Package 'gmoTree'

May 8, 2026

Title Get and Modify 'oTree' Data

Version 1.4.1

Date 2025-02-04

Description Efficiently manage and process data from 'oTree' experiments. Import 'oTree' data and clean them by using functions that handle messy data, dropouts, and other problematic cases. Create IDs, calculate the time, transfer variables between app data frames, and delete sensitive information. Review your experimental data prior to running the experiment and automatically generate a detailed summary of the variables used in your 'oTree' code. Information on 'oTree' is found in Chen, D. L., Schonger, M., & Wickens, C. (2016) <doi:10.1016/j.jbef.2015.12.001>.

License GPL (>= 3)

URL https://zauchnerp.github.io/gmoTree/, https://github.com/ZauchnerP/gmoTree/, https://github.com/ZauchnerP/gmoTree

BugReports https://github.com/ZauchnerP/gmoTree/issues

Depends R (>= 4.4.0)

Imports data.table (>= 1.15.4), dplyr (>= 1.1.4), knitr (>= 1.47), openxlsx (>= 4.2.5.2), pander (>= 0.6.5), plyr (>= 1.8.9), rlang (>= 1.1.4), rlist (>= 0.4.6.2), rmarkdown (>= 2.27), stringr (>= 1.5.1), lifecycle (>= 1.0.4)

Suggests testthat (>= 3.2.1), withr (>= 3.0.0)

VignetteBuilder knitr

BuildVignettes true

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.3.2

Author Patricia F. Zauchner [aut, trl, cre, cph] (ORCID:
<https://orcid.org/0000-0002-5938-1683>)

Maintainer Patricia F. Zauchner <patricia.zauchner@gmx.at>

Repository CRAN

Date/Publication 2025-02-04 22:40:07 UTC

Contents

`apptime` *Calculate the time that was spent on an app*

Description

Calculate the time spent on one app or several apps.

Usage

```
apptime(
  oTree,
  apps = NULL,
  pcode = NULL,
  plabel = NULL,
  group_id = NULL,
  seconds = FALSE,
  rounded = TRUE,
```

```
  digits = 2,
  sinfo = "session_code",
  combine = FALSE
)
```

Arguments

oTree	A list of data frames created with `import_otree()`.
apps	Character string or character vector. Name(s) of the app(s) for which the time should be calculated.
pcode	Character string. The value of the `participant.code` variable if the time should only be calculated for one specified participant.
plabel	Character string. The value of the `participant.label` variable if the time should only be calculated for one specified participant.
group_id	Integer. The value of the `group_id` variable if the time should only be calculated for one specified group. The `group_id` variable can be created with `make_ids()`.
seconds	Logical. `TRUE` if the output should be in seconds instead of minutes.
rounded	Logical. `TRUE` if the output should be rounded.
digits	Integer. The number of digits to which the output should be rounded. This parameter has no effect unless `rounded = TRUE`.
sinfo	Character string. `"session_id"` to use session ID for additional information in the data frame of single durations, `"session_code"` to use session codes, or `NULL` if no session column should be shown.
combine	Logical. `TRUE` if all variables relating to epoch time should be merged, and all variables relating to participant code should be merged when data from multiple versions of oTree are used.

Value

This function returns a list for each app containing information on the mean, the minimum, and maximum time the participants spent on the app, a data frame with information on the time each participant spent on the app, and eventually, vectors of relevant background information on these numbers.

If the experiment's duration is only calculated for one participant, the output returns an `NA` (per app) if the person did not make it to the app(s).

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Show how much time all participants spent on app "survey"
apptime(oTree, apps = "survey")

# Show how much time the participant "a7dppel1" spent on
# the app "survey"
```

```
apptime(oTree, pcode = "a7dppel1", apps = "survey")

# Show how much time the participants in group 4 spent on
# the app "survey"
oTree <- make_ids(oTree,
    gmake = TRUE,
    from_var = "dictator.1.group.id_in_subsession"
)
apptime(oTree, group_id = 4, apps = "survey")

# Show how much time all participants spent on all apps
apptime(oTree)
```

`assignv` *Assign a variable from all_apps_wide*

Description

Assign a variable from `$all_apps_wide` to the other app data frames.

Usage

```
assignv(oTree, variable, newvar)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`variable`	Character string. The variable in the `$all_apps_wide` data frame that should be assigned to all other apps.
`newvar`	Character string. The name of the newly created variable.

Value

This function returns a duplicate of the original list of data frames but with an additional column in all data frames. The additional column contains data from the specified variable found in `$all_apps_wide`.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Assign variable "survey.1.player.gender" and name it "gender"
oTree <- assignv(oTree = oTree,
                 variable = "survey.1.player.gender",
                 newvar = "gender")

# Show the new variable in some of the other app data frames
oTree$dictator$gender
```

```
oTree$chatapp$gender

# The variable is now duplicated in app "survey" because it is obtained from
# there (This can be avoided by naming the new variable the same as the old
# variable)
oTree$survey$gender
oTree$survey$player.gender

# In app "all_apps_wide," the variable is also there twice (This can be
# avoided by naming the new variable the same as the old variable)
oTree$all_apps_wide$gender
oTree$all_apps_wide$survey.1.player.gender
```

`assignv_to_aaw` *Assign a variable to all_apps_wide*

Description

Assign a variable from one of the app data frames to `$all_apps_wide`.

Usage

```
assignv_to_aaw(oTree, app, variable, newvar, resafter = NULL)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`app`	Character string. The data frame from which the variable is taken.
`variable`	Character string. The name of the variable that should be assigned to `$all_apps_wide`.
`newvar`	Character string. The name of the newly created variable in the `$all_apps_wide` data frame.
`resafter`	Character string. The name of the variable that precedes the new variable. If `NULL`, the new variable will be placed at the end of the data frame.

Value

This function returns a duplicate of the original oTree list of data frames but with an additional column in the `$all_apps_wide` data frame that contains the variable in question.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Check out the old variable
oTree$survey$player.age

# Create a new variable
```

```
oTree$survey$younger30 <- ifelse(oTree$survey$player.age < 30, 1, 0)

# Assign the variable younger30 to all_apps_wide
oTree2 <- assignv_to_aaw(
  oTree = oTree,
  app = "survey",
  variable = "younger30",
  newvar = "younger30")

# Show the new variable in the all_apps_wide data frame
oTree2$all_apps_wide$younger30

# Check the position of the new variable
match("younger30",names(oTree2$all_apps_wide))

# Place the new variable immediately after the "survey.1.player.age" variable
oTree2 <- assignv_to_aaw(oTree,
                         app = "survey",
                         variable = "younger30",
                         newvar = "younger30",
                         resafter = "survey.1.player.age")

# Show the new variable in the all_apps_wide data frame
oTree2$all_apps_wide$younger30

# Show the position of the new variable
match("younger30", names(oTree2$all_apps_wide))
```

---

codebook *Create a codebook for the oTree code*

Description

Create a codebook of your oTree code by automatically scanning your project folder and retrieving the information of the apps' `Constants`, `Subsession`, `Group` and `Player` variables.

Usage

```
codebook(
  path = ".",
  fsource = "init",
  output = "both",
  output_dir = NULL,
  output_file = "codebook",
  output_format = "pdf_document_simple",
  output_open = TRUE,
  app_doc = TRUE,
  app = NULL,
  app_rm = NULL,
```

```
  doc_info = TRUE,
  sort = NULL,
  settings_replace = "global",
  user_settings = NULL,
  include_cons = TRUE,
  include_subs = FALSE,
  preamb = FALSE,
  encoding = "UTF-8",
  title = "Codebook",
  subtitle = "created with gmoTree",
  params = NULL,
  date = "today",
  splitvarname = FALSE,
  sep_list = "newline",
  initial = TRUE
)
```

Arguments

path	Character string. Path of the oTree experiment.
fsource	Character string. "init" if information should be taken from the init.py files (newer oTree code with 5.x format). "models" (or "model") if the information should be taken from the models.py files (older oTree code with 3.x format).
output	Character string. "list" if the output should contain a list of variables and their information. "file" if the output should be a file such as a Word or PDF file. "both" if the output should contain a file and a list.
output_dir	Character string. The absolute path where the function's output will be saved. Only absolute paths are allowed for this parameter. Relative paths can be specified in the output_file parameter.
output_file	Character string. The name of the output file generated by the function. The file name can be provided with or without an extension. Relative paths are also allowed in the file name.
output_format	Character string. Specifies the format of the file output. This value is passed to the output_format argument of rmarkdown::render. Allowed options are: "html_document", "word_document", "odt_document", "rtf_document", "md_document", "latex_document", "pdf_document", "pdf_document_simple", or their short forms "html", "word", "odt", "rtf", "md", "latex", "pdf", "pdf_simple". Important: The "pdf_document" format uses xelatex for PDF generation. If your document does not require advanced LaTeX features, it is recommended to use "pdf_document_simple".
output_open	Logical. TRUE if file output should be opened after creation.
app_doc	Logical. TRUE if app documentation should be included in the output file.
app	Character string or character vector. Name of the included app(s). Default is to use all apps. Cannot be used simultaneously with app_rm.
app_rm	Character string or character vector. Name of the excluded app(s). Default is to exclude no apps. Cannot be used simultaneously with app.

doc_info — Logical. TRUE if a message with information on all variables without documentation should also be returned. FALSE if this message should be suppressed.

sort — Character vector. Vector that specifies the order of the apps in the codebook.

settings_replace — Character string or NULL. Specifies how to handle references to settings variables. Use "global" to replace references with the global settings variables defined in settings.py. Use "user" to replace references with the variables provided in the user_settings argument. Use NULL to leave references to settings variables unchanged. Caution: This function does not use variables defined in SESSION_CONFIGS. If you vary settings variables in SESSION_CONFIGS, set settings_replace to "user" and manually replace them using the user_settings argument.

user_settings — List. List of variables in the settings.py file that are used to replace setting variable references. This is only used if settings_replace = "user" and should be used when setting variables are defined within the SESSION_CONFIGS.

include_cons — Logical. TRUE if there should be a section for the Constants variables in the codebook.

include_subs — Logical. TRUE if there should be a section for the Subsession variables in the codebook.

preamb — Deprecated. preamb = TRUE is no longer supported. Please remove preambles from your old codebooks.

encoding — Character string. Encoding of the created Markdown file. As in knitr::knit, this argument is always assumed to be UTF-8 and ignored.

title — Character string. Title of output file.

subtitle — Character string. Subtitle of output file.

params — List. List of variable name and value pairs to be passed to the RmD file. Only relevant if argument output "file" or "both" if chosen.

date — Character string or NULL. Date that is passed to the Rmd file. Either "today", NULL, or a user defined date. Only relevant if argument output "file" or "both" if chosen.

splitvarname — Logical. TRUE if long variable names should be split across multiple lines in the output file tables. If FALSE, table columns should adjust to fit the longest variable names.

sep_list — Character string. Determines how sub-lists are displayed in the file output. Use "newline" to separate sub-lists with newline characters (\\n), or "vector" to display them as strings in c(...) format.

initial — Logical. TRUE if initial values should be included in the output file. FALSE if they should not be included.

Details

This code works only when dictionaries are not used (for example, in the session configurations in settings.py).

Caution 1: Multiline comments are ignored, meaning that all variables commented out in this manner will nevertheless be included in the codebook. In contrast, variables commented out with line comments will not appear in the codebook.

Caution 2: If there are commas in the value strings, they might be used to split the text. Please manually insert a backslash symbol in front of the commas to avoid that (i.e., escape them). E.g. `"Yes, I will" -> "Yes\, I will"`.

Caution 3: This code cannot interpret variables that were imported from other files (for example CSV files) and that have special formatting included (e.g., special string formatting in Python such as `float(1.4)` to represent a float number).

Caution 4: This code was developed and tested with basic oTree codes and has not been verified for compatibility with oTree versions later than 5.4.0. If you experience issues with newer versions or more complex code structures, please open an issue on GitHub.

Caution 5: Custom exports and variables from the `Participant` `Session` classes are not part of the codebook. Also built-in variables are not presented in the codebook.

Further info: `None` values are presented as `"None"` (i.e. as a string) in the list and the file output.

Value

The function returns two main types of outputs:

(a) a list of variables along with their information

(b) a file containing the codebook for the experiment

If `doc_info` is `TRUE` it also returns a message containing the names of all variables that have no documentation.

Examples

```
# The examples use a slightly modified version of the official oTree
# sample codes.

# Make a codebook and resort the apps
combined_codebook <- codebook(
  path = system.file("extdata/ocode_new", package = "gmoTree"),
  output = "list",
  fsource = "init",
  doc_info = FALSE)

# Show the structure of the codebook
str(combined_codebook, 1)
str(combined_codebook$bargaining$Player, 1)

# Make a codebook with only the "bargaining" app
combined_codebook <- codebook(
  path = system.file("extdata/ocode_new", package = "gmoTree"),
  output = "list",
  fsource = "init",
  app = "bargaining",
  doc_info = FALSE)
```

```
# Show the structure of the codebook
str(combined_codebook, 1)
str(combined_codebook$bargaining$Player, 1)

# Make a codebook with all but the "bargaining" app
combined_codebook <- codebook(
  path = system.file("extdata/ocode_new", package = "gmoTree"),
  output = "list",
  fsource = "init",
  app_rm = "bargaining",
  doc_info = FALSE)

# Show the structure of the codebook
str(combined_codebook, 1)
str(combined_codebook$bargaining$Player, 1)

# Use oTree code in 3.x format
combined_codebook <- codebook(
  path = system.file("extdata/ocode_z", package = "gmoTree"),
  fsource = "model",
  output = "list",
  doc_info = FALSE)

# Show the structure of the codebook
str(combined_codebook, 1)

# Show information on missing documentation or complex code
combined_codebook <- codebook(
  path = system.file("extdata/ocode_new", package = "gmoTree"),
  fsource = "init",
  output = "list",
  app_rm = "bargaining",
  doc_info = TRUE)

## Not run:

# Create a codebook PDF with authors' names and todays' date
codebook(
  path = system.file("extdata/ocode_z", package = "gmoTree"),
  fsource = "init",
  doc_info = FALSE,
  output = "file",
  output_format = "pdf_document",
  date = "today",
  title = "My Codebook",
  subtitle = "codebook created with gmoTree",
  params = list(author = c("Max Mustermann", "John Doe"))
  )

# Create a codebook PDF and save it in a subfolder of the
# current folder:
# "C:/Users/username/folder/R_analyses/cb/cb.pdf"
getwd() # "C:/Users/username/folder/R_analyses"
```

```
dir.create("cb")
combined_codebook <- gmoTree::codebook(
  path = "C:/Users/username/folder/R_analyses/oTree",
  fsource = "models",
  output = "both",
  output_file = "cb/cb.pdf",
  output_format = "pdf_document")

# You can also omit *.pdf after the file name
combined_codebook <- gmoTree::codebook(
  path = "C:/Users/username/folder/R_analyses/oTree",
  fsource = "models",
  output = "both",
  output_file = "cb/cb",
  output_format = "pdf_document")

## End(Not run)
```

`delete_cases` *Delete specific cases*

Description

Delete specified cases from all data frames in the list of data frames.

Caution 1: This function does not delete cases from the original CSV and Excel files!

Caution 2: This function does not delete cases from custom data frames if these data frames do not have a variable named `participant.code`!

Caution 3: This function does not delete any data from the `$Chats` data frame! (As the interpretation of chat data depends on how participants engage with each other, the data must be deleted with more care than deleting data in other apps. Hence, this function does not delete data in this data frame. Please do this manually if necessary!)

Usage

```
delete_cases(
  oTree,
  pcodes = NULL,
  plabels = NULL,
  saved_vars = NULL,
  reason,
  omit = FALSE,
  info = FALSE
)
```

Arguments

oTree	A list of data frames created with import_otree().
pcodes	Character string or character vector. The value(s) of the participant.code variable of the participant(s) whose data should be removed.
plabels	Character string or character vector. The value(s) of the participant.label variable of the participant(s) whose data should be removed.
saved_vars	Character string or character vector. The name(s) of variable(s) that need(s) to be stored in the list of information on deleted cases in $info$deleted_cases.
reason	Character string. The reason for deletion that should be stored in the list of information on deleted cases in $info$deleted_cases.
omit	Logical. TRUE if the deleted cases should not be added to the information on deleted cases in $info$deleted_cases.
info	Logical. TRUE if a brief information on the case deletion process should be printed.

Value

This function returns a duplicate of the original oTree list of data frames that do not include the deleted cases.

It adds information on the deleted cases to $info$deleted_cases. (This list is also filled by other functions.)

In this list, you can find the following information:

- $codes = A vector with the participant codes of all deleted cases.
- $count = The number of participants in $codes.
- $full and $unique = The data frames $full and $unique contain information on each deleted participant and the reason why they were deleted. The entries to the $full and the $unique data frames are the same. Columns end_app and end_page are left empty intentionally because they are only filled by the delete_dropouts() function.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Delete only one case
oTree2 <- delete_cases(oTree,
  pcodes = "xmxl46rm",
  reason = "requested")

# Show changes in row numbers
print(paste("Row numbers before deletion: ",
  nrow(oTree$all_apps_wide), nrow(oTree$survey),
  nrow(oTree$Time), nrow(oTree$Chats)))

print(paste("Row numbers after deletion: ",
  nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
```

```
  nrow(oTree2$Time), nrow(oTree2$Chats)))

# Delete several cases
deletionlist <- c("4zhzdmzo", "xmxl46rm")
oTree2 <- delete_cases(oTree,
  pcodes = deletionlist,
  reason = "requested")

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
  nrow(oTree2$Time), nrow(oTree2$Chats)))

# Show information on all deleted cases (also dropouts):
oTree2$info$deleted_cases$full

# Save one variable
oTree2 <- delete_cases(oTree,
  pcodes = deletionlist,
  reason = "requested",
  saved_vars = "participant._index_in_pages")

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
  nrow(oTree2$Time), nrow(oTree2$Chats)))

# Check the "full" deletion information
oTree2$info$deleted_cases$full

# Save some variables
oTree2 <- delete_cases(oTree,
  pcodes = deletionlist,
  reason = "requested",
  saved_vars = c(
    "participant._index_in_pages",
    "participant._max_page_index"))

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
  nrow(oTree2$Time), nrow(oTree2$Chats)))

# Check the "full" deletion information
oTree2$info$deleted_cases$full

# Get a list of all deleted cases
# (If there is already a list, the new list is added to it)
oTree2$info$deleted_cases$codes

# Show number of all deleted cases
length(oTree2$info$deleted_cases$codes)
oTree2$info$deleted_cases$count

# Delete a session and delete a single case from another session
oTree2 <- delete_sessions(oTree,
```

```
  scodes = c("vd1h01iv"),
  reason = "Server Crash",
  saved_vars = "dictator.1.group.id_in_subsession")
oTree2 <- delete_cases(oTree2,
  pcodes = "4zhzdmzo",
  reason = "requested")

# Check the "full" deletion information
oTree2$info$deleted_cases$full

# See codes of deleted variables
oTree2$info$deleted_cases$codes

# See number of deleted variables
oTree2$info$deleted_cases$count
```

`delete_dropouts` *Delete dropouts*

Description

Delete the data of all participants who did not end the experiment at (a) certain page(s) and/or app(s).

Caution 1: This function does not delete cases from the original CSV and Excel files!

Caution 2: This function does not delete cases from custom export data frames if they do not have a variable named `participant.code` and a variable named `session.code`!

Caution 3: This function does not delete any data from the `$Chats` data frame! (As the interpretation of chat data depends on how participants engage with each other, the data must be deleted with more care than deleting data in other apps. Hence, this function does not delete data in this data frame. Please do this manually if necessary!)

Usage

```
delete_dropouts(
  oTree,
  final_apps = NULL,
  final_pages = NULL,
  saved_vars = NULL,
  inconsistent = NULL,
  reason = "ENC",
  info = FALSE
)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.

final_apps	Character string or character vector. The name(s) of the app(s) at which the participants have to finish the experiment.
final_pages	Character string or character vector. The name(s) of the page(s) at which the participants have to finish the experiment.
saved_vars	Character string or character vector. The name(s) of variable(s) that need(s) to be stored in the list of information on deleted cases in $info$deleted_cases.
inconsistent	Character string. Should the function continue or be stopped if at least one participant has inconsistent end_pages, inconsistent end_apps, or both? To continue, type "yes", to stop the function, type "no".
reason	Character string. The reason for deletion that should be stored in the list of information on deleted cases in $info$deleted_cases.
info	Logical. TRUE if a brief information on the dropout deletion process should be printed.

Value

This function returns a duplicate of the original list of data frames but without the deleted cases.

It adds information on the deleted cases to $info$deleted_cases. (This list is also filled by other functions.)

In this list, you can find the following information:

- $full = A data frame that contains information on all participants who did not finish the study; it shows their participant codes, the names of the apps in which they left the experiment, the names of the pages in which they left the experiment, the names of the app data frames in which this information was found, and the dropout reason ("ENC", experiment not completed, combined with the name of the data frame in which the dropout was observed). Because participants usually appear in multiple app data frames, the $info$deleted_cases$full data frame may contain several entries for each person.
- $unique = A data frame that contains similar information as the $full data frame but with only one row per participant and no information on the data frame in which the dropout was observed.
- $all_end = A table that provides information on the app and page combinations where participants ended the experiment. This table also includes information for participants who did not drop out of the experiment. The $all_end table is only shown if an $all_apps_wide data frame exists.
- $codes = A vector containing the participant codes of all deleted participants.
- $count = The number of all deleted participants.

It is important to note that if only the argument final_pages is set, this function does not distinguish between page names that reoccur in different apps.

If the columns end_app and end_page in the output are empty, these variables were not saved by oTree for the specific participants. This could be because empty rows were not deleted. This can be done by using the argument del_empty = TRUE when using import_otree().

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# First, show some row numbers
print(paste(nrow(oTree$all_apps_wide), nrow(oTree$survey),
nrow(oTree$Time), nrow(oTree$Chats)))

# Delete all cases that didn't end the experiment on the page "Demographics"
# within the app "survey"
oTree2 <- delete_dropouts(oTree,
                          final_apps = c("survey"),
                          final_pages = c("Demographics"))

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
nrow(oTree2$Time), nrow(oTree2$Chats)))

# Delete all cases that didn't end the experiment on the page "Demographics"
# This page can be in any app
oTree2 <- delete_dropouts(oTree, final_pages = "Demographics")

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
nrow(oTree2$Time), nrow(oTree2$Chats)))

# Delete all cases that didn't end the experiment on
# any page in the app "survey"
oTree <- delete_dropouts(oTree, final_apps = "survey")

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
nrow(oTree2$Time), nrow(oTree2$Chats)))

# Get list of information on all deleted cases
# (If there is already a list, the new list is added to it!)
oTree2$info$deleted_cases
```

`delete_duplicate` *Delete duplicate data*

Description

Delete duplicate rows from all app data frames and `$all_apps_wide`.

Usage

```
delete_duplicate(oTree)
```

Arguments

oTree	A list of data frames created with import_otree().

Value

This function returns a duplicate of the original oTree list of data frames but without duplicate rows in all app data frames and $all_apps_wide. This function does not modify the custom export data frames and the the $Time and $Chats data frames.

This function does NOT add information to $info$deleted_cases, because it does not delete any important information but only cleans up a messy data import.

However, the function adjusts $info$initial_n, if an $all_apps_wide data frame exists.

Examples

```
# Set data folder first
withr::with_dir(system.file("extdata", package = "gmoTree"), {

# Import all oTree files in this folder and its subfolders
oTree <- import_otree()
})

# First, show some row numbers
print(paste(nrow(oTree$all_apps_wide), nrow(oTree$survey),
            nrow(oTree$Time), nrow(oTree$Chats)))

# Delete duplicate rows
oTree <- delete_duplicate(oTree)

# Show row numbers again
print(paste(nrow(oTree$all_apps_wide), nrow(oTree$survey),
             nrow(oTree$Time), nrow(oTree$Chats)))
```

---

delete_plabels *Delete participant labels in all apps*

Description

If you work with MTurk, the MTurk IDs will be stored in the participant labels variable. This function deletes this variable in $all_apps_wide and every app data frame in the list of data frames that was created by import_otree() and/or all variables referring to MTurk, such as participant.mturk_worker_id.

Caution: This function does not delete the variables from the original CSV and Excel files!

Usage

```
delete_plabels(oTree, del_plabel = TRUE, del_mturk = TRUE)
```

Arguments

oTree	A list of data frames created with import_otree().
del_plabel	Logical. TRUE if all participant labels should be deleted.
del_mturk	Logical. TRUE if all MTurk variables should be deleted.

Value

This function returns a duplicate of the original oTree list of data frames that do not include the participant labels and/or the MTurk variables.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Show participant labels
oTree$all_apps_wide$participant.label
oTree$survey$participant.label

# Delete all participant labels
oTree2 <- delete_plabels(oTree)

# Show participant labels again
oTree2$all_apps_wide$participant.label
oTree2$survey$participant.label
```

---

delete_sessions *Delete all cases of one session*

Description

Delete cases from specific sessions in all data frames within the list of data frames.

Caution 1: This function does not delete cases from the original CSV and Excel files!

Caution 2: This function does not delete cases from custom exports if the custom exports do not have a variable named participant.code and a variable named session.code!

Usage

```
delete_sessions(oTree, scodes, saved_vars = NULL, reason, info = FALSE)
```

Arguments

oTree	A list of data frames created with import_otree().
scodes	Character string or character vector. The session.code(s) of the session(s) whose data should be removed.
saved_vars	Character string or character vector. The name(s) of variable(s) that need(s) to be stored in the list of information on deleted cases in $info$deleted_cases.

reason	Character string. The reason for deletion that should be stored in the list of information on deleted cases in $info$deleted_cases.
info	Logical. TRUE if a brief information on the session deletion process should be printed.

Value

This function returns a duplicate of the original oTree list of data frames that do not include the deleted sessions.

It adds information on the deleted cases to $info$deleted_cases. (This list is also filled by other functions.)

In this list, you can find the following information:

- $full and $unique = The data frames $full and $unique contain information on all participants whose data were deleted. The entries to the $full and the $unique data frames in this list are the same. Columns end_app and end_page are left empty intentionally because they are only filled by the delete_dropouts() function. Columns participant.code and reason are filled.
- $codes = A vector containing the participant codes of all deleted participants.
- $count = The number of all deleted participants.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Delete one session
oTree2 <- delete_sessions(oTree,
  scodes = "7bfqtokx",
  reason = "Only tests")

# Show changes in row numbers
print(paste("Row numbers before deletion: ",
  nrow(oTree$all_apps_wide), nrow(oTree$survey),
  nrow(oTree$Time), nrow(oTree$Chats)))

print(paste("Row numbers after deletion: ",
  nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
  nrow(oTree2$Time), nrow(oTree2$Chats)))

# Delete two sessions and show deletion message
oTree2 <- delete_sessions(oTree,
  scodes = c("7bfqtokx", "vd1h01iv"),
  reason = "Only tests",
  info = TRUE)

# Show row numbers again
print(paste(nrow(oTree2$all_apps_wide), nrow(oTree2$survey),
  nrow(oTree2$Time), nrow(oTree2$Chats)))
```

```
# Delete session and save variable to the info list
oTree2 <- delete_sessions(oTree,
  scodes = c("7bfqtokx", "vd1h01iv"),
  reason = "Server Crash",
  saved_vars = "dictator.1.group.id_in_subsession")

# Check the "full" deletion information
oTree2$info$deleted_cases$full

# See codes of deleted variables
oTree2$info$deleted_cases$codes

# See number of deleted variables
oTree2$info$deleted_cases$count

# Delete a single case and then delete a session
oTree2 <- delete_cases(oTree,
  pcodes = "4zhzdmzo",
  reason = "requested")
oTree2 <- delete_sessions(oTree2,
  scodes = c("vd1h01iv"),
  reason = "Server Crash",
  saved_vars = "dictator.1.group.id_in_subsession")

# Check the "full" deletion information
oTree2$info$deleted_cases$full

# See codes of deleted variables
oTree2$info$deleted_cases$codes

# See number of deleted variables
oTree2$info$deleted_cases$count
```

`extime` *Calculate the time that was spent on the whole experiment*

Description

Calculate the time spent on the experiment. If not stated otherwise, the calculation only starts at the end of the first page!

Usage

```
extime(
  oTree,
  pcode = NULL,
  plabel = NULL,
  group_id = NULL,
  seconds = FALSE,
  rounded = TRUE,
```

```
  digits = 2L,
  startat = 1L,
  tz = "UTC",
  sinfo = "session_code",
  combine = TRUE
)
```

Arguments

oTree	A list of data frames created with import_otree().
pcode	Character string. The value of the participant.code variable if the time should only be calculated for one specified participant.
plabel	Character string. The value of the participant.label variable if the time should only be calculated for one specified participant.
group_id	Integer. The value of the group_id variable if the time should only be calculated for one specified group. The group_id variable can be created with make_ids().
seconds	Logical. TRUE if the output should be in seconds instead of minutes.
rounded	Logical. TRUE if the output should be rounded.
digits	Integer. The number of digits to which the output should be rounded. This parameter has no effect unless rounded = TRUE.
startat	Integer or character string "real" Whether the start of the experiment should be taken from the time at a certain index of each person's vector of page_indexes in the $Time data frame or from the time_started variable in $all_apps_wide ("real"). Important: If integer, it represents the position within the page index sequence, not the numeric value of the page_index variable.
tz	Character string. Time zone.
sinfo	Character string. "session_id" to use session ID for additional information in the data frame of single durations, "session_code" to use session codes, or NULL if no session column should be shown.
combine	Logical. TRUE if all variables referring to epoch time should be merged, and all variables referring to participant code should be merged in case data of several versions of oTree are used. If FALSE, the function returns an error if several oTree versions' data are present.

Details

This functions calculates the time spent on the experiment by using the variable that refers to the time stamp. If that variable is not present, the function alternatively uses seconds_on_page2, which can be created with the pagesec() function.

Value

This function returns either a single value if only the data of one person is calculated or a list of information on the time several participants spent on the experiment.

In this list, you can find the following information:

- $mean_duration = The experiment's average duration.
- $min_duration = The experiment's minimum duration.
- $max_duration = The experiment's maximum duration.
- $single_durations = A data frame of all durations that are used for calculating the min, max, and mean duration.
- $messages = All important notes to the calculations.
- $only_one_page = A vector of all individuals who only have one time stamp.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Show time for one participant
extime(oTree, pcode = "wk247s9w")

# Make a data frame of durations
extime(oTree)

# Make a data frame of durations (beginning from the end of the second page)
extime(oTree, startat = 2)
```

`import_otree` *Import oTree data*

Description

Import data files that were created by oTree.

All files containing the pattern YYYY-MM-DD at the end of their file names are considered oTree files. Bot outputs are saved by oTree without the date included. Hence, to import bot data, you must either rename the original bot files using the YYYY-MM-DD format or use the argument `onlybots = TRUE`. By using the second option, only data of bot files are imported. Since custom export files share the same names as the app files, they must be renamed by adding 'custexp_' as a prefix to correctly import them.

Caution! Data can be downloaded from within the session and globally at the same time. If both files are downloaded, this can lead to the `$all_apps_wide` data being there twice! You can remove duplicate data by using `delete_duplicate()`.

Caution! When importing Excel files, this function does not check for erroneous data structures and will combine all data frames with the same file name patterns. Before using the `CSV = FALSE` argument, clean up your data appropriately.

Usage

```
import_otree(
  path = ".",
  file_names = NULL,
  final_apps = NULL,
  final_pages = NULL,
  recursive = TRUE,
  csv = TRUE,
  onlybots = FALSE,
  del_empty = TRUE,
  info = FALSE,
  encoding = "UTF-8"
)
```

Arguments

path	Character string or character vector. The path(s) to the files (default is the working directory).
file_names	Character string or character vector. The name(s) of the file(s) to be imported. If not specified, all files in the path and subfolders are imported.
final_apps	Character string or character vector. The name(s) of the app(s) at which the participants have to finish the experiment. If the argument final_apps is left empty, you can still call for deleting the participants who did not finish the experiment with delete_dropouts().
final_pages	Character string or character vector. The name(s) of the page(s) at which the participants have to finish the experiment. If the argument final_pages is left empty, you can still call for deleting the participants who did not finish the experiment with delete_dropouts().
recursive	Logical. TRUE if the files in the path's subfolders should also be imported.
csv	Logical. TRUE if only CSV files should be imported. FALSE if only Excel files should be imported.
onlybots	Logical. TRUE if only bot-created files should be imported.
del_empty	Logical. TRUE if all empty cases should be deleted from the $all_apps_wide or normal app data frames (not Time or Chats).
info	Logical. TRUE if a brief information on the data import should be printed.
encoding	Character string. Encoding of the CSV files that are imported. Default is "UTF-8".

Value

Returns a list of data frames (one data frame for each app and $all_apps_wide) and a list of information on this list of data frames in $info.

See detailed information on the imported files in $info$imported_files.

If $all_apps_wide is imported, see the number of imported cases in $info$initial_n. In this number, empty rows are already considered. So, if empty rows are deleted with del_empty=TRUE,

initial_n counts all rows that are not empty. Cases that are deleted because the participants did not make it to the last page and/or app are not subtracted from this number.

Information: Empty rows are rows without the participant._current_app_name variable set. Empty rows are deleted from all app data frames and $all_apps_wide when using del_empty = TRUE. Empty rows in the $Chats and $Time data frames as well as data frames whose names start with custexp_ (custom export) are not deleted.

If old and new oTree versions are combined, the $Time data frame contains variables called participant_code and participant__code (the difference is in the underscores). Caution! If there is an unusual amount of NAs, check if everything got imported correctly. Sometimes, the CSV or Excel file may be corrupted, and all information is only found in one column.

Examples

```
# Set data folder first
withr::with_dir(system.file("extdata", package = "gmoTree"), {

# Import all oTree files in this folder and its subfolders
oTree <- import_otree()

# Show the structure of the import
str(oTree, max.level = 1)

# Show the names of all imported files
oTree$info$imported_files

# Delete empty cases and delete every case of a person
# who didn't end the experiment in the app "survey"
oTree <- import_otree(
  del_empty = TRUE,
  final_apps = "survey",
  info = TRUE)

# Show the structure of the import
str(oTree, max.level = 1)

# Import bot files
import_otree(
  path = "./bot_data",
  onlybots = TRUE,
  csv = TRUE,
  info = TRUE)

# Show the structure of the import
str(oTree, max.level = 1)

# Import with file names (path separately)
oTree2 <- import_otree(
     del_empty = TRUE,
     path = "./exp_data",
     file_names = c("all_apps_wide-2023-03-27.csv",
                    "ChatMessages-2023-03-27.csv",
```

```
                    "PageTimes-2023-03-27.csv"),
      onlybots = FALSE,
      csv = TRUE,
      info = TRUE)

# Show the structure of the import
str(oTree, max.level = 1)

# Import with file names (without path separately)
oTree2 <- import_otree(
      del_empty = TRUE,
      file_names = c("exp_data/all_apps_wide-2023-03-27.csv",
                     "exp_data/ChatMessages-2023-03-27.csv",
                     "exp_data/PageTimes-2023-03-27.csv"),
      onlybots = FALSE,
      csv = TRUE,
      info = TRUE)

# Show the structure of the import
str(oTree, max.level = 1)
})
```

`make_ids` *Make IDs*

Description

Make session IDs and, optionally, group IDs and participant IDs that span across all data frames created by `import_otree()`. Information for these IDs is taken from `$all_apps_wide` but can be defined otherwise.

Note: Older versions of oTree may already contain a variable called session_id in their `$Time` data frames. This variable is overwritten by this function!

Important: Combine duplicate data before running this function!

Usage

```
make_ids(
  oTree,
  gmake = FALSE,
  pmake = TRUE,
  from_app = "all_apps_wide",
  from_var = NULL,
  sstart = 1L,
  gstart = 1L,
  pstart = 1L,
  emptyrows = NULL,
  icw = FALSE
)
```

Arguments

oTree	A list of data frames created with import_otree().
gmake	Logical. TRUE if a variable called group_id should be made. If from_var is not NULL, gmake is automatically set to TRUE.
pmake	Logical. TRUE if a variable called participant_id should be made.
from_app	Character string. Name of the data frame from which the session, group, and participant information should be taken. All normal app data frames and $all_apps_wide are allowed.
from_var	Character string. Name of the variable from which the group information should be taken. This argument is only relevant when $all_apps_wide is used as from_app and has group information that contradicts each other.
sstart	Integer. The number that serves as a starting point for session IDs.
gstart	Integer. The number that serves as a starting point for group IDs.
pstart	Integer. The number that serves as a starting point for participant IDs.
emptyrows	Character string. "no" if the function should stop if there are empty rows in from_app. "yes" if the function should continue to make IDs.
icw	Logical. TRUE if the warning message should be ignored that states that IDs cannot be made because of an oTree bug.

Value

ID variables are made in $all_apps_wide, all app data frames, the $Time data frame, and the $Chats data frame. See list of the additional ID variables in $info$additional_variables.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Make session IDs only
oTree2 <- make_ids(oTree)

# Show new variables
oTree2$all_apps_wide$session_id

# Make session IDs and group IDs
# Not working with this data set because group ID is not the same in all apps
## Not run:
      oTree2 <- make_ids(oTree, gmake = TRUE)

      # Show new variables
      oTree2$all_apps_wide$session_id
      oTree2$all_apps_wide$group_id

## End(Not run)

# Get IDs from variable "dictator.1.group.id_in_subsession"
```

```
# in the data frame "all_apps_wide"
oTree2 <- make_ids(oTree,
                   gmake = TRUE,
                   from_var = "dictator.1.group.id_in_subsession")

# Show new variables
oTree2$all_apps_wide$session_id
oTree2$all_apps_wide$group_id

# Get IDs from another app than all_apps_wide
oTree2 <- make_ids(oTree, gmake = TRUE, from_app = "dictator")

# Show new variables
oTree2$all_apps_wide$session_id
oTree2$all_apps_wide$group_id
```

`messy_chat` *Check if the Chats data frame is messy*

Description

Check if the `$Chats` data frame includes both session-related variables and participant-related variables that appear multiple times. This may occur when data from different oTree versions, which use different variable names, are combined.

If desired, the function can merge these variables, storing the data using the newer oTree version's variable names and removing the outdated variables.

Usage

```
messy_chat(
  oTree,
  combine = FALSE,
  session = TRUE,
  participant = TRUE,
  info = FALSE
)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`combine`	Logical. `TRUE` if all variables referring to the session code should be merged and/or all variables referring to participant code should be merged in case data of several versions of oTree are used.
`session`	Logical. `TRUE` if all variables referring to the session code should be checked and merged. Merging only works if `combine = TRUE`.
`participant`	Logical. `TRUE` if all variables referring to the participant code should be checked and merged. Merging only works if `combine = TRUE`.
`info`	Logical. `TRUE` if a brief information on the process should be printed.

Value

This function searches for multiple variables related to the session code or the participant code in the `$Chats` data frame. which can occur when data from both old and new oTree versions are used.

If `combine = FALSE`, the function will throw an error if such variables are found.

If `combine = TRUE`, the function will not throw an error if such variables are found. Instead, it automatically combines the variables into new variables and adds them to the original `$Chats` data frame. This function then returns a duplicate of the original oTree list but with the `$Chats` data frame modified.

The new variables are called `participant_code` and `session_code`.

Examples

```
# Set data folder first
withr::with_dir(system.file("extdata", package = "gmoTree"), {

# Import all oTree files in this folder and its subfolders
oTree <- import_otree()
})

# Show all Chats column names
print(colnames(oTree$Chats))

# Run function
oTree <- messy_chat(oTree, combine = TRUE)

# Show all Chats column names again
print(colnames(oTree$Chats))
```

---

messy_time *Check if the Time data frame is messy*

Description

Checks if the Time data frame includes both participant-related variables and time stamp variables that appear multiple times. This may occur when data from different oTree versions, which use different variable names, are combined.

If desired, the function can merge these variables, storing the data using the newer oTree version's variable names and removing the outdated variables.

Usage

```
messy_time(
  oTree,
  combine = FALSE,
  epoch_time = TRUE,
  participant = TRUE,
```

```
  info = FALSE
)
```

Arguments

oTree	A list of data frames created with import_otree().
combine	Logical. TRUE if all variables referring to epoch time should be merged and/or all variables referring to participant code should be merged in case data of several versions of oTree are used.
epoch_time	Logical. TRUE if all variables referring to the time stamp should be checked and merged. Only works if combine = TRUE.
participant	Logical. TRUE if all variables referring to the participant code should be checked and merged. Only works if combine = TRUE.
info	Logical. TRUE if a brief information on the process should be printed.

Value

This function searches for multiple variables related to the time stamp or the participant code in the $Time data frame, which can occur when data from both old and new oTree versions are used.

If combine = FALSE, the function will throw an error if such variables are found.

If combine = TRUE, the function will not throw an error if such variables are found. Instead, it automatically combines the variables into new variables and adds them to the original $Time data frame. This function then returns a duplicate of the original oTree list but with the $Time data frame modified.

The new variables are called epoch_time_completed and participant_code.

Examples

```
# Set data folder first
withr::with_dir(system.file("extdata", package = "gmoTree"), {

# Import all oTree files in this folder and its subfolders
oTree <- import_otree()
})

# Show all Time column names
print(colnames(oTree$Time))

# Run function
oTree <- messy_time(oTree, combine = TRUE)

# Show all Time column names again
print(colnames(oTree$Time))
```

oTree *Sample experimental data*

Description

Sample experimental data

Usage

```
oTree
```

Format

A list of data frames created by import_otree().

Source

The data set was created by using modified versions of the official oTree sample experiments that can be downloaded when installing oTree. In detail, the following apps were used: "start," "dictator," "chatapp," "survey."

pagesec *Calculate the seconds spent on each page*

Description

Create a new variable in the `$Time` data frame that contains the time spent on each page.

Usage

```
pagesec(oTree, rounded = TRUE, digits = 2, minutes = FALSE, combine = FALSE)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`rounded`	Logical. `TRUE` if the output should be rounded.
`digits`	Integer. The number of digits to which the output should be rounded. This parameter has no effect unless `rounded = TRUE`.
`minutes`	Logical. `TRUE` if the output should be minutes instead of seconds.
`combine`	Logical. `TRUE` if all variables referring to epoch time should be merged, and all variables referring to participant code should be merged in case data of several versions of oTree are used.

Value

This function returns a duplicate of the original oTree list of data frames that also contains a column in the `$Time` data frame named `seconds_on_page2` or `minutes_on_page`.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Create two new columns: seconds_on_page2 and minutes_on_page
oTree <- pagesec(oTree, rounded = TRUE, minutes = TRUE)

# Show the Time data frame
head(oTree$Time, n = 30)
```

---

show_constant *Show constant columns*

Description

Show all variables with no variation in their values within each data frame of the list of data frames (except the ones in the info list). This function is useful for identifying unnecessary variables before running an experiment. It allows checking for columns with any unchanging value or only a specific value.

Usage

```
show_constant(oTree, value = "any")
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`value`	The value that is controlled to be the same within a column. If the value is set to `"any"`, the function checks for columns where any possible values are identical.

Value

This function returns a list of vectors, one for each app, `$all_apps_wide`, the `$Time` and/or the `$Chats` data frame. Each vector contains the names of all variables with constant values. If there are no variables with constant values, the vector is empty.

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Show all columns that contain only NAs
show_constant(oTree = oTree)
```

```
show_constant(oTree = oTree, value = NA)

# Show all columns that contain only -99
show_constant(oTree = oTree, value = -99)
```

`show_dropouts` *Show participants who did not finish the experiment*

Description

Show information on the people who did not finish the experiment at (a) certain page(s) and/or app(s).

Usage

```
show_dropouts(oTree, final_apps = NULL, final_pages = NULL, saved_vars = NULL)
```

Arguments

`oTree`	A list of data frames created with `import_otree()`.
`final_apps`	Character string or character vector. The name(s) of the app(s) at which the participants have to finish the experiment.
`final_pages`	Character string or character vector. The name(s) of the page(s) at which the participants have to finish the experiment.
`saved_vars`	Character string or character vector. The name(s) of variable(s) that need(s) to be shown in the list of information on dropout cases.

Value

This function returns a list of information on participants who did not finish the experiment.

In this list, you can find the following information:

- `$full` = A data frame that contains information on all participants who did not finish the study; it shows their participant codes, the names of the apps in which they left the experiment, the names of the pages in which they left the experiment, the names of the app data frames in which this information was found, and the dropout reason (`"ENC"`, experiment not completed, combined with the name of the data frame in which the dropout was observed). Because participants usually appear in multiple app data frames, the `$full` data frame may contain several entries for each person.
- `$unique` = A data frame that contains similar information as the `$full` data frame but with only one row per participant and no information on the data frame in which the dropout was observed.
- `$all_end` = A table that provides information on the app and page combinations where participants ended the experiment. This table also includes information on participants who did not drop out of the experiment. The `$all_end` table is only shown if an `$all_apps_wide` data frame exists.

- $codes = A vector containing the participant codes of all participants who did not finish the experiment.
- $count = The number of all participants who did not finish the experiment.

It is important to note that if only the argument final_pages is set, this function does not distinguish between page names that reoccur in different apps.

If the columns end_app and end_page in the output are empty, these variables were not saved by oTree for the specific participants. This could be because empty rows were not deleted. This can be done by using the argument del_empty = TRUE" when using import_otree().

Examples

```
# Use package-internal list of oTree data frames
oTree <- gmoTree::oTree

# Show everyone who did not finish with the app "survey"
show_dropouts(oTree, final_apps = "survey")

# Show everyone who did not finish with the page "Demographics"
show_dropouts(oTree, final_pages = "Demographics")

# Show everyone who finished with the following apps: "survey," "dictator"
final_apps <- unique(oTree$all_apps_wide$participant._current_app_name)
final_apps <- final_apps[final_apps != "survey"]
final_apps <- final_apps[final_apps != "dictator"]
show_dropouts(oTree, final_apps = final_apps)
```

Index