULL, zones = NULL, quiet = FALSE)
```

Arguments

`path`	Paths to folders, CSV files or ZIP archives. Folders and ZIP archives are searched recursively for files whose names contain `-collection-` or `-calibration-`. CSV files named directly are read whatever their name.
`x`	Optional `gorilla_data` object from `read_gorilla()` of the same experiment, used to add the task and screen names.
`zones`	Names of the zones to assign samples to. All by default.
`quiet`	If `TRUE`, no message about files that could not be used is shown.

Details

Positions are given in normalised coordinates: `0` to `1` across the area of the task, with the origin at the bottom left, as in the export. Raw coordinates in pixels are kept as `x_raw` and `y_raw`, but they depend on the participant's screen. Predictions outside the area of the task, below `0` or above `1`, are kept; webcam eye tracking produces them when participants look away or the prediction is poor.

Each gaze sample is assigned to the zone of the screen it falls in. When zones overlap, the smallest zone containing the sample wins. Use `zones` to test only some zones, for example the pictures of a visual world task and not the response buttons placed on them.

The recordings do not say which task they belong to or, for gaze data, the name of the screen. Pass the main export as `x` to add `task_name` and `screen`, so that the samples can be joined to `gorilla_trials()` by `participant_private_id`, `tree_node_key`, `display`, `trial` and `screen`.

Value

An object of class `gorilla_eyetracking`: a list with

gaze One row per gaze sample, with the participant, node, task, spreadsheet, trial and screen, the kind of sample (`type`, usually `"prediction"`), the time of the sample in UTC (`time`), the time since the start of the recording in milliseconds (`elapsed`), the predicted gaze position (`x`, `y`, `x_raw`, `y_raw`) and the zone it falls in (`zone_id`, `zone_name`).

zones One row per zone and recording, with the object ID (`zone_id`), the name, and the edges `left`, `right`, `bottom` and `top` in normalised coordinates.

calibration One row per calibration point and attempt, with the outcome of the attempt (`"succeeded"`, `"failed - retrying"` or `"failed - max failures"`), the position of the point, the centre and standard deviation of the gaze predictions while the participant looked at it, and `error`, the distance between point and centre, all in normalised coordinates. Attempts that ended before any point was measured have one row with missing point values.

files One row per file read, with its kind and number of rows.

See Also

`read_mousetracking()` for the Mouse Tracking component.

Examples

```
## Not run:
g <- read_gorilla("data")
eye <- read_eyetracking("data", x = g)
eye$gaze
eye$calibration

## End(Not run)
```

`read_gorilla` *Read Gorilla exports*

Description

Reads the CSV or Excel files downloaded from the Data tab of the 'Gorilla' Experiment Builder and sorts them into tasks, questionnaires and participants. Both kinds of download are read: one file per node of the experiment tree (`data_exp_12345-v6_task-ab12.csv`) and the combined files (`data_exp_12345-v6_tasks.csv`, `data_exp_12345-v6_questionnaires.csv`).

Usage

```
read_gorilla(path, quiet = FALSE)
```

Arguments

path	Paths to folders, CSV or Excel files, or ZIP archives. Folders and ZIP archives are searched recursively for CSV and Excel files whose names start with data_exp_, the names 'Gorilla' gives its downloads. Files named directly are read whatever their name.
quiet	If TRUE, no message about skipped files is shown.

Details

Excel files (.xlsx) are read with every cell as text, so they give the same tables as the CSV download. Some Excel downloads store numbers as numbers, though, and a code such as 007 saved that way has already lost its leading zeros in the file. Download CSV files if such codes matter.

Files from Task Builder 1 and 2 and from Questionnaire Builder 1 and 2 can be mixed, and so can files from several versions of the experiment. The END OF FILE row at the end of each file is removed, and files from nodes that no participant reached, which hold no data, are skipped. Files that were opened in a spreadsheet program and saved again are read too: with semicolons or tabs as separators, in the Windows-1252 encoding, or as Unicode text (UTF-16 with a byte order mark, as Excel saves it). A file that is not UTF-8 is read as Windows-1252 with a warning, because other encodings cannot always be told apart from it.

Some problems give a warning, because the tables may not be what you expect: the same events in more than one file, as after reading both kinds of download, a file as CSV and as Excel or a second copy of a download; lines with the wrong number of columns, as in a file that was cut off or edited by hand; and a column name that appears twice in a file, as when a Task Builder 1 spreadsheet has a column with the same name as a column of the export.

Column names are converted to snake case: Participant Private ID becomes participant_private_id, Spreadsheet: Speaker becomes spreadsheet_speaker and Store: Alter becomes store_alter. Spreadsheet columns from Task Builder 1, which have no prefix in the export, get the same spreadsheet_ prefix.

A column becomes numeric only if every value in it is a number without leading zeros, so codes such as 007 and response keys such as T and F stay text. Identifiers and responses always stay text.

Value

An object of class gorilla_data: a list with

tasks All rows of all task files, one row per event, with cleaned column names, plus builder (the task builder that made the file) and file (the file the row comes from). See gorilla_trials() for one row per response.

questionnaires All questionnaire answers in long format, one row per participant and item, with columns item, response, quantised and file. See gorilla_wide() for one row per participant.

participants One row per participant. See gorilla_participants().

files One row per file read, with the experiment, version, node, builder, task and number of rows.

See Also

gorilla_trials(), gorilla_wide(), gorilla_participants()

Examples

```
path <- gorilla_example()
g <- read_gorilla(path)
g
g$files
```

`read_mousetracking` *Read mouse tracking data*

Description

Reads the files that the Mouse Tracking component of Task Builder 2 saves for each trial. 'Gorilla' does not put these recordings in the main export but in files of their own, named like `123456-3-9100037-task-ab12-500000`.

Usage

```
read_mousetracking(path, x = NULL, zones = NULL, quiet = FALSE)
```

Arguments

`path`	Paths to folders, CSV files or ZIP archives. Folders and ZIP archives are searched recursively for files whose names contain `-mousetracking-`. CSV files named directly are read whatever their name.
`x`	Optional `gorilla_data` object from `read_gorilla()` of the same experiment, used to add the task and screen names.
`zones`	Names of the zones to assign samples to. All by default.
`quiet`	If `TRUE`, no message about files that could not be used is shown.

Details

Positions are given in normalised coordinates: `0` to `1` across the area of the task, with the origin at the bottom left, as in the export. Raw coordinates in pixels are kept as `x_raw` and `y_raw`; in these, as recorded by the browser, `y` counts from the top of the window.

Each sample is assigned to the zone of the screen it falls in. When zones overlap, the smallest zone containing the sample wins. Use `zones` to test only some zones.

The recordings do not say which task they belong to. Pass the main export as `x` to add `task_name`, so that the samples can be joined to `gorilla_trials()` by `participant_private_id`, `tree_node_key`, `display`, `trial` and `screen`.

Value

An object of class `gorilla_mousetracking`: a list with

mouse One row per sample, with the participant, node, task, spreadsheet, trial and screen, the kind of sample (`type`, usually `"mouse"`), a timer of the browser in milliseconds (`timestamp`, not a date), the time since the start of the recording in milliseconds (`elapsed`), the position of the pointer (`x`, `y`, `x_raw`, `y_raw`) and the zone it is in (`zone_id`, `zone_name`).

zones One row per zone and recording, with the object ID (`zone_id`), the name, and the edges `left`, `right`, `bottom` and `top` in normalised coordinates.

files One row per file read, with its kind and number of rows.

See Also

`read_eyetracking()` for the Eye Tracking component.

Examples

```
## Not run:
g <- read_gorilla("data")
mouse <- read_mousetracking("data", x = g)
mouse$mouse

## End(Not run)
```

`summary.gorilla_data` *Summarise Gorilla data*

Description

Describes each task and questionnaire in one row, as a first check of the data after reading them.

Usage

```
## S3 method for class 'gorilla_data'
summary(object, fast = 200, slow = 10000, ...)
```

Arguments

`object`	A `gorilla_data` object from `read_gorilla()`.
`fast`, `slow`	Reaction times in milliseconds below `fast` or above `slow` are counted as fast or slow responses.
`...`	Not used.

Value

An object of class `gorilla_summary`: a list with two tibbles.

tasks One row per task: builder, number of participants, number of responses per participant (minimum, median and maximum), the proportion of responses marked as correct, the median reaction time, the proportions of fast and slow responses and the median time participants spent on the task, in minutes. The proportion correct only means something for tasks with scoring set up in 'Gorilla': components without scoring mark every response as correct, and rating scales count a response as correct only if it matches the answer column of the spreadsheet.

questionnaires One row per questionnaire: number of participants, number of items and the proportion of items left unanswered.

See Also

gorilla_checks() to list participants who may need a closer look.

Examples

```
g <- read_gorilla(gorilla_example(), quiet = TRUE)
summary(g)
summary(g)$tasks
```

---

write_gorilla *Write tidy data to CSV files*

Description

Writes the tidy tables to a folder of CSV files, for analysis in other software or for sharing. The folder gets up to four files: participants.csv from gorilla_participants(), trials.csv from gorilla_trials(), and questionnaires.csv from gorilla_wide() with one row per participant. questionnaires_long.csv has the questionnaire answers with one row per answer, as in x$questionnaires. Files for data the export does not have are not written.

Usage

```
write_gorilla(x, path, by_task = FALSE, overwrite = FALSE)
```

Arguments

x	A gorilla_data object from read_gorilla().
path	Folder to write to. It is created if it does not exist.
by_task	Write the trials of each task to its own file, named after the task, instead of one trials.csv.
overwrite	Replace files that already exist in path.

Details

The files are UTF-8 with a comma as separator. Missing values are left empty, and times are written in ISO 8601 format in UTC. The exports contain the identifiers Gorilla collected, so use blind_ids() first if the files will be shared.

Value

The paths of the written files, invisibly.

Examples

```
g <- read_gorilla(gorilla_example())
out <- file.path(tempdir(), "gorilla_csv")
files <- write_gorilla(g, out)
basename(files)
readr::read_csv(files[1], show_col_types = FALSE)
unlink(out, recursive = TRUE)
```

Index

blind_ids, 2
blind_ids(), *13*

gorilla_checks, 3
gorilla_checks(), *13*
gorilla_example, 4
gorilla_participants, 5
gorilla_participants(), *10*, *13*
gorilla_trials, 6
gorilla_trials(), *8*, *10, 11*, *13*
gorilla_wide, 7
gorilla_wide(), *10*, *13*

read_eyetracking, 8
read_eyetracking(), *12*
read_gorilla, 9
read_gorilla(), *3*, *5–8*, *11–13*
read_mousetracking, 11
read_mousetracking(), *9*

summary.gorilla_data, 12
summary.gorilla_data(), *4*

write_gorilla, 13