Package 'implicitMeasures'

October 13, 2022

Type Package

Title Compute Scores for Different Implicit Measures

Version 0.2.1

Author Ottavia M. Epifania [aut, cre],
Pasquale Anselmi [ctb],
Egidio Robusto [ctb]

Maintainer Ottavia M. Epifania <otta.epifania@gmail.com>

Description A tool for computing the scores for the Implicit Association Test (IAT; Greenwald, McGhee & Schwartz (1998) <doi:10.1037/0022-3514.74.6.1464>) and the Single Category-IAT (SC-IAT: Karpinski & Steinman (2006) <doi:10.1037/0022-3514.91.1.16>). Functions for preparing the data (both for the IAT and the SC-IAT), plotting the results, and obtaining a table with the scores of implicit measures descriptive statistics are provided.

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports ggplot2, stringr, tidyr, xtable

Suggests testthat (>= 2.1.0), knitr, rmarkdown, tableHTML, data.table, spelling

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-16 13:40:13 UTC

R topics documented:

`clean_iat` *Prepare and clean IAT data.*

Description

Select IAT blocks for the *D-score* computation and eventually save demographic data.

Usage

```
clean_iat(
  data,
  sbj_id = "participant",
  block_id = "blockcode",
  mapA_practice = "practice_MappingA",
  mapA_test = "test_MappingA",
  mapB_practice = "practice_MappingB",
  mapB_test = "test_MappingB",
  latency_id = "latency",
  accuracy_id = "correct",
  trial_id = NULL,
  trial_eliminate = NULL,
  demo_id = NULL,
  trial_demo = NULL
)
```

Arguments

`data`	Dataframe containing IAT data.
`sbj_id`	Column identifying participants' IDs. This variable can be a `character`, `numeric`, or `factor`.
`block_id`	String. Column identifying IAT blocks. The `block_id` variable should be a `factor` with each level identifying an IAT block.
`mapA_practice`	String. Label for the practice blocks of Mapping A (as it appears in the `block_id` variable).
`mapA_test`	String. Label for the test blocks of Mapping A (as it appears in the `block_id` variable).
`mapB_practice`	String. Label for the practice blocks of Mapping B (as it appears in the `block_id` variable).
`mapB_test`	String. Label for the test blocks of Mapping B (as it appears in the `block_id` variable).
`latency_id`	String. Column identifying response times (in millisecond). If the IAT had a built-in correction, latencies of the incorrect responses should be those inflated with the built-in correction.
`accuracy_id`	String. Column identifying the IAT accuracy responses. The `accuracy_id` variable should be a numeric variable identifying the correct responses (with 1) and the incorrect responses (with 0).
`trial_id`	Character. Column identifying the trials. Specify this only if you want to delete some specific trials.
`trial_eliminate`	Character or character vector. Label(s) identifying the trials in `trial_id` to eliminate.
`demo_id`	Character. Column identifying demographic blocks. It can be the same as `block_id`.
`trial_demo`	Character or character vector identifying the name of the blocks in `demo_id` containing the demographic information.

Value

List of dataframe.

`data_keep` Dataframe with class `iat_clean`. The dataframe contains the data of the blocks specified in `mapA_practice`, `mapA_test`, `mapB_practice`, `mapB_test`. If you have specified the trials to eliminate through `trial_eliminate`, `data_keep` will contain the already cleaned dataset. This dataset should be passed to the `computeD` function.

`data_eliminate` Dataframe containing all the discarded blocks and trials.

`data_demo` Dataframe containing demographic variables. It will be present only if you specified the `demo_id` and `trial_demo` arguments.

Examples

```
data("raw_data") # load data
iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                           block_id = "blockcode",
                           mapA_practice = "practice.iat.Milkbad",
                           mapA_test = "test.iat.Milkbad",
                           mapB_practice = "practice.iat.Milkgood",
                           mapB_test = "test.iat.Milkgood",
                           latency_id = "latency",
                           accuracy_id = "correct",
                           trial_id = "trialcode",
                           trial_eliminate = c("reminder", "reminder1"),
                           demo_id = "blockcode",
                           trial_demo = "demo")
iat_data <- iat_cleandata[[1]] # select the first element of the list (IAT data)
head(iat_data)
demo_data <- iat_cleandata[[3]] # select the third element of the list
                             # (demographic data)
head(demo_data)
```

---

clean_sciat *Prepare and clean SC-IAT data*

Description

Select the SC-IAT blocks, for either one or two SC-IATs. Eventually save demographic data.

Usage

```
clean_sciat(
  data,
  sbj_id = "participant",
  block_id = "blockcode",
  accuracy_id = "correct",
  latency_id = "latency",
  block_sciat_1 = NULL,
  block_sciat_2 = NULL,
  trial_id = NULL,
  trial_eliminate = NULL,
  demo_id = NULL,
  trial_demo = NULL
)
```

Arguments

`data`	Dataframe containing SC-IAT data.
`sbj_id`	Column identifying participants' IDs. This variable can be a `character`, `numeric`, or `factor`.

`block_id` String. Column identifying SC-IAT blocks. The `block_id` variable should be a `factor` with each level identifying a SC-IAT block.

`accuracy_id` String. Column identifying the IAT accuracy responses. The `accuracy_id` variable should be a numeric variable identifying the correct responses (with 1) and the incorrect responses (with 0).

`latency_id` String. Column identifying response times (in millisecond).

`block_sciat_1` Character or character vector. Labels identifying the first SC-IAT blocks as they are named in the `block_id`.

`block_sciat_2` Character or character vector. Labels identifying the second (if present) SC-IAT blocks as they are named in the `block_id`.

`trial_id` Character. Column identifying the trials. Specify this only if you want to delete some specific trials. If a response window was used for the SC-IAT administration the label of the non-response must be included in this variable.

`trial_eliminate` Character or character vector. Labels of the trials to eliminate in the `trial_id` to eliminate (NOTE: don't use this command to delete the responses exceeding the response time window).

`demo_id` Character. Character. Column identifying demographic blocks. It can be the same as `block_id`.

`trial_demo` Character or character vector identifying the name of the blocks in `demo_id` containing the demographic information.

Value

List of dataframe.

`sciat1` Data frame with class `sciat_clean` containing the data of the first SC-IAT as specified `block_sciat_1`. If any labels was specified in `trial_eliminate`, `data_keep` will contain the already cleaned dataset.

`sciat2` Data frame with class `sciat_clean` containing the data of the second (if any) SC-IAT as specified through `block_sciat_2`. If any labels was specified in `trial_eliminate`, `data_keep` will contain the already cleaned dataset.

`data_demo` Data frame. Present only when `variable_demo` and `trial_demo` arguments are specified.

Examples

```
data("raw_data")
sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          block_sciat_1 = c("test.sc_dark.Darkbad",
                                            "test.sc_dark.Darkgood"),
                          block_sciat_2 = c("test.sc_milk.Milkbad",
                                            "test.sc_milk.Milkgood"),
                          trial_id  = "trialcode",
```

```
                          trial_eliminate = c("reminder",
                                              "reminder1"))
sciat1 <- sciat_data[[1]]
sciat2 <- sciat_data[[2]]
```

`compute_iat` *Compute IAT D-score*

Description

Compute *D-score* for the IAT according to different algorithms.

Usage

```
compute_iat(data, Dscore = c("d1", "d2", "d3", "d4", "d5", "d6"))
```

Arguments

`data`	Dataframe with class `iat_clean`.
`Dscore`	Character. Indicates which *D-score* to compute. For details on the algorithms, please refer to Greenwald et al. (2003).

Value

Dataframe with class `"dscore"`. The number of rows of the dataframe corresponds to the total number of participants. Variables are defined as follows (the values are specific for each participant):

`participant` Respondents' IDs.

`n_trial` Number of trails before data cleaning.

`nslow10000` Number of slow trials (> 10,000 ms).

`nfast400` Number of fast trials (< 400 ms).

`nfast300` Number of fast trials (< 300 ms).

`accuracy.practice_MappingA` Proportion of correct responses in practice block of Mapping A.

`accuracy.practice_MappingB` Proportion of correct responses in practice block of Mapping B.

`accuracy.test_MappingA` Proportion of correct responses in test block of Mapping A.

`accuracy.test_MappingB` Proportion of correct responses in test block of Mapping B.

`accuracy.MappingA` Proportion of correct responses in Mapping A.

`accuracy.MappingB` Proportion of correct responses in Mapping B.

`RT_mean.MappingA` Mean response time in Mapping A.

`RT_mean.MappingB` Mean response time in Mapping B.

`mean_practice_MappingA` Mean response time in practice block of Mapping A.

`mean_practice_MappingB` Mean response time in practice block of Mapping B.

`mean_test_MappingA` Mean response time in test block of Mapping A.

`mean_test_MappingB` Mean response time in test block of Mapping B.

`d_practice_dX` *D-score*s compute_iat on the practice blocks. The X stands for the selected *D-score* procedure.

`d_test_dX` *D-score*s compute_iat on the test blocks. The X stands for the selected *D-score* procedure.

`dscore_dX` The average *D-score* for the practice and test *D-score*s. The X stands for the selected *D-score* procedure.

`cond_ord` Indicates the order with which the associative conditions have been presented, either `"MappingA_First"` or `"MappingB_First"`.

`legendMappingA` Indicates the corresponding value of Mapping A in the original dataset.

`legendMappingB` Indicates the corresponding value of Mapping B in the original dataset.

Examples

```
# compute D-score 2 for the IAT data ###
  data("raw_data") # import data
  iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                           block_id = "blockcode",
                           mapA_practice = "practice.iat.Milkbad",
                           mapA_test = "test.iat.Milkbad",
                           mapB_practice = "practice.iat.Milkgood",
                           mapB_test = "test.iat.Milkgood",
                           latency_id = "latency",
                           accuracy_id = "correct",
                           trial_id = "trialcode",
                           trial_eliminate = c("reminder", "reminder1"),
                           demo_id = "blockcode",
                           trial_demo = "demo")
  iat_data <- iat_cleandata[[1]]
# calculate D-score
  iat_dscore <- compute_iat(iat_data,
                          Dscore =  "d2")
```

`compute_sciat` *Compute the D-score for the SC-IAT*

Description

Compute the D-score for the SC-IAT.

Usage

```
compute_sciat(
  data,
  mappingA = "mappingA",
  mappingB = "mappingB",
  non_response = NULL
)
```

Arguments

data	Data frame with class clean_sciat.
mappingA	String. Label identifying the mapping A of the SC-IAT in the block_id variable.
mappingB	String. Label identifying the mapping B of the SC-IAT in the block_id variable.
non_response	String. Labels of the trials identifying the non-responses, a.k.a responses beyond the response time window, as it was specified in trial_id (if included).

Value

A dataframe with class compute_sciat. The number of rows of the dataframe corresponds to the total number of participants. Variables are defined as follows (the values are specific for each participant):

participant Respondents' IDs.

n_trial Number of trial before data cleaning.

no_response If there were any trials identifying the non response, it indicates the number of non responses per each participant. Otherwise, it is equal for all participants ("none").

nslow10000 Number of slow trials (> 10,000 ms).

out_accuracy Indicates whether the participants had more than 25 % of incorrect responses in at least one of the critical blocks and hence should be eliminated ("out") or not ("keep").

nfast400 Number of fast trials (< 400 ms).

nfast300 Number of fast trials (< 350 ms – deleted).

accuracy.mappingA Proportion of correct responses in Mapping A.

accuracy.mappingB Proportion of correct responses in mapping B.

RT_mean.MappingA Mean response time in Mapping A.

RT_mean.MappingB Mean response time in Mapping B.

cond_ord Indicates the order with which the associative conditions have been presented, either "MappingA_First" or "MappingB_First".

legendMappingA Indicates the corresponding value of Mapping A in the original dataset.

legendMappingB Indicates the corresponding value of Mapping B in the original dataset.

d_sciat SC-IAT *D*.

Examples

```
# calculate D for the SCIAT
  data("raw_data") # load data
sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          block_sciat_1 = c("test.sc_dark.Darkbad",
                                            "test.sc_dark.Darkgood"),
                          block_sciat_2 = c("test.sc_milk.Milkbad",
```

```
                                          "test.sc_milk.Milkgood"),
                        trial_id  = "trialcode",
                        trial_eliminate = c("reminder",
                                            "reminder1"))
sciat1 <- sciat_data[[1]] # compute D for the first SC-IAT
d_sciat1 <- compute_sciat(sciat1,
                  mappingA = "test.sc_dark.Darkbad",
                  mappingB = "test.sc_dark.Darkgood",
                  non_response = "alert")
head(d_sciat1) # dataframe containing the SC-IAT D of the of the
               # first SC-IAT

sciat2 <- sciat_data[[2]] # Compute D for the second SC-IAT
d_sciat2 <- compute_sciat(sciat2,
                  mappingA = "test.sc_milk.Milkbad",
                  mappingB = "test.sc_milk.Milkgood",
                  non_response = "alert")
head(d_sciat2)
```

`descript_d` *Descriptive table of either the IAT D-score or the SC-IAT Ds*

Description

Descriptive statistics for the IAT *D-score* or the SC-IAT *D*.

Usage

```
descript_d(data, latex = FALSE)
```

Arguments

`data`	Dataframe with either class `dscore` or class `dsciat`.
`latex`	Logical. If `TRUE`, the table for the descriptive statistics will be printed in latex format. Default is `FALSE`.

Value

Dataframe, containing the mean, s.d., minimum and maximum of the IAT (`D-score`, `D-practice`, and `D-test`) or the SC-IAT (`D-Sciat`, `RT.MappingA`, `RT.MappingB`).

Examples

```
# compute D-score 2 for the IAT data ###
  data("raw_data") # import data
  iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          mapA_practice = "practice.iat.Milkbad",
                          mapA_test = "test.iat.Milkbad",
```

```
                          mapB_practice = "practice.iat.Milkgood",
                          mapB_test = "test.iat.Milkgood",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          trial_id = "trialcode",
                          trial_eliminate = c("reminder", "reminder1"),
                          demo_id = "blockcode",
                          trial_demo = "demo")
  iat_data <- iat_cleandata[[1]]
# calculate D-score
  iat_dscore <- compute_iat(iat_data,
                          Dscore =  "d2")
  descript_d(iat_dscore) # descriptive statistics for the IAT

 # calculate D for the SCIAT
  data("raw_data") # load data
sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                        block_id = "blockcode",
                        latency_id = "latency",
                        accuracy_id = "correct",
                        block_sciat_1 = c("test.sc_dark.Darkbad",
                                          "test.sc_dark.Darkgood"),
                        block_sciat_2 = c("test.sc_milk.Milkbad",
                                          "test.sc_milk.Milkgood"),
                        trial_id  = "trialcode",
                        trial_eliminate = c("reminder",
                                            "reminder1"))

 sciat1 <- sciat_data[[1]] # compute D for the first SC-IAT
 d_sciat1 <- compute_sciat(sciat1,
                   mappingA = "test.sc_dark.Darkbad",
                   mappingB = "test.sc_dark.Darkgood",
                   non_response = "alert")
  descript_d(d_sciat1,
             latex = TRUE) # descriptive statistics for the SC-IAT in latex
                           # format
```

`dsciat1` *Data set with SC-IAT D-scores (Dark)*

Description

A data set containing the results of the computation of the D-score on the Dark SC-IAT data set. This data set is used for testing the replicability of the results obtained with the `compute_sciat()` functions.

Usage

```
data("dsciat1")
```

Format

A dataframe with 15 variables, as those described in the documentation for the `compute_sciat()`function.

`dsciat2` *Data set with SC-IAT D-scores (Milk)*

Description

A data set containing the results of the computation of the D-score on the Dark SC-IAT data set. This data set is used for testing the replicability of the results obtained with the `compute_sciat()` functions.

Usage

```
data("dsciat2")
```

Format

A dataframe with 15 variables, as those described in the documentation for the `compute_sciat()` function.

`d_density` *Plot IAT or SC-IAT scores (distribution)*

Description

Plot the distribution of the IAT *D-score* or the SC-IAT *D*.

Usage

```
d_density(
  data,
  graph = c("histogram", "density", "violin"),
  n_bin = 80,
  col_fill = "royalblue",
  col_point = "red",
  include_stats = FALSE
)
```

Arguments

data	Dataframe with either class dscore or dsciat.
graph	String. Indicates the graphs to display. Default is histogram
n_bin	Numeric. Indicates the number of bins to display.
col_fill	String. Indicates the color for filling the bars of the histogram or the curve of the density. Default is royalblue.
col_point	String. Indicates the color for the individual scores –each point – in the violin plot. Default is red.
include_stats	Logical. Indicates whether to add descriptive statistics. The mean is depicted with a solid line. The two dashed lines represent +/- 2 *s.d.* from the mean. Default is FALSE.

Value

A ggplot object.

Examples

```
  # Plotting the IAT D-score
  data("raw_data") # import data
  iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          mapA_practice = "practice.iat.Milkbad",
                          mapA_test = "test.iat.Milkbad",
                          mapB_practice = "practice.iat.Milkgood",
                          mapB_test = "test.iat.Milkgood",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          trial_id = "trialcode",
                          trial_eliminate = c("reminder", "reminder1"),
                          demo_id = "blockcode",
                          trial_demo = "demo")
  iat_data <- iat_cleandata[[1]]
# calculate D-score
  iat_dscore <- compute_iat(iat_data,
                      Dscore =  "d2")
  d_density(iat_dscore) # Default graph
  d_density(iat_dscore, graph = "histogram",
          n_bin = 30) # Histogram with a different number of bins
  d_density(iat_dscore, graph = "density") # IAT D-score density plot
  d_density(iat_dscore, graph = "violin") # IAT D-score violin plot

  # Plot the SC-IAT D for the first SC-IAT
  data("raw_data") # load data
  sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                         block_id = "blockcode",
                         latency_id = "latency",
                         accuracy_id = "correct",
                         block_sciat_1 = c("test.sc_dark.Darkbad",
```

```
                                    "test.sc_dark.Darkgood"),
                          block_sciat_2 = c("test.sc_milk.Milkbad",
                                    "test.sc_milk.Milkgood"),
                          trial_id  = "trialcode",
                          trial_eliminate = c("reminder",
                                      "reminder1"))

sciat1 <- sciat_data[[1]] # compute D for the first SC-IAT
d_sciat1 <- compute_sciat(sciat1,
                mappingA = "test.sc_dark.Darkbad",
                mappingB = "test.sc_dark.Darkgood",
                non_response = "alert")
 d_density(d_sciat1, graph = "histogram",
        include_stats = TRUE) # SC-IAT D histogram with descriptive
                              #  statistics
```

`d_point` *Plot either IAT or SC-IAT scores (points)*

Description

Plot the individual *D-score* or SC-IAT *D*.

Usage

```
d_point(
  data,
  point_size = 1,
  x_label = "Participant",
  x_values = TRUE,
  order_sbj = c("default", "D-increasing", "D-decreasing"),
  col_point = "springgreen4",
  include_stats = FALSE
)
```

Arguments

`data`	Dataframe with either class `dscore` or `dsciat`.
`point_size`	Numeric. Indicates the size of the points in the graph. Default is 1.
`x_label`	Character. Label of the x-axis. Default is `Participant`.
`x_values`	Logical. Shows the values for x-axis (default = `TRUE`).
`order_sbj`	Character. Defines the order with which the participants are displayed. Default is the default order of participants in the dataframe.
`col_point`	Character. Defines the color of the points. Default is `"springgreen4"`.
`include_stats`	Logical. Indicates whether to add descriptive statistics. The `mean` is depicted with a solid line. The two dashed lines represent +/2 *s.d.* from the mean. Default is `FALSE`.

Value

A ggplot object

Examples

```
# Plotting the IAT D-score
  data("raw_data") # import data
  iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          mapA_practice = "practice.iat.Milkbad",
                          mapA_test = "test.iat.Milkbad",
                          mapB_practice = "practice.iat.Milkgood",
                          mapB_test = "test.iat.Milkgood",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          trial_id = "trialcode",
                          trial_eliminate = c("reminder", "reminder1"),
                          demo_id = "blockcode",
                          trial_demo = "demo")
  iat_data <- iat_cleandata[[1]]
# calculate D-score
  iat_dscore <- compute_iat(iat_data,
                       Dscore =  "d2")
  d_point(iat_dscore) # default plot
  d_point(iat_dscore, order_sbj = "D-increasing") # D-score with increasing
                                                  # order
  d_point(iat_dscore, order_sbj = "D-decreasing",
         col_point = "salmon") # D-score with decreasing order changed color
# Plot the SC-IAT D for the first SC-IAT
  data("raw_data") # load data
  sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                        block_id = "blockcode",
                        latency_id = "latency",
                        accuracy_id = "correct",
                        block_sciat_1 = c("test.sc_dark.Darkbad",
                                          "test.sc_dark.Darkgood"),
                        block_sciat_2 = c("test.sc_milk.Milkbad",
                                          "test.sc_milk.Milkgood"),
                        trial_id  = "trialcode",
                        trial_eliminate = c("reminder",
                                            "reminder1"))

 sciat1 <- sciat_data[[1]] # compute D for the first SC-IAT
 d_sciat1 <- compute_sciat(sciat1,
                 mappingA = "test.sc_dark.Darkbad",
                 mappingB = "test.sc_dark.Darkgood",
                 non_response = "alert")
  d_point(d_sciat1, col_point = "salmon",
          include_stats = TRUE) # SC-IAT D with descriptive statistics
```

iatdscores *Data set with IAT D-scores*

Description

A data set containing the results for all the possible D-score algorithms for the IAT. All the algorithms are identified by their corresponding label (such as "dscore_d1"). This data set is used for testing the replicability of the results of the `compute_iat()` function over time.

Usage

```
data("iatdscores")
```

Format

A dataframe with 7 variables, the first one contains the respondents' id, the other 6 columns contain a specific D-score algorithm.

IAT_rel *IAT reliability*

Description

Compute the practice – test IAT reliability.

Usage

```
IAT_rel(data)
```

Arguments

`data`	dataframe with class `"dscore"` (Gawronski et al., 2017).

Value

List of two objects:

`Test-practice reliability` contains the IAT reliability.

`Number of Participants` Contains the number of participants on which the reliability was computed.

Examples

```
# compute D-score 2 for the IAT data ###
  data("raw_data") # import data
  iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          mapA_practice = "practice.iat.Milkbad",
                          mapA_test = "test.iat.Milkbad",
                          mapB_practice = "practice.iat.Milkgood",
                          mapB_test = "test.iat.Milkgood",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          trial_id = "trialcode",
                          trial_eliminate = c("reminder", "reminder1"),
                          demo_id = "blockcode",
                          trial_demo = "demo")
  iat_data <- iat_cleandata[[1]]
# calculate D-score
  iat_dscore <- compute_iat(iat_data,
                         Dscore =  "d2")
  IAT_rel(iat_dscore)
```

`multi_dsciat` *Plot SC-IATs scores*

Description

Plot the scores from two different SC-IATs.

Usage

```
multi_dsciat(
  sciat1,
  sciat2,
  graph = c("density", "violin", "point"),
  x_values = TRUE,
  gcolors = c("dark", "greens", "blues", "pinks"),
  label_sc1 = "SC-IAT1",
  label_sc2 = "SC-IAT2",
  label_y = "SC-IAT scores",
  dens_mean = TRUE
)
```

Arguments

`sciat1`	Dataframe with class `dsciat`. Contains the *D* for the first SC-IAT.
`sciat2`	Dataframe with class `dsciat`. Contains the *D* for the second SC-IAT.
`graph`	String. Type of graph to display. Default is `density`.

x_values	Logical. Shows the values for x-axis (default = TRUE). Only for the point graph.
gcolors	String. Colors palette for plotting the results. Default is dark.
label_sc1	String. Label to display in the graph for the first SC-IAT. Default is SC-IAT1.
label_sc2	String. Label to display in the graph for the first SC-IAT. Default is SC-IAT2.
label_y	String. Label to plot on the y-axis.
dens_mean	Logical. Whether to include the mean in the density plot. Default is TRUE.

Value

A ggplot object

Examples

```
# calculate D for the SCIAT
  data("raw_data") # load data
sciat_data <- clean_sciat(raw_data, sbj_id = "Participant",
                          block_id = "blockcode",
                          latency_id = "latency",
                          accuracy_id = "correct",
                          block_sciat_1 = c("test.sc_dark.Darkbad",
                                            "test.sc_dark.Darkgood"),
                          block_sciat_2 = c("test.sc_milk.Milkbad",
                                            "test.sc_milk.Milkgood"),
                          trial_id  = "trialcode",
                          trial_eliminate = c("reminder",
                                              "reminder1"))

 sciat1 <- sciat_data[[1]] # compute D for the first SC-IAT
 d_sciat1 <- compute_sciat(sciat1,
                    mappingA = "test.sc_dark.Darkbad",
                    mappingB = "test.sc_dark.Darkgood",
                    non_response = "alert") # dataframe with the first D
                                            # SC-IAT

 sciat2 <- sciat_data[[2]] # Compute D for the second SC-IAT
 d_sciat2 <- compute_sciat(sciat2,
                    mappingA = "test.sc_milk.Milkbad",
                    mappingB = "test.sc_milk.Milkgood",
                    non_response = "alert") # dataframe with the first
                                            # D SC-IAT
 multi_dsciat(d_sciat1, d_sciat2) # plot the D of two SC-IATs with default
                                  # settings
```

---

multi_dscore *Compute and plot multiple D-scores*

Description

Compute and plot multiple *D-score*s.

Usage

```
multi_dscore(data, ds = c("built-in", "error-inflation"))
```

Arguments

data	Dataframe of class iat_clean.
ds	String. Indicates which D-score to compute. built-in compute only *D-score* with the built-in error correction (D1 and D2), error-inflation compute the *D-score*s without built-in correction (D3 to D6).

Value

A list. The first object is a dataframe containing all the computed Dscores. The second object is a ggplot object, depicting the distribution of the *D-score*s through violin plots.

@import tidyr

Examples

```
# Compute multiple IAT D-scores
 data("raw_data") # import data
 iat_cleandata <- clean_iat(raw_data, sbj_id = "Participant",
                         block_id = "blockcode",
                         mapA_practice = "practice.iat.Milkbad",
                         mapA_test = "test.iat.Milkbad",
                         mapB_practice = "practice.iat.Milkgood",
                         mapB_test = "test.iat.Milkgood",
                         latency_id = "latency",
                         accuracy_id = "correct",
                         trial_id = "trialcode",
                         trial_eliminate = c("reminder", "reminder1"),
                         demo_id = "blockcode",
                         trial_demo = "demo")
 iat_data <- iat_cleandata[[1]]

 # compute the mulitple scores and prepare the graphs for the built-in
 # strategies

 multiple_scores <- multi_dscore(iat_data, ds = "built-in")

 data_multiple <- multiple_scores$dscores # store the D-score in a dataframe

 # plot the results
 multiple_scores$graph
```

raw_data *Dataset with one IAT and two SC-IATs*

Description

A dataset containing the data from 152 participants who completed one IAT and two SC-IATs. The object of both the implicit measures was chocolate, either Milk or Dark chocolate:

Usage

```
data(raw_data)
```

Format

A dataframe with 6 variables, as follows:

- Participant. Participants ID.
- latency. Latency of the response times in millisecond.
- correct. Response accuracy (0–correct, 1–error).
- trialcode. Factor with 32 levels identifying the trial for each response, both for the implicit measures and the demographic questionnaire. It contains also the trials that have to be eliminated, defined as follows:
  - alert. Defines the SC-IAT trials beyond the response time window.
  - Reminder, Reminder1. Identify the instruction page.
- blockcode. Factor with 13 levels as follow:
  - practice.iat.Milkbad. IAT practice blocks, Mapping A.
  - practice.iat.Milkbad. IAT practice blocks, Mapping B.
  - practice.sc_dark.Darkbad. Dark SC-IAT practice blocks, Mapping A.
  - practice.sc_dark.Darkbad. Dark SC-IAT practice blocks, Mapping B.
  - practice.sc_milk.Milkbad. Milk SC-IAT practice blocks, Mapping A.
  - practice.sc_milk.Milkgood. Milk SC-IAT practice blocks, Mapping B.
  - test.iat.Milkbad. IAT test blocks, Mapping A.
  - test.iat.Milkgood. IAT test blocks, Mapping B.
  - test.sc_dark.Darkbad. Dark SC_IAT test blocks, Mapping A.
  - test.sc_dark.Darkbad. Dark SC-IAT test blocks, Mapping B.
  - test.sc_milk.Milkbad. Milk SC-IAT test blocks, Mapping A.
  - test.sc_milk.Milkgood. Milk SC-IAT test blocks, Mapping B.
  - demo. Demographic questionnaire.
- response. Character registering the type of response for the demographic .

Index