# Package ‘synr’

July 23, 2025

**Title** Explore and Process Synesthesia Consistency Test Data

**Version** 1.0.0

**Description** Explore synesthesia
consistency test data, calculate consistency scores,
and classify participant data as valid or invalid.

**Depends** R (>= 3.6.0)

**Imports** methods (>= 3.6), data.table (>= 1.12), ggplot2 (>= 3.3.0),
dbscan (>= 1.1)

**Suggests** testthat (>= 2.1.0), dplyr (>= 1.0.0), knitr, rmarkdown,
tidyr, plotly

**License** MIT + file LICENSE

**URL** `https://datalowe.github.io/synr/`, `https://github.com/datalowe/synr`

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Lowe Wilsson [aut, cre]

**Maintainer** Lowe Wilsson <`datalowe@posteo.de`>

**Repository** CRAN

**Date/Publication** 2024-01-13 21:50:02 UTC

## Contents

---

`synr-package` *synr: Explore and Process Synesthesia Consistency Test Data*

---

### Description

synr helps you work with data resulting from grapheme-color consistency tests for synesthesia.

To learn more about synr, start with the vignettes: `browseVignettes(package = "synr")`

### Author(s)

**Maintainer**: Lowe Wilsson `<datalowe@posteo.de>`

### See Also

Useful links:

- `https://datalowe.github.io/synr/`
- `https://github.com/datalowe/synr`

---

`create_grapheme` *Create a grapheme instance*

---

### Description

Takes in a symbol/grapheme and sets of response times/colors, then creates a Grapheme instance that holds the passed information and returns it.

### Usage

```
create_grapheme(
  symbol,
  response_times = NULL,
  response_colors,
  color_space_spec = "Luv"
)
```

## Arguments

| | |
|---|---|
| `symbol` | A one-element character vector holding a symbol/grapheme. |
| `response_times` | (optional) A numeric vector. Times from presentation to response, in order. |
| `response_colors` | A character vector. Response colors, as hex color codes. |
| `color_space_spec` | A one-element character vector. What color space is to be used? The following color spaces are supported: "XYZ", "sRGB", "Apple RGB", "Lab", and "Luv" |

## Examples

```
create_grapheme(symbol="a", response_times=c(2.3, 6.7, 0.4),
response_colors=c("84AE99", "9E3300", "000000"), color_space_spec="Luv")
```

---

`create_participant` *Create a Participant instance.*

---

## Description

Takes in a participant id, set of symbols for which graphemes should be created and participant trial/response data. Returns a Participant instance with all the input data linked to it. For each grapheme, if there are data for less trials than the number specified by n_trials_per_grapheme, NA values are added to affected graphemes' associated vectors of response times/colors.

## Usage

```
create_participant(
  participant_id,
  grapheme_symbols,
  n_trials_per_grapheme,
  trial_symbols,
  response_times = NULL,
  response_colors,
  color_space_spec = "Luv",
  test_date = NULL
)
```

## Arguments

| | |
|---|---|
| `participant_id` | A one-element character vector holding a participant id. |
| `grapheme_symbols` | A character vector of symbols/graphemes for which Grapheme instances should be created and linked to the Participant instance. |
| `n_trials_per_grapheme` | A one-element numeric vector holding the number of trials per grapheme. |

| | |
|---|---|
| `trial_symbols` | A character vector that holds one symbol/grapheme for each trial of the participant's consistency test run. |
| `response_times` | (optional) A numeric vector. Consistency test times from presentation to response, in order. |
| `response_colors` | A character vector. Consistency test response colors, as hex color codes. |
| `color_space_spec` | A one-element character vector. What color space is to be used? The following color spaces are supported: "XYZ", "sRGB", "Apple RGB", "Lab", and "Luv" |
| `test_date` | (optional) A one-element character vector in the format "YYYY-MM-DD" that indicates on what date the participant finished the consistency test. |

## Examples

```
participant_id <- "1"
target_symbols_vec <- c("A", "D", "7")
symbol_vec <- c("A", "D", "7",
                "D", "A", "7",
                "7", "A", "D")
times_vec <- c(1.1, 0.4, 5,
               0.3, 2.4, 7.3,
               1, 10.2, 8.4)
color_vec <- c("98FF22", "138831", "791322",
               "8952FE", "DC8481", "7D89B0",
               "001100", "887755", "FF0033")
p <- create_participant(participant_id=participant_id,
                        grapheme_symbols=target_symbols_vec,
                        n_trials_per_grapheme=3,
                        trial_symbols=symbol_vec,
                        response_times=times_vec,
                        response_colors=color_vec,
                        color_space_spec="Luv")
```

---

`create_participantgroup` *Create a ParticipantGroup instance using long-format data*

---

## Description

Takes in a data frame of raw 'long format' consistency test data and returns a ParticipantGroup instance, to which all the relevant data are linked. See the example data frame ‘synr_exampledf_long_small‘ and its documentation (‘help(synr_exampledf_long_small)‘) for more information on the format that this function expects data to be in.

### Usage

```
create_participantgroup(
  raw_df,
  n_trials_per_grapheme = 3,
  id_col_name,
  symbol_col_name,
  color_col_name,
  time_col_name = NULL,
  color_space_spec = "Luv"
)
```

### Arguments

| | |
|---|---|
| `raw_df` | A data frame of ’long format’ raw consistency test data. |
| `n_trials_per_grapheme` | A one-element numeric vector holding the number of trials per grapheme that was used in the consistency test the data are from. |
| `id_col_name` | A one-element character vector that holds the name of the participant id column in raw_df. |
| `symbol_col_name` | A one-element character vector that holds the name of the grapheme/symbol column in raw_df. |
| `color_col_name` | A one-element character vector that holds the name of the response color (hex codes) column in raw_df. |
| `time_col_name` | (optional) A one-element character vector that holds the name of the response time (time from stimulus presentation to response) column in raw_df. |
| `color_space_spec` | A one-element character vector specifying which color space to use for calculations with participant data. One of "XYZ", "sRGB", "Apple RGB", "Lab", and "Luv". |

### Examples

```
pg <- create_participantgroup(
  raw_df=synr_exampledf_long_small,
  n_trials_per_grapheme=2,
  id_col_name="participant_id",
  symbol_col_name="trial_symbol",
  color_col_name="response_color",
  time_col_name="response_time",
  color_space_spec="Luv"
)
cons_means <- pg$get_mean_consistency_scores()
print(cons_means)
```

`create_participantgroup_widedata`
*Create a ParticipantGroup instance*

## Description

Takes in a data frame of raw consistency test data and returns a ParticipantGroup instance, to which all the relevant data are linked. See the example data frame synr_exampledf_wide_small and its documentation (help(synr_exampledf_wide_small)) for information on the format that this function expects data to be in.

Participant id and (optional) test date column names are specified with the exact column names used in the data frame passed to the function. Symbol (i. e. grapheme), response color and response time (optional) columns are specified using regular expressions. You can read about regular expressions using R here if you want to, but basically what you want to do is this: say your columns with response colors are named "chosen_color_001", "chosen_color_002" and so on. You then simply set color_col_regex="chosen_color" when calling this function. The important thing is that you specify a part of the column names that is unique for the type of column you want to indicate. So if your symbol/grapheme columns are named "grapheme_1", "grapheme_2" ... and your participant id column is named "graphparticipant_1" ..., then symbol_col_regex="graph" wouldn't work, but symbol_col_regex="grapheme_" or even symbol_col_regex="graphe" would.

## Usage

```
create_participantgroup_widedata(
  raw_df,
  n_trials_per_grapheme = 3,
  participant_col_name,
  symbol_col_regex,
  color_col_regex = "colou*r",
  time_col_regex = NULL,
  testdate_col_name = NULL,
  color_space_spec = "Luv"
)
```

## Arguments

| | |
|---|---|
| `raw_df` | A data frame of raw consistency test data. |
| `n_trials_per_grapheme` | A one-element numeric vector holding the number of trials per grapheme that was used in the consistency test the data are from. |
| `participant_col_name` | A one-element character vector that holds the column name used for the column in raw_df that holds participant id's. (e. g. "participant_id" for the synr::synr_exampledf_wide_small) |
| `symbol_col_regex` | A one-element character vector with a regular expression (see above) unique to columns in the passed data frame that hold trial graphemes/symbols. |

`color_col_regex`
: A one-element character vector with a regular expression (see above) unique to columns in the passed data frame that hold response color hex codes.

`time_col_regex`
: (optional) A one-element character vector with a regular expression (see above) unique to columns in the passed data frame that hold response times (times from stimulus presentation to response).

`testdate_col_name`
: (optional) A one-element character vector that holds the column name used for the column in raw_df that holds test dates (dates when participants finished the consistency test).

`color_space_spec`
: A one-element character vector. What color space is to be used for analyses of the data? The following color spaces are supported: "XYZ", "sRGB", "Apple RGB", "Lab", and "Luv"

## Examples

```
pg <- create_participantgroup_widedata(raw_df=synr_exampledf_wide_small,
                                n_trials_per_grapheme=2,
                                participant_col_name="participant_id",
                                symbol_col_regex="symbol",
                                color_col_regex="colou*r",
                                time_col_regex="response_time",
                                color_space_spec="Luv"
)
cons_means <- pg$get_mean_consistency_scores()
print(cons_means)
```

---

`Grapheme-class` *A Reference Class for representing consistency test graphemes*

---

## Description

A Reference Class for representing consistency test graphemes

## Fields

`symbol` A one-element character vector containing the symbol/set of symbols that describe(s) the grapheme, e. g. '7' or 'Monday'. Set at class new() call or using set_symbol method.

`response_colors` A matrix where each row specifies color coordinates for each participant response. Set using set_colors method.

`response_times` A numeric vector of response times. Set using set_times method.

`color_space` A one-element character vector which describes the color space that response colors are coded in. Set when using set_colors method.

## Methods

`get_abbreviated_symbol()` Return a short (3 character) representation of the grapheme's symbol.

`get_consistency_score(na.rm = FALSE, method = "euclidean")` Calculate the consistency score based on the Grapheme instance's response colors. Throws an error if no responses have been registered yet. Always returns NA if all grapheme responses are NA. If na.rm=FALSE, returns NA if any grapheme response is NA. If na.rm=TRUE, returns the consistency score for non-NA responses. This function relies on the base/stats function dist() and so supports only distance calculation methods implemented by dist() (use help(dist) to learn more about it).

`get_mean_color(na.rm = FALSE)` Average all registered response colors and return the result (using the color space set at grapheme initialization) as a 3-element vector. Example: if color space is RGB, element 1 represents mean R value, element 2 mean G value, element 3 B value.

If na.rm=FALSE and any of the response colors is missing, return a 3-element NA vector. If na.rm=TRUE, return a 3-element NA vector if all response colors are missing, otherwise return mean of all available colors.

`get_mean_response_time(na.rm = FALSE)` Get the mean of the grapheme's associated response times.

`get_num_non_na_colors()` Get the number of response colors that are non-NA, returned as a one-element numeric vector.

`get_plot_data_list()` Get a list of the grapheme's data, bundled up in a format ready for use in Participant.get_plot_data() method as a row of plot data.

`has_only_non_na_colors()` Returns TRUE if the grapheme only has responses with valid colors, FALSE if there are responses with nonvalid colors or there are no responses at all.

`set_colors(hex_codes, color_space_spec)` Set response colors, using passed RGB hex codes. Converts the hex codes to color coordinates in the specified color space. Supports the following color spaces: "XYZ", "sRGB", "Apple RGB", "Lab", and "Luv". For all NA values passed, a row of NA values will be included in the matrix (preserving order of responses). Returned/set response colors are in the format of a matrix where each row represents one response/color, and each column represents one color coordinate axis (there are always 3 axes used for the currently supported color spaces)

`set_symbol(symbol_chars)` Set the grapheme's symbol attribute, using a passed one-element character vector.

`set_times(times)` Add response times, using passed numeric vector.

## Examples

```
a <- synr::Grapheme$new(symbol='a')
a$set_colors(c("#101010", NA), "Luv")
a$set_times(c(5, 10))
a$get_num_non_na_colors()
```

## `Participant-class` *A Reference Class for representing consistency test participants*

### Description

A Reference Class for representing consistency test participants

### Fields

`id` A one-element character vector containing the participant's ID. Set at class new() call.

`test_date` A one-element Date vector which specifies the date on which the participant did the consistency test.

`graphemes` A list of `Grapheme` class instances.

### Methods

`add_grapheme(grapheme)` Add a passed grapheme to the participant's list of graphemes. The grapheme's entry in the list is named based on the grapheme's symbol. Note that if you try to add a grapheme with a symbol that's identical to one of the graphemes already in the participant's list of graphemes, the already existing same-symbol grapheme is overwritten.

`add_graphemes(grapheme_list)` Go through a passed list of Grapheme instances and add each one using the add_grapheme() method.

`check_valid_get_twcv( min_complete_graphemes = 5, dbscan_eps = 20, dbscan_min_pts = 4, max_var_tight_clu`
Checks if this participant's data are valid based on passed arguments. This method aims to identify participants who had too few responses or varied their response colors too little, by marking them as invalid. Note that there are no absolutely correct values, as what is 'too little variation' is highly subjective. You might need to tweak parameters to be in line with your project's criteria, especially if you use another color space than CIELUV, since the default values are based on what seems to make sense in a CIELUV context. If you use the results in a research article, make sure to reference synr and specify what parameter values you passed to the function.

This method relies heavily on the DBSCAN algorithm and the package 'dbscan', and involves calculating a synr-specific 'Total Within-Cluster Variance' (TWCV) score. You can find more information, and what the parameters here mean, in the documentation for the function `validate_get_twcv`.

**Parameters:**

- `min_complete_graphemes` The minimum number of graphemes with complete (all non-NA color) responses that the participant data must have for them to not be categorized as invalid based on this criterion. Defaults to 5.
- `dbscan_eps` Radius of 'epsilon neighborhood' when applying DBSCAN clustering. Defaults to 20.
- `dbscan_min_pts` Minimum number of points required in the epsilon neighborhood for core points (including the core point itself). Defaults to 4.
- `max_var_tight_cluster` Maximum variance for an identified DBSCAN cluster to be considered 'tight-knit'. Defaults to 150.

- `max_prop_single_tight_cluster` Maximum proportion of points allowed to be within a single 'tight-knit' cluster (exceeding this leads to classification as invalid). Defaults to 0.6.
- `safe_num_clusters` Minimum number of identified DBSCAN clusters (including 'noise' cluster only if it consists of at least 'dbscan_min_pts' points) that guarantees validity if points are 'non-tight-knit'. Defaults to 3.
- `safe_twcv` Minimum total within-cluster variance (TWCV) score that guarantees validity if points are 'non-tight-knit'. Defaults to 250.
- `complete_graphemes_only` A logical vector. If TRUE, only data from graphemes that have all non-NA color responses are used; if FALSE, even data from graphemes with some NA color responses are used. Defaults to TRUE.
- `symbol_filter` A character vector (or NULL) that specifies which graphemes' data to use. Defaults to NULL, meaning data from all of the participant's graphemes will be used.

**Returns:** A list with components

- `valid` TRUE if categorized as valid, otherwise FALSE.
- `reason_invalid` One-element character vector describing why participant's data were deemed invalid, or empty string if valid is TRUE.
- `twcv` One-element numeric (or NA if there are no/too few graphemes with complete responses) vector indicating participant's calculated TWCV.
- `num_clusters` One-element numeric (or NA if there are no/too few graphemes with complete responses) vector indicating the number of identified clusters counting toward the tally compared with 'safe_num_clusters'.

`get_all_colored_symbols(symbol_filter = NULL)` Returns a character vector of symbols corresponding to graphemes for which all responses have an associated non-NA color. If a character vector is passed to symbol_filter, only symbols in the passed vector are returned.

`get_consistency_scores( method = "euclidean", symbol_filter = NULL, na.rm = FALSE )` Returns a list of grapheme symbols with associated consistency scores. If na.rm = TRUE, for each grapheme a consistency score calculation is forced (except if ALL response colors associated with the grapheme are NA). That probably isn't what you want, because it leads to things like a perfect consistency score if all except one response color are NA. Defaults to na.rm = FALSE.

If a character vector is passed to symbol_filter, only consistency scores for graphemes with symbols in the passed vector are returned.

Use the method argument to specify what kind of color space distances should be used when calculating consistency score (usually 'manhattan' or 'euclidean' - see documentation for the base R dist function for all options)

`get_grapheme_mean_colors(symbol_filter = NULL, na.rm = FALSE)` Returns a list of grapheme symbols with associated mean colors, using the color space set at participant creation. Colors are represented by 3-element vectors.

Example: if color space is RGB, vector element 1 represents grapheme mean R value, element 2 mean G value, element 3 B value.

If na.rm = TRUE, for each grapheme a mean color is calculated even if one its associated response colors is missing. Defaults to na.rm = FALSE.

If a character vector is passed to symbol_filter, only mean colors for graphemes with symbols in the passed vector are returned.

`get_mean_consistency_score( symbol_filter = NULL, method = "euclidean", na.rm = FALSE )`
: Returns the mean consistency score with respect to Grapheme instances associated with the participant.

  If na.rm = FALSE, calculates the mean consistency score if all of the participants' graphemes only have response colors that are non-NA, otherwise returns NA. If na.rm = TRUE, returns the mean consistency score for all of the participant's graphemes that only have non-NA response colors, while ignoring graphemes that have at least one NA response color value. Note that NA is returned in either case, if ALL of the participants' graphemes have at least one NA response color value.

  If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating the mean score.

  Use the method argument to specify what kind of color space distances should be used when calculating consistency score (usually 'manhattan' or 'euclidean' - see documentation for the base R dist function for all options)

`get_mean_response_time(symbol_filter = NULL, na.rm = FALSE)` Returns the mean response time, with respect to all Grapheme instances associated with the participant. Weights response times based on number of valid responses that each grapheme has. If na.rm = TRUE, returns mean response time even if there are missing response times. If na.rm = FALSE, returns mean response time if there is at least one response time value for at least one of the participants' graphemes. If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating the mean response time.

`get_nonna_color_resp_mat(symbol_filter = NULL)` Returns an n-by-3 matrix of all non-NA color responses' data, where each column represents a color axis and each row a response color. If a character vector is passed to symbol_filter, only data from responses associated with graphemes with corresponding symbols are included.

`get_number_all_colored_graphemes(symbol_filter = NULL)` Returns the number of graphemes for which all responses have an associated non-NA color. If a character vector is passed to symbol_filter, only graphemes with symbols in the passed vector are counted.

`get_participant_mean_color(symbol_filter = NULL, na.rm = FALSE)` Returns average of all of participants' registered response colors (based on the color space set at participant initialization) as a 3-element vector. Example: if color space is RGB, element 1 represents mean R value, element 2 mean G value, element 3 B value.

  If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating the mean color.

  If na.rm = FALSE, calculates the mean response color if all of the participants' graphemes only have response colors that are non-NA, otherwise returns NA. If na.rm = TRUE, returns the mean response color based on all non-NA response colors.

`get_plot( cutoff_line = FALSE, mean_line = FALSE, grapheme_size = 2, grapheme_angle = 0, grapheme_spacing`
: Returns a ggplot2 plot that describes this participant's grapheme color responses and per-grapheme consistency scores.

  If cutoff_line = TRUE, the plot will include a blue line that indicates the value 135.30, which is the synesthesia cut-off score recommended by Rothen, Seth, Witzel & Ward (2013) for the L*u*v color space. If mean_line = TRUE, the plot will include a green line that indicates the participant's mean consistency score for graphemes with all-valid response colors (if the participant has any such graphemes). If a vector is passed to symbol_filter, this green line represents the mean score for ONLY the symbols included in the filter.

Pass a value to grapheme_size to adjust the size of graphemes shown at the bottom of the plot, e. g. increasing the size if there's a lot of empty space otherwise, or decreasing the size if the graphemes don't fit. The grapheme_angle argument allows rotating graphemes. grapheme_spacing is for adjusting how far grapheme symbols are spaced from each other.

If a character vector is passed to symbol_filter, only data for graphemes with symbols in the passed vector are used.

Graphemes are sorted left-to-right by 1. length and 2. unicode value (this means among other things that digits come before letters).

`get_plot_data(symbol_filter = NULL)` Returns a data frame with the following columns:

1. grapheme (grapheme names - of type character)

2. consistency_score (of type numeric)

3... color_resp<x>, where x is a digit: hold response hex color codes (number of columns depends on number of response colors associated with each grapheme).

The data frame is intended to be used for plotting participant data, using .get_plot(). The call will end with an error if not all of the participant's graphemes have the same number of color responses. This is intended.

If a character vector is passed to symbol_filter, only data for graphemes with symbols in the passed vector are used.

`get_symbols()` Returns a character vector with all symbols for graphemes associated with the participant.

`has_graphemes()` Returns TRUE if there is at least one grapheme in the participant's graphemes list, otherwise returns FALSE

`save_plot( save_dir = NULL, file_format = "png", dpi = 300, cutoff_line = FALSE, mean_line = FALSE, graphem`

Saves a ggplot2 plot that describes this participant's grapheme color responses and per-grapheme consistency scores, using the ggsave function.

If a character vector is passed to symbol_filter, only data for graphemes with symbols in the passed vector are used.

If save_dir is not specified, the plot is saved to the current working directory. Otherwise, the plot is saved to the specified directory. The file is saved using the specified file_format, e. g. JPG (see ggplot2::ggsave documentation for list of supported formats), and the resolution specified with the dpi argument.

If cutoff_line = TRUE, the plot will include a blue line that indicates the value 135.30, which is the synesthesia cut-off score recommended by Rothen, Seth, Witzel & Ward (2013) for the L*u*v color space. If mean_line = TRUE, the plot will include a green line that indicates the participant's mean consistency score for graphemes with all-valid response colors (if the participant has any such graphemes). If a vector is passed to symbol_filter, this green line represents the mean score for ONLY the symbols included in the filter.

Pass a value to grapheme_size to adjust the size of graphemes shown at the bottom of the plot, e. g. increasing the size if there's empty space otherwise, or decreasing the size if the graphemes don't fit. Similarly, you can use the grapheme_angle argument to rotate the graphemes, which might help them fit better.

Apart from these, all other arguments that ggsave accepts (e. g. 'scale') also work with this function, since all arguments are passed on to ggsave.

`set_date(in_date)` Takes in a one-element character vector with a date in the format 'YYYY-MM-DD' and sets the participant's test_date to the specified date.

## `ParticipantGroup-class` *A Reference Class for representing a group of consistency test participants*

### Description

A Reference Class for representing a group of consistency test participants

### Fields

`participants` A list of `Participant` class instances.

### Methods

`add_participant(participant)` Add a passed participant to the participantgroup's list of participants. The participant's entry in the list is named based on the participant's id. Note that if you try to add a participant with an id that's identical to one of the participants already in the participantgroup's list of participants, the already existing same-id participant is overwritten.

`add_participants(participant_list)` Go through a passed list of Participant instances and add each one using the add_participant() method.

`check_valid_get_twcv_scores( min_complete_graphemes = 5, dbscan_eps = 20, dbscan_min_pts = 4, max_var_ti`
Checks if participants' data are valid based on passed arguments. This method aims to identify participants who had too few responses or varied their response colors too little, by marking them as invalid. Note that there are no absolutely correct values, as what is 'too little variation' is highly subjective. You might need to tweak parameters to be in line with your project's criteria, especially if you use another color space than CIELUV, since the default values are based on what seems to make sense in a CIELUV context. If you use the results in a research article, make sure to reference synr and specify what parameter values you passed to the function.

This method relies heavily on the DBSCAN algorithm and the package 'dbscan', and involves calculating a synr-specific 'Total Within-Cluster Variance' (TWCV) score. You can find more information, and what the parameters here mean, in the documentation for the function `validate_get_twcv`. Note that DBSCAN clustering and related calculations are performed on a per-participant basis, before they are summarized in the data frame returned by this method.

**Parameters:**

- `min_complete_graphemes` The minimum number of graphemes with complete (all non-NA color) responses that a participant's data must have for them to not be categorized as invalid based on this criterion. Defaults to 7.
- `dbscan_eps` Radius of 'epsilon neighborhood' when applying (on a per-participant basis) DBSCAN clustering. Defaults to 30.
- `dbscan_min_pts` Minimum number of points required in the epsilon neighborhood for core points (including the core point itself). Defaults to 4.
- `max_var_tight_cluster` Maximum variance for an identified DBSCAN cluster to be considered 'tight-knit'. Defaults to 150.

- `max_prop_single_tight_cluster` Maximum proportion of points allowed to be within a single 'tight-knit' cluster (if a participant's data exceed this limit, they are classified as invalid). Defaults to 0.6.
- `safe_num_clusters` Minimum number of identified DBSCAN clusters (including 'noise' cluster only if it consists of at least 'dbscan_min_pts' points) that guarantees validity of a participant's data if points are 'non-tight-knit'. Defaults to 3.
- `safe_twcv` Minimum total within-cluster variance (TWCV) score that guarantees a participant's data's validity if points are 'non-tight-knit'. Defaults to 250.
- `complete_graphemes_only` A logical vector. If TRUE, only data from graphemes that have all non-NA color responses are used; if FALSE, even data from graphemes with some NA color responses are used. Defaults to TRUE.
- `symbol_filter` A character vector (or NULL) that specifies which graphemes' data to use. Defaults to NULL, meaning data from all of the participants' graphemes will be used.

**Returns:** A data frame with columns

- `valid` Holds TRUE for participants whose data were classified as valid, FALSE for participants whose data were classified as invalid.
- `reason_invalid` Strings which describe for each participant why their data were deemed invalid. Participants whose data were classified as valid have empty strings here.
- `twcv` Numeric column which holds participants' calculated TWCV scores (NA for participants who had no/too few graphemes with complete responses).
- `num_clusters` One-element numeric (or NA if there are no/too few graphemes with complete responses) vector indicating the number of identified clusters counting toward the tally compared with 'safe_num_clusters'.

`get_ids()` Returns a character vector with all ids for participants associated with the participant-group.

`get_mean_colors(symbol_filter = NULL, na.rm = FALSE)` Returns an nx3 data frame of mean colors for participants in the group, where the columns represent chosen color space axis 1, 2, and 3, respectively (e.g. 'R', 'G', 'B' if 'sRGB' was specified upon participantgroup creation).

If na.rm=FALSE, for each participant calculates the mean color if all of the participants' graphemes only have response colors that are non-NA, otherwise puts NA values for that participant's row in matrix. If na.rm=TRUE, for each participant calculates the mean color for all of the participant's valid response colors, while ignoring NA response colors. Note that for participants whose graphemes ALL have at least one NA response color value, an NA is put in the row corresponding to that participant, regardless of what na.rm is set to.

If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating each participant's mean color.

`get_mean_consistency_scores( method = "euclidean", symbol_filter = NULL, na.rm = FALSE )` Returns a vector of mean consistency scores for participants in the group. If na.rm=FALSE, for each participant calculates the mean consistency score if all of the participants' graphemes only have response colors that are non-NA, otherwise puts an NA value for that participant in returned vector. If na.rm=TRUE, for each participant calculates the mean consistency score for all of the participant's graphemes that only have non-NA response colors, while ignoring graphemes that have at least one NA response color value. Note that for participants whose graphemes ALL have at least one NA response color value, an NA is put in the returned vector for that participant, regardless of what na.rm is set to.

If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating each participant's mean score.

Use the method argument to specify what kind of color space distances should be used when calculating consistency scores (usually 'manhattan' or 'euclidean' - see documentation for the base R dist function for all options)

`get_mean_response_times(symbol_filter = NULL, na.rm = FALSE)` Returns the mean response times, with respect to Grapheme instances associated with each participant. If na.rm=TRUE, for each participant returns mean response time even if there are missing response times. If na.rm=FALSE, returns mean response time if there is at least one response time value for at least one of the participants' graphemes. If a character vector is passed to symbol_filter, only data from graphemes with symbols in the passed vector are used when calculating each participant's mean response time.

`get_numbers_all_colored_graphemes(symbol_filter = NULL)` Returns a vector with numbers representing how many graphemes with all-valid (non-na) response colors that each participant has. If a character vector is passed to symbol_filter, only data connected to graphemes with symbols in the passed vector are used.

`has_participants()` Returns TRUE if there is at least one participant in the participantgroup's participants list, otherwise returns FALSE

`save_plots( save_dir = NULL, file_format = "png", dpi = 300, cutoff_line = FALSE, mean_line = FALSE, graphe`
Goes through all participants and for each one produces and saves a ggplot2 plot that describes the participant's grapheme color responses and per-grapheme consistency scores, using the ggsave function.

If a character vector is passed to symbol_filter, only data for graphemes with symbols in the passed vector are used.

If path is not specified, plots are saved to the current working directory. Otherwise, plots are saved to the specified directory. The file is saved using the specified file_format, e. g. JPG (see ggplot2::ggsave documentation for list of supported formats), and the resolution specified with the dpi argument.

If cutoff_line=TRUE, each plot will include a blue line that indicates the value 135.30, which is the synesthesia cut-off score recommended by Rothen, Seth, Witzel & Ward (2013) for the L*u*v color space. If mean_line=TRUE, the plot will include a green line that indicates the participant's mean consistency score for graphemes with all-valid response colors (if the participant has any such graphemes). If a vector is passed to symbol_filter, this green line represents the mean score for ONLY the symbols included in the filter.

Pass a value to grapheme_size to adjust the size of graphemes shown at the bottom of the plot, e. g. increasing the size if there's empty space otherwise, or decreasing the size if the graphemes don't fit. Similarly, you can use the grapheme_angle argument to rotate the graphemes, which might help them fit better.

Apart from the ones above, all other arguments that ggsave accepts (e. g. 'scale') also work with this function, since all arguments are passed on to ggsave.

## `point_3d_variance` *Calculate sample variance of 3D point distance from centroid*

## Description

Calculates sample variance of points' distances in 3D space from their centroid. This function is normally only used indirectly through 'validate_get_twcv'.

## Usage

```
point_3d_variance(point_matrix)
```

## Arguments

| | |
|---|---|
| `point_matrix` | An n-by-3 numerical matrix where each row corresponds to a single point in 3D space. |

## Value

A one-element numeric vector holding calculated variance

## Details

The variance here is taken to mean the sum of variances for each dimension/axis:

$$\frac{\sum_{i=1}^{n}(x_i - x_m)^2 + (y_i - y_m)^2 + (z - z_m)^2}{n-1}$$

Where $X/Y/Z$ represent one axis each, $a_m$ represents the mean of all points' coordinates on an axis, and $n$ represents the total number of points.

## See Also

`centroid_3d_sq_dist`

---

`synr_exampledf_large` *Raw consistency test data example, long format*

---

## Description

A data frame with an example of raw consistency test data that are compatible with the synr package's 'create_participantgroup' function. The color and 'symbol' data are from five actual participants who did a test that included all letters, digits and weekdays, with 3 trials per grapheme. The response times are randomly generated. Note that response times are optional. If you don't have them, you can still use synr - see 'help(create_participantgroup_widedata)'.

## Usage

```
synr_exampledf_large
```

### Format

A data frame with 516 rows and 4 columns:

**participant_id** Participant ID

**trial_symbol** Column of trial symbols/graphemes

**response_color** Column of trial response colors

**response_time** Column of trial response times

## `synr_exampledf_long_small` *Raw consistency test data example, long format (small)*

### Description

A data frame with an example of raw consistency test data that are compatible with the synr package's 'create_participantgroup' function, with completely made updata for three participants from a hypothetical test that included three graphemes ("A", "D", 7) and two responses per grapheme. More graphemes and/or responses per grapheme can be handled by the package (though participant plots do not function correctly if there are more than three responses per grapheme). Note that response times are optional. If you don't have them, you can still use synr - see 'help(create_participantgroup_widedata)'.

### Usage

```
synr_exampledf_long_small
```

### Format

A data frame with 18 rows and 4 columns:

**participant_id** Participant ID

**trial_symbol** Column of trial symbols/graphemes

**response_color** Column of trial response colors

**response_time** Column of trial response times

## `synr_exampledf_wide_small` *Raw consistency test data example, wide format (small)*

### Description

A data frame with an example of raw consistency test data that are compatible with the synr package's 'create_participantgroup_widedata' function, with data for three participants from a test that included three graphemes ("A", "D", 7) and two responses per grapheme. More graphemes and/or responses per grapheme can be handled by the package (though participant plots do not function correctly if there are more than three responses per grapheme)

### Usage

```
synr_exampledf_wide_small
```

### Format

A data frame with 3 rows and 8 columns:

**participant_id** Participant ID

**symbol_1** Column with symbol/grapheme connected to first response

**response_color_1** Column with color of first response

**response_time_1** (optional) Column with time from presentation to response, for first response

**symbol_2** Column with symbol/grapheme connected to second response

**response_color_2** Column with color of second response

**response_time_2** (optional) Column with time from presentation to response, for second response

**symbol_3** Column with symbol/grapheme connected to third response

**response_color_3** Column with color of third response

**response_time_3** (optional) Column with time from presentation to response, for third response

**symbol_4** Column with symbol/grapheme connected to fourth response

**response_color_4** Column with color of fourth response

**response_time_4** (optional) Column with time from presentation to response, for fourth response

**symbol_5** Column with symbol/grapheme connected to fifth response

**response_color_5** Column with color of fifth response

**response_time_5** (optional) Column with time from presentation to response, for fifth response

**symbol_6** Column with symbol/grapheme connected to sixth response

**response_color_6** Column with color of sixth response

**response_time_6** (optional) Column with time from presentation to response, for sixth response

`total_within_cluster_variance`
*Calculate Total Within Cluster Variance of 3D points*

## Description

Calculates *Total Within Cluster Variance(TWCV)* of 3D points. This function is normally only used indirectly through ‘validate_get_twcv‘.

## Usage

```
total_within_cluster_variance(point_matrix, cluster_vector)
```

## Arguments

| | |
|---|---|
| `point_matrix` | An n-by-3 numerical matrix where each row corresponds to a single point in 3D space. |
| `cluster_vector` | A numerical vector of cluster assignments, of length n (ie one assignment per point). |

## Value

A one-element numeric vector holding calculated variance

## TWCV

TWCV is a synr-specific term for a measure that aims to describe spread of points in 3D space while taking into account that points belong to distinct clusters. TWCV is calculated in a multi-step process:

1. Each cluster’s centroid is calculated.
2. All points’ squared distances to their corresponding centroids are calculated.
3. The point-to-centroid squared distances are summed up.
4. The sum of squared distances is divided by the total number of points, minus the number of clusters (to account for decreased degrees of freedom).

## See Also

`centroid_3d_sq_dist`

## validate_get_twcv *Check if color data are valid and get TWCV*

### Description

Checks if passed color data are valid, i. e. are bountiful and varied enough according to passed validation criteria. This function is normally only used indirectly through ‘Participant$check_valid_get_twcv()‘ or ‘ParticipantGroup$get_valid_twcv()‘.

### Usage

```
validate_get_twcv(
  color_matrix,
  dbscan_eps = 20,
  dbscan_min_pts = 4,
  max_var_tight_cluster = 150,
  max_prop_single_tight_cluster = 0.6,
  safe_num_clusters = 3,
  safe_twcv = 250
)
```

### Arguments

| | |
|---|---|
| `color_matrix` | An n-by-3 numerical matrix where each row corresponds to a single point in 3D color space. |
| `dbscan_eps` | One-element numerical vector: radius of ‘epsilon neighborhood’ when applying DBSCAN clustering. |
| `dbscan_min_pts` | One-element numerical vector: Minimum number of points required in the epsilon neighborhood for core points (including the core point itself). |
| `max_var_tight_cluster` | One-element numerical vector: maximum variance for a cluster to be considered ’tight-knit’. |
| `max_prop_single_tight_cluster` | One-element numerical vector: maximum proportion of points allowed to be within a ’tight-knit’ cluster (if this threshold is exceeded, the data are categorized as invalid). |
| `safe_num_clusters` | One-element numerical vector: minimum number of clusters that guarantees validity if points are ’non-tight-knit’. |
| `safe_twcv` | One-element numerical vector: minimum total within-cluster variance (TWCV) score that guarantees validity if points are ’non-tight-knit’. |

## Value

A list with components

| | |
|---|---|
| `valid` | One-element logical vector |
| `reason_invalid` | One-element character vector, empty if valid is TRUE |
| `twcv` | One-element numeric (or NA if can't be calculated) vector, indicating TWCV |
| `num_clusters` | One-element numeric (or NA if can't be calculated) vector, indicating the number of identified clusters counting toward the tally compared with 'safe_num_clusters' |

## Details

This function relies heavily on the DBSCAN algorithm and its implementation in the R package ‘dbscan‘, for clustering color points. For further information regarding the 'dbscan_eps' and 'dbscan_min_pts' parameters as well as DBSCAN itself, please see the ‘dbscan‘ documentation. Once clustering is done, passed validation criteria are applied:

- If too high a proportion of all color points (cut-off specified with ‘max_prop_single_tight_cluster‘) fall within a single 'tight-knit' cluster (with a cluster variance less than or equal to ‘max_var_tight_cluster‘), then the data are always classified as invalid.
- If the first criterion is cleared, *and* points form more than ‘safe_num_cluster‘ clusters, data are always classified as valid.
- If the first criterion is cleared, *and* the Total Within-Cluster Variance (TWCV) score is greater than or equal to ‘safe_twcv‘, data are always classified as valid.

Note that this means data can be classified as valid by either having at least 'safe_num_cluster' clusters, *or* by having points composing a smaller number of clusters but spaced relatively far apart *within* these clusters.

The DBSCAN 'noise' cluster only counts towards the 'cluster tally' (compared with 'safe_num_cluster') if it includes at least 'dbscan_min_pts' points. Points in the noise cluster are however always included in other calculations, e. g. total within-cluster variance (TWCV).

## See Also

`point_3d_variance` for single-cluster variance, `total_within_cluster_variance` for TWCV.

# Index