Package ‘esmtools’

July 22, 2025

Title Preprocessing Experience Sampling Method (ESM) Data

Version 1.0.1

Maintainer Jordan Revol <jordan.revol@kuleuven.be>

Description Tailored explicitly for Experience Sampling Method (ESM) data, it contains a suite of functions designed to simplify preprocessing steps and create subsequent reporting. It empowers users with capabilities to extract critical insights during preprocessing, conducts thorough data quality assessments (e.g., design and sampling scheme checks, compliance rate, careless responses), and generates visualizations and concise summary tables tailored specifically for ESM data. Additionally, it streamlines the creation of informative and interactive preprocessing reports, enabling researchers to transparently share their dataset preprocessing methodologies. Finally, it is part of a larger ecosystem which includes a framework and a web gallery (<https://preprocess.esmtools.com/>).

License GPL (>= 3)

URL https://gitlab.kuleuven.be/ppw-okpiv/researchers/u0148925/esmtools/, https://package.esmtools.com/, https://preprocess.esmtools.com/

BugReports https://gitlab.kuleuven.be/ppw-okpiv/researchers/u0148925/esmtools/-/issues

Depends R (>= 4.0.0)

Imports base64enc (>= 0.1-3), dplyr (>= 1.1.0), DT (>= 0.28), fs (>= 1.6.0), ggplot2 (>= 3.4.0), ggpubr (>= 0.6.0), grDevices, htmltools, jsonlite (>= 1.8.0), kableExtra (>= 1.3.0), knitr (>= 1.43), lubridate (>= 1.9.0), stats, stringr (>= 1.5.0), tidyr (>= 1.3.0), tools

Suggests readxl (>= 1.4.2), spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Author Jordan Revol [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-5511-3617>), Koen Niemeijer [ctb] (ORCID: <https://orcid.org/0000-0002-0816-534X>)

Repository CRAN

Date/Publication 2024-03-13 19:00:02 UTC

Contents

`add_script_to_rmd` *Add Custom CSS and JavaScript to R Markdown Document*

Description

This function adds custom CSS and JavaScript code to an R Markdown document. The function reads CSS and JavaScript code from specific files (e.g., 'button.css' and 'button.js') located in the "esmtools" package. It then creates appropriate HTML elements to include the code within the R Markdown document. This function is automatically called when importing the esmtools package and rendering the rmarkdown package.

Usage

```
add_script_to_rmd()
```

Value

Returns an text object marked as HTML containing the combined CSS and JavaScript code stored within the inst folder of the package.

Examples

```
if (interactive()) {
  add_script_to_rmd()
}
```

---

button   *Create an HTML button with optional toggleable content*

Description

This function generates an HTML button with optional toggleable content. When used in an RMarkdown document, the content following the button will be hidden by default and can be toggled on and off by clicking the button. To use it, simply call the function within inline R code, do not call this function within a chunk.

Usage

```
button(text = "Description")
```

Arguments

text	The text to display on the button. Default is "Description".

Value

A character string containing the HTML code for the button and optional toggleable content. In RMarkdown, this will be displayed as an HTML button. In regular R scripts, this function will return an empty string.

Examples

```
# In RMarkdown, use inline code (e.g., `r button()` and `r endbutton()`)
# to create a toggleable button.

if (interactive()) {
  button("Supplementary materials")
  # Hidden content goes here.
  endbutton()
}
```

`calendar_plot` *Generate a calendar plot*

Description

The 'calendar_plot()' function creates a calendar visualization that displays the number of occurrences of beeps over one or more years in a calendar format. The function utilizes the 'ggplot2' package to create the calendar plot.

Usage

```
calendar_plot(
  .data,
  timevar = NULL,
  interval = "halfyear",
  week_start = getOption("lubridate.week.start", 1),
  weight_heights = NULL
)
```

Arguments

`.data`	A dataframe that contains the time variable
`timevar`	The time variable name
`interval`	Specifies the time interval over which to group and display data in the calendar plot, with options "halfyear" (default) or "year."
`week_start`	When set to 1 (by default), the week starts on Monday, when set to 7, the week starts on Sunday following USA calendar.
`weight_heights`	A list of weights for adjusting the height of individual plots in the output. Default is NULL.

Details

The 'calendar_plot()' function generates a calendar plot where each cell represents a day of the year, and the color intensity of the cell (and the occurence number if text_count=TRUE) reflects the number of beep occurrences on that day. This allows for easy identification of patterns and trends in beep occurrences over time.

Value

A ggplot object, a calendar plot.

Examples

```
if (interactive()) {
  esmdata_sim$sent <- as.POSIXct(esmdata_sim$sent)
  calendar_plot(esmdata_sim, timevar = "sent", interval = "halfyear")
  calendar_plot(esmdata_sim, timevar = "sent", interval = "year")
}
```

`codebook_table` *Create Codebook Table*

Description

The `codebook_table()` function generates a codebook table for a given dataframe, providing descriptive statistics and visualizations for each variable. See an example here: https://preprocess.esmtools.com/pages/90_Codebook_table_esmtools.html.

Usage

```
codebook_table(
  df,
  origin_cbook = NULL,
  origin_vars = "Variable",
  n_unique_thres = 6,
  include_txt = FALSE,
  include_date = TRUE,
  histograms = TRUE,
  boxplots = TRUE,
  html_output = NULL,
  kable_out = TRUE
)
```

Arguments

`df`	The dataframe for which the codebook table is generated.
`origin_cbook`	A dataframe of a hand-maid codebook table that is stored in an csv of xlsx file. The file must be first imported in the R session. If the variable containing the basis codebook, the resulting codebook table will include the original codebook information. This allows merging and incorporating a hand-made codebook into the output. Default is `NULL`.
`origin_vars`	A character string specifying the name of the variable column in the `origin_cbook`. Default is "Variable".
`n_unique_thres`	The threshold for the number of unique values to consider a variable as categorical. Default is 6.
`include_txt`	Logical indicating whether to include text variable statistics in the codebook table. Default is `FALSE`.
`include_date`	Logical indicating whether to include date variable statistics in the codebook table. Default is `TRUE`.
`histograms`	Logical indicating whether to include histograms in the codebook table. Default is `TRUE`.

`boxplots`	Logical indicating whether to include boxplots in the codebook table. Default is `TRUE`.
`html_output`	Define a file name (e.g., "path/to/codebook_table.html") to create an html output version of the codebook table.
`kable_out`	When TRUE, output is in kable version. If FALSE, use DT package.

Value

A a codebook table generated using kable or the DT package.

Examples

```
if (interactive()) {
 # Load library
 library(esmtools)
 library(readxl)

 # Load the hand-made codebook
 path_original <- system.file("extdata", "cbook_part1.xlsx", package = "esmtools")
 original_codebook <- read_excel(path_original)

 # Create codebook table based on the hand-made codebook and the dataset
 codebook_table(df = esmdata_sim, origin_cbook = original_codebook)
}
```

`controws` *Get Context of Rows*

Description

This function extracts specific rows from a data frame along with their context, based on a given range. It can take either a vector of row numbers or a boolean vector as input. Additionally, the function allows specifying the direction of the context (above, below, or both) and marks each row as either targeted or context.

Usage

```
controws(data, input, context = 1, direction = "both")
```

Arguments

`data`	A data frame from which rows and their context will be extracted.
`input`	A numeric vector of row numbers or a logical vector. If a logical vector is given, the function will extract rows where the vector is TRUE.
`context`	An integer specifying the number of rows above and/or below the target row to include as context. Default is 1.
`direction`	A character string specifying the direction of context to include. Valid options are "up" for rows above, "down" for rows below, and "both" for both directions. Default is "both".

Value

A dataframe containing the targeted rows and their contextual rows. An additional column '.RowType' indicates whether it is a row given as input ("->") or part of the context ("").

Examples

```
# Example dataframe
data <- data.frame(matrix(rnorm(100), ncol = 5))

# Get context for rows 8 and 15 (using row numbers)
result <- controws(data, c(8, 15), 1, "both")
result
```

`dataInfo` *Display information regarding the dataset in a succinct way.*

Description

The 'dataInfo()' function displays detailed information about a dataset in a similar style as 'sessionInfo()'. It provides various details such as size, creation and update times, number of columns and rows, number of participants, variable names, and more. This information is useful for reproducibility, tracking the dataset, and ensuring transparency in data analysis workflows.

Usage

```
dataInfo(
  file_path = NULL,
  read_fun = NULL,
  idvar = NULL,
  timevar = NULL,
  validvar = NULL,
  citation = NULL,
  URL = NULL,
  DOI = NULL,
  path = TRUE,
  variables = TRUE
)
```

Arguments

`file_path`	The path or URL of the dataset file.
`read_fun`	The function used to read the dataset file.
`idvar`	The identifier variable(s) in the dataset, represented as a character vector.
`timevar`	A time variable(s) name in the dataset. Preference is to use the sent timestamp variable (the time when the beep was sent to the participant).

`validvar`	The validation variable name in the dataset, represented as a numerical vector. If NULL, the function do not display compliance rate information.
`citation`	A character element to cite the article or document associated with the script.
`URL`	The citation information for the dataset (article associated), represented as a character string. If NULL, the function will not display the citation information.
`DOI`	The Digital Object Identifier (DOI) of the dataset, if applicable. If NULL, the function will not display the DOI information.
`path`	If TRUE, the function will display the path information.
`variables`	A logical value indicating whether to display the names of the dataset's variables. Set to TRUE to display variable information, and FALSE to omit it. The default is TRUE.

Details

The 'dataInfo()' function provides a comprehensive summary of information about the dataset. The information returned includes:

- Size: The size of the dataset in octets.
- File extension
- Creation and Update Times: The date and time when the data file was created and last updated.
- Number of Columns and Rows
- Number of Participants
- Average Observations per Participant
- Compliance Mean: The mean compliance value for the dataset.
- Data Collection Period: The duration or period during which the data was collected.
- Path: The path or URL of the dataset file.
- Variable Names: The names of the variables in the dataset.
- Associated Links: Any associated URL, DOI, or citation links for the dataset.

Value

The 'dataInfo()' function displays detailed information about the dataset. It can also be store as a list in a variable.

A kable object that summarizes the information on the data, the current R session, and the article or document associated with the script.

Examples

```
library(dplyr)

# Load data
file_path <- system.file("extdata", "esmdata_sim.csv", package = "esmtools")

# Create a function to read the data
read_fun <- function(x) read.csv2(x) %>%
    mutate(sent = as.POSIXct(as.character(sent), format="%Y-%m-%d %H:%M:%S"))

# Get data information
dataInfo(
  file_path = file_path, read_fun = read_fun,
  idvar = "id", timevar = "sent"
)
```

---

endbutton — *End the HTML button container*

Description

This function ends the HTML container for the button and optional toggleable content created using the 'button()' function. When used in an RMarkdown document, it closes the container and ensures proper rendering. To use it, simply call the function within inline R code, do not call this function within a chunk.

Usage

```
endbutton()
```

Value

A character string containing the closing HTML tags for the button container. In RMarkdown, this will ensure that the content following the button is properly displayed. In regular R scripts, this function will return an empty string.

Examples

```
# In RMarkdown, use inline code (e.g., `r button()` and `r endbutton()`)
# to create a toggleable button.

if (interactive()) {
  button("Supplementary materials")
  # Hidden content goes here.
  endbutton()
}
```

`esmdata_preprocessed` *Preprocessed ESM data set*

Description

This dataset has undergone random value alterations to ensure privacy and cannot be utilized for formal research purposes. For additional information, refer to the associated article.

Usage

```
esmdata_preprocessed
```

Format

A data frame with 6 rows and 20 columns:

dyad Dyad identification number

id Participant identification number

role Role of a participant within a dyad, indicated by a character (0=father or 1=mother)

age Age of the participant

compliance Participant's proportion of completed surveys

obsno ESM questionnaire (beep) number indicating serial order

daycum Cumulative day count

beepno Beep number within a day

valid Indicator of observation validity (1=valid, 0=invalid)

scheduled Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was scheduled

sent Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was sent

start Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was opened by the participant

end Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was ended by the participant

pos_aff Positive affect score (0-100)

neg_aff Negative affect score (0-100)

perc_stress_child parents reported if their child did experience a stressful event since the last beep

perc_fun_child parents reported if their child had experienced anything fun since the last beep (yes/no)

perc_fun_signaled parents rated to which extent their child showed they experienced this fun event (0-100)

pos_aff_pc Person-centered positive affect score

neg_aff_pc Person-centered negative affect score

`esmdata_raw` *Raw ESM data set*

Description

Raw dataset from a pilot ESM study. This dataset has undergone random value alterations to ensure privacy and cannot be utilized for formal research purposes. For additional information, refer to the associated article.

Usage

```
esmdata_raw
```

Format

A data frame with 6 rows and 13 columns:

dyad Dyad identification number

id Participant identification number

role Role of a participant within a dyad, indicated by a character (0=father or 1=mother)

age Age of the participant

scheduled Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was scheduled

sent Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was sent

start Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was opened by the participant

end Timestamps (e.g., "2023/04/14 10:23:47") of when the ESM questionnaire was ended by the participant

pos_aff Positive affect score (0-100)

neg_aff Negative affect score (0-100)

perc_stress_child parents reported if their child did experience a stressful event since the last beep

perc_fun_child parents reported if their child had experienced anything fun since the last beep (yes/no)

perc_fun_signaled parents rated to which extent their child showed they experienced this fun event (0-100) esmdata_sim *Simulated ESM Data Set*

Description

Mimic the structure of a dataset from a heterosexual romantic couples study (i.e., ESM dyadic design). This simulated study focuses on the emotional affect dynamics of romantic partners taking into account the context (location and contact with the partner). In addition, this study assumes that couples are randomized into two treatments. Thus, the dataset includes a baseline phase and a treatment phase, where couples are randomly allocated to one of the two treatments. Here are the general characteristics of the dataset. More information on: https://preprocess.esmtools.com/terminology.html/

Usage

```
esmdata_sim
```

Format

A data frame with 4200 rows and 18 columns:

dyad Dyad identification number

id Participant identification number

role Role of a participant within a dyad (make the partner distinguishable)

age Age of the participant

cond_dyad Treatment condition to which the dyad was assigned, as follows: 0=control group and 1=experimental group

obsno ESM questionnaire (beep) number of the observation that indicates their serial order

scheduled Timestamps (e.g., “2023/04/14 10:23:47”) of when the ESM questionnaire was scheduled

sent Timestamps (e.g., “2023/04/14 10:23:47”) of when the ESM questionnaire was sent

start Timestamps (e.g., “2023/04/14 10:23:47”) of when the ESM questionnaire was opened by the participant

end Timestamps (e.g., “2023/04/14 10:23:47”) of when the ESM questionnaire was item ended by the participant

PA1 A positive affect (PA) item with a slider scale (1-100)

PA2 A positive affect (PA) item with a slider scale (1-100)

PA3 A positive affect (PA) item with a slider scale (1-100)

NA1 A negative affect (NA) item with a slider scale (1-100)

NA2 A negative affect (NA) item with a slider scale (1-100)

NA3 A negative affect (NA) item with a slider scale (1-100)

location Categorical item with 4 possible answers (home, work, public space, other)

contact Dichotomous item (1=contact, 0=no contact) `folrows` *Extract Following Rows from a Dataframe*

Description

The `folrows()` function extracts a specified number of consecutive rows, including the starting row, from a dataframe. Rows are selected consecutively from randomly chosen starting positions. The function ensures that the selected rows fall within the valid row range of the dataframe.

Usage

```
folrows(.data, n = 5, nb_sample = 1)
```

Arguments

`.data`	The dataframe from which rows are to be extracted.
`n`	The number of consecutive rows to be extracted, including the starting row. Default is 5.
`nb_sample`	The number of random starting positions to be sampled. Default is 1.

Value

A dataframe containing the consecutive rows extracted from the randomly chosen starting positions.

Examples

```
# Extract 3 consecutive rows starting from a random position in the dataset
folrows(esmdata_sim, n = 3)

# Extract 4 consecutive rows starting from 2 random positions in a custom dataframe
folrows(esmdata_sim, n = 4, nb_sample = 2)
```

`getEsmtoolsEnv` *Get the internal package environment*

Description

This function provides access to the internal environment used by the esmtools package. This environment is used for internal state management and should be used with caution.

Usage

```
getEsmtoolsEnv()
```

Value

An environment object used internally by the esmtools package.

Examples

```
esmtools_env <- getEsmtoolsEnv()
print(esmtools_env)
```

`heatcalendar_plot` *Generate a heatmap calendar plot*

Description

The 'heatcalendar_plot()' function creates a heatmap-style calendar visualization that represents the density of events over time. Each cell in the heatmap represents a specific day, and its color intensity reflects the number of events that occurred on that day. This function utilizes the 'ggplot2' R package to create the heatmap calendar plot.

Usage

```
heatcalendar_plot(
  .data,
  timevar = NULL,
  week_start = getOption("lubridate.week.start", 1)
)
```

Arguments

`.data`	A dataframe that contains the time variable
`timevar`	The time variable name
`week_start`	When set to 1 (by default), the week starts on Monday, when set to 7, the week starts on Sunday following USA calendar.

Details

The 'heatcalendar_plot()' function generates a heatmap-style calendar plot where each cell corresponds to a day of the year. The color intensity of each cell represents the event density, allowing for the identification of patterns and trends in event occurrences over time.

Value

A heatmap calendar plot

Examples

```
if (interactive()) {
  esmdata_sim$sent <- as.POSIXct(as.character(esmdata_sim$sent))
  heatcalendar_plot(esmdata_sim, timevar = "sent")
}
```

`longstring` *Calculate Longstring Value*

Description

This function calculates the longstring value for a given vector of values.

Usage

```
longstring(x, tolerance = NULL)
```

Arguments

`x`	A vector of values for which the longstring value needs to be calculated.
`tolerance`	An optional parameter indicating the tolerance for considering consecutive values as equal (default is NULL).

Value

A dataframe containing the following information:

val The longstring value(s) based on the most frequent value(s) or the unique value(s).

longstr The length of the longest run of consecutive values.

avgstr The average length of non-NA runs of consecutive values (not computed when tolerance is used).

Examples

```
df <- data.frame(
  rbind(
    c(1, 1, 2, 2, 2, 3, 3, 3, 3),
    c(1, 2, 3, 4, 4, 4, 4, 2, 6)
  )
)
# Example 1: Without tolerance
longstring(df)

# Example 2: With tolerance
longstring(df, tolerance = 1)
```

participants_book *Generate a participants' book*

Description

The 'participants_book()' function creates a participants' book, which provides a concise summary of participants' data from an intensive longitudinal study, such as Experience Sampling Method (ESM). The book is generated using the DT package, and each participant is represented by a row in the table. The participants' book displays various information about participants' response behaviors, including compliance rate, study duration, and start time. Additionally, it provides descriptive statistics and time series plots for the variables of interest. See an example here: https://preprocess.esmtools.com/pages/90_Participant_book.html.

Usage

```
participants_book(
  df,
  idvar = "id",
  obsnovar = "obsno",
  focusvar = NULL,
  timevar = NULL,
  validvar = NULL,
  compliancevar = NULL,
  obsno_max = NULL,
  list_behaviors = c("min_date", "max_date", "nb_answer", "compliance"),
  list_stats = c("mean", "sd", "n_length", "n_unique"),
  viz = list(c("ts", "hist")),
  html_output = NULL,
  kable_out = TRUE,
  min_max_regularize = TRUE
)
```

Arguments

`df`	A dataframe containing the participant data
`idvar`	The name of the column in the dataframe that represents the participant identifier
`obsnovar`	The name of the column in the dataframe that represents the beep number
`focusvar`	A vector of variable names representing the variables of interest in the dataframe
`timevar`	The name of the column in the dataframe that represents the timestamp
`validvar`	The name of the column in the dataframe that represents the validity of responses
`compliancevar`	The name of the column in the dataframe that represents the compliance rate (alternatively, if NULL, the compliance score is computed as the number of rows divided by obsno_max)
`obsno_max`	The maximum number of beeps for computing the compliance score (only used if compliancevar is NULL)

`list_behaviors`	A vector indicating the types of participants' response behavior information to display. Valid options are "min_date", "max_date", "nb_answer", and "compliance".
`list_stats`	A vector indicating the types of descriptive statistics to display. Valid options are "mean", "sd", "range", "n_length", and "n_unique".
`viz`	A vector indicating the visualization type to display for each focusvar variable. Valid options are "ts" (time series plot) and "hist" (histogram).
`html_output`	Define a file name (e.g., "path/to/participant_book.html") to create an html output version of the participant book.
`kable_out`	When TRUE, output is in kable version. If FALSE, use DT package.
`min_max_regularize`	Whether to regularize the y-axis or x-axis limits across plots based on the global minimum and maximum values. Default is TRUE.

Value

A kable or datatable object (from the DT package), representing the participants' book.

Examples

```
participants_book(esmdata_preprocessed,
  idvar = "id",
  obsnovar = "obsno",
  focusvar = c("pos_aff", "neg_aff"),
  timevar = "start",
  validvar = "valid",
  obsno_max = 70,
  list_behaviors = c("min_date", "max_date", "nb_answer", "compliance"),
  list_stats = c("mean", "sd", "range", "n_length", "n_unique"),
  viz = list(c("ts", "hist"), c("ts", "hist"))
)
```

`print.dataInfo` *Print method of the dataInfo object*

Description

This function provides a customized print method for objects created by the 'dataInfo()' function. It displays each element of the data information list in a user-friendly format.

Usage

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

Arguments

`x`	A list of data information from the 'dataInfo' function.
`...`	Dots are for compatibility and not used.

Value

The function returns the kable object invisibly from the 'dataInfo()' function, allowing for its use in further function calls or command chaining without printing the object again.

---

randrows — *Randomly Sample Rows from a Dataframe*

Description

The `randrows()` function randomly samples a specified number of rows from a dataframe.

Usage

```
randrows(.data, n = 5)
```

Arguments

`.data`	The dataframe from which rows are to be sampled.
`n`	The number of rows to be randomly sampled. Default is 5.

Value

A dataframe containing the randomly sampled rows.

Examples

```
# Randomly sample 3 rows from the dataset
randrows(esmdata_sim, n = 3)
```

txt *Custom Text for Rmarkdown with Counting*

Description

This function generates custom text and provides the option to include an optional count that can be integrated into the text. It creates an HTML span element with a unique identifier (ID). This identifier can be used for customizing the appearance of the text using CSS code. Default identifiers are "text_issue", "text_inspect", "text_mod". To use it, simply call the function within inline code, do not call this function within a chunk.

Usage

```
txt(
  id = "text_issue",
  title = "Issue",
  text = "",
  count = TRUE,
  output_file = NULL
)
```

Arguments

`id`	A character string specifying the unique identifier for the HTML span element. Default identifiers are "text_issue", "text_inspect", "text_mod".
`title`	A character string containing the text to display.
`text`	Descriptive text.
`count`	A logical value indicating whether to include a count in the text.
`output_file`	A character string specifying the json file should be created. If not provided, the template will be copied to the current working directory.

Value

A character string containing the HTML span element with the specified ID and text.

Examples

```
if (interactive()) {
  # In RMarkdown, use 'r txt()`.
  # Do not call this function within a chunk

  txt("text_issue", "Issue", text = "Wrong participant number")
  txt("text_mod", "Modification", text = "Change the participant number from 1 to 32",
      count = FALSE)
}
```

use_template *Import esmtools RMarkdown Templates*

Description

The `use_template()` function allows you to copy esmtools RMarkdown templates from the package to your working directory for further customization and usage.

Usage

```
use_template(template_name, output_dir, overwrite = FALSE)
```

Arguments

`template_name`	A character string specifying the name of the template to be copied. Valid options are "preprocess_esm", "advanced_preprocess_esm", and "data_characteristics_report".
`output_dir`	A mandatory character string specifying the directory where the template should be copied.
`overwrite`	Logical indicating whether to overwrite existing files with the same name in the output directory. Default is `FALSE`.

Value

Invisible `NULL`

Examples

```
if (interactive()) {
  # Copy and paste the "preprocess_esm" template to the current working directory
  use_template("preprocess_esm")

  # Copy and paste the "advanced_preprocess_esm" template to a specific directory
  use_template("advanced_preprocess_esm", output_dir = "/path/to/templates")

 # Copy and paste the "data_characteristics_report" template to the current working directory
  # and overwrite existing file (if any).
  use_template("data_characteristics_report", overwrite = TRUE)
}
```

`vars_consist` *Create Unique Values Dataframe*

Description

This function takes a dataframe, a group variable, and a vector of variables. It creates a new dataframe that contains unique values for each variable within each group modality. Only one group variable should be used, otherwise, an error will be thrown. The resulting dataframe has the group variable as the first column and each variable's unique values as subsequent columns. The `merge_val` function is applied to merge the unique values within each group.

Usage

```
vars_consist(.df, group, vars, max_length = 4)
```

Arguments

`.df`	A dataframe containing the data.
`group`	The name of the group variable.
`vars`	A vector of variable names to consider.
`max_length`	The maximum length of the merged values. If the length of the merged values exceeds this value, the values are truncated and an ellipsis is added at the end. Default is 4.

Value

A dataframe with unique values for each variable within each group modality.

Examples

```
vars_consist(esmdata_sim, group = "id", vars = c("age", "cond_dyad"))
```

Index