

Package ‘eyeTrackR’

March 29, 2020

Type Package

Title Organising and Analysing Eye-Tracking Data

Version 1.0.1

Date 2020-03-28

Maintainer Hayward Godwin <hg102@soton.ac.uk>

Description A set of functions for organising and analysing datasets from experiments run using 'Eyelink' eye-trackers. Organising functions help to clean and prepare eye-tracking datasets for analysis, and mark up key events such as display changes and responses made by participants. Analysing functions help to create means for a wide range of standard measures (such as 'mean fixation durations'), which can then be fed into the appropriate statistical analyses and graphing packages as necessary.

License GPL-3

Depends data.table, stringr, plyr, R (>= 3.5)

RoxygenNote 7.0.1

LazyData False

Suggests knitr, rmarkdown

Encoding UTF-8

NeedsCompilation no

Author Hayward Godwin [aut, cre],
Alexander Muhl-Richardson [ctb]

Repository CRAN

Date/Publication 2020-03-29 11:00:14 UTC

R topics documented:

analyse.behavioural.data	2
analyse.calculate.means	3
analyse.fix.count	4
analyse.fix.duration	5
analyse.fix.first_duration	6

analyse.fix.totaltime	7
analyse.sac.amplitude	8
analyse.visit.count	9
fixationreport	10
messagereport	10
organise.behavioural.base	11
organise.checks.random_trial	12
organise.contingencies.descriptives	13
organise.exclusions.fix_durations	14
organise.message.descriptives	16
organise.message.fix_contingencies	16
organise.message.markup	17
organise.message.removals	18
organise.message.replace_spaces	19
organise.message.return_specific	20
organise.responses.markup	21

Index 23

analyse.behavioural.data
Analyse behavioural data

Description

Analyse behavioural data

Usage

```
analyse.behavioural.data(bd_df, aggregation_column_list = c())
```

Arguments

bd_df Behavioural data frame/table
aggregation_column_list
 List of columns to group by

Value

Provides behavioural information for the experiment as a data.table.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT
data(fixationreport)
data(messagereport)
```

```
# REPLACE SPACES IN MESSAGES
```

```

messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
organise.message.descriptives(messagereport)

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
  fixreport_df = fixationreport, message="DISPLAY_START")
fixationreport <- organise.message.markup(message_df=messagereport,
  fixreport_df = fixationreport, message="DISPLAY_CHANGE")

# NOW DO ACCURACY AND RT MARKUP
fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport <-organise.message.fix_contingencies(fixationreport,
  list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

# SET UP TRUE RT
fixationreport[,TRUE_RT:=RESPONSE_TIME-DISPLAY_START,]

behaviouralData <- analyse.behavioural.data(fixationreport,
  aggregation_column_list = list('TRIALTYPE_TEXT'))

```

analyse.calculate.means

Generic function for calculating means

Description

Generic function for calculating means

Usage

```

analyse.calculate.means(
  fixreport_df,
  aggregation_column_list,
  output_column_expression,
  final_output_column_expression,
  spss,
  dvColumnName,
  prefixLabel = "",
  debug = FALSE
)

```

Arguments

fixreport_df Fixation report

aggregation_column_list	List of columns to group by
output_column_expression	Output column expression
final_output_column_expression	Final output column expression
spss	Should the function output for SPSS?
dvColumnName	Column name of the dependent variable
prefixLabel	Prefix label
debug	Should debug information be provided?

Value

A data.table ready for SPSS analyses, which is also saved to disk as a text file.

Examples

```
# THIS IS A UTILITY FUNCTION THAT YOU WOULD NOT NORMALLY USE YOURSELF
```

analyse.fix.count	<i>Analyse mean fixation count</i>
-------------------	------------------------------------

Description

Analyse mean fixation count

Usage

```
analyse.fix.count(
  fixreport_df,
  aggregation_column_list = c(),
  spss = FALSE,
  prefixLabel = ""
)
```

Arguments

fixreport_df	Fixation report
aggregation_column_list	List of columns to group by
spss	Should the function save output for SPSS?
prefixLabel	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for fixation counts, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the for-spss file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT
data(fixationreport)
fixCounts <- analyse.fix.count(fixationreport, aggregation_column_list = list('TRIALTYPE_TEXT'))
```

`analyse.fix.duration` *Analyse mean fixation duration*

Description

Analyse mean fixation duration

Usage

```
analyse.fix.duration(  
  fixreport_df,  
  aggregation_column_list = c(),  
  spss = FALSE,  
  prefixLabel = ""  
)
```

Arguments

<code>fixreport_df</code>	Fixation report
<code>aggregation_column_list</code>	List of columns to group by
<code>spss</code>	Should the function save output for SPSS?
<code>prefixLabel</code>	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for fixation durations, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the for-spss file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT
data(fixationreport)
fixDurs <- analyse.fix.duration(fixationreport, aggregation_column_list = list('TRIALTYPE_TEXT'))
```

```
analyse.fix.first_duration
      Analyse first fixation duration
```

Description

Analyse first fixation duration

Usage

```
analyse.fix.first_duration(
  fixreport_df,
  aggregation_column_list = c(),
  spss = FALSE,
  prefixLabel = ""
)
```

Arguments

fixreport_df	Fixation report
aggregation_column_list	List of columns to group by
spss	Should the function save output for SPSS?
prefixLabel	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for first fixation durations, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the for-spss file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT
data(fixationreport)
firstDurations <- analyse.fix.first_duration(fixationreport,
      aggregation_column_list = list('TRIALTYPE_TEXT'))
```

`analyse.fix.totaltime` *Analyse total fixation time*

Description

Analyse total fixation time

Usage

```
analyse.fix.totaltime(  
  fixreport_df,  
  aggregation_column_list = c(),  
  spss = FALSE,  
  prefixLabel = ""  
)
```

Arguments

<code>fixreport_df</code>	Fixation report
<code>aggregation_column_list</code>	List of columns to group by
<code>spss</code>	Should the function save output for SPSS?
<code>prefixLabel</code>	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for total fixation times, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the for-spss file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT  
data(fixationreport)  
fixTotaltime <- analyse.fix.totaltime(fixationreport,  
  aggregation_column_list = list('TRIALTYPE_TEXT'))
```

analyse.sac.amplitude *Analyse saccade amplitude*

Description

Analyse saccade amplitude

Usage

```
analyse.sac.amplitude(  
  fixationreport_df,  
  aggregation_column_list = c(),  
  spss = FALSE,  
  prefixLabel = ""  
)
```

Arguments

fixationreport_df	Fixation report
aggregation_column_list	List of columns to group by
spss	Should the function save output for SPSS?
prefixLabel	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for saccade amplitudes, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the `for-spss` file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT  
data(fixationreport)  
amplitudes <- analyse.sac.amplitude(fixationreport,  
  aggregation_column_list = list('TRIALTYPE_TEXT'))
```

`analyse.visit.count` *Analyse visit count*

Description

Analyse visit count

Usage

```
analyse.visit.count(  
  fixreport_df,  
  aggregation_column_list = c(),  
  spss = FALSE,  
  prefixLabel = ""  
)
```

Arguments

<code>fixreport_df</code>	Fixation report
<code>aggregation_column_list</code>	List of columns to group by
<code>spss</code>	Should the function save output for SPSS?
<code>prefixLabel</code>	Prefix label

Value

If `spss` is set to `FALSE` (which is the default), you'll get an object containing `data.tables` of by-trial means for number of visits to each object, by-trial means for participants, and overall descriptive statistics for use when creating graphs based on your data. If `spss` is set to `TRUE`, then you'll be provided with a 'wide' version of the data for analysis in packages such as SPSS. The function will also save a copy of the for-spss file for you as well.

Examples

```
# BREAK UP BY TARGET-PRESENT AND TARGET-ABSENT TRIALS - THE COLUMN TRIALTYPE_TEXT  
data(fixationreport)  
fixationreport[,CURRENT_FIX_INTEREST_AREA_RUN_ID:=1,]  
visitCounts <- analyse.visit.count(fixationreport, aggregation_column_list = list('TRIALTYPE_TEXT'))
```

fixationreport	<i>Example fixation report dataset</i>
----------------	--

Description

Fixation report data from a visual search experiment.

Usage

```
data(fixationreport)
```

Format

A data.table object.

References

TBA.

Examples

```
data(fixationreport)
```

messagereport	<i>Example message report dataset</i>
---------------	---------------------------------------

Description

Message report data from a visual search experiment.

Usage

```
data(messagereport)
```

Format

A data.table object.

References

TBA.

Examples

```
data(messagereport)
```

`organise.behavioural.base`*Save RT and Accuracy split by specified columns.*

Description

Save RT and Accuracy split by specified columns.

Usage

```
organise.behavioural.base(  
  fixreport_df,  
  grouping_column_list,  
  response_period_start = ""  
)
```

Arguments

`fixreport_df` Fixation report.
`grouping_column_list`
List of columns to split by.
`response_period_start`
Message that starts the RT timer.

Value

Summarised behavioural information as a data.table.

Examples

```
data(fixationreport)  
data(messagereport)  
  
# REPLACE SPACES IN MESSAGES  
messagereport <- organise.message.replace_spaces(messagereport)  
  
# TAKE A LOOK  
print(organise.message.descriptives(messagereport))  
  
# MARKUP  
fixationreport <- organise.message.markup(message_df=messagereport,  
fixreport_df = fixationreport, message="DISPLAY_START")  
  
fixationreport <- organise.message.markup(message_df=messagereport,  
fixreport_df = fixationreport, message="DISPLAY_CHANGE")  
  
# NOW DO ACCURACY AND RT MARKUP
```

```

fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport<-organise.message.fix_contingencies(fixationreport,
list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

# SET UP TRUE RT
fixationreport[,TRUE_RT:=RESPONSE_TIME-DISPLAY_START,]

behaviouralData <- analyse.behavioural.data(fixationreport,
aggregation_column_list = list('TRIALTYPE_TEXT'))

# RANDOM TRIAL TO CHECK THINGS OUT
print(organise.checks.random_trial(fixationreport))

# FIX CONTINGENCIES
print(organise.contingencies.descriptives(fixationreport))

# REMOVE MISSING EVENTS - HERE, TRIALS WHICH LACKED A RESPONSE
messageRemovals <- organise.message.removals(fixreport_df=fixationreport,
required_message_list=list("DISPLAY_CHANGE", "RESPONSE_TIME"))

# LOOK AT MESSAGE REMOVALS
print(messageRemovals[[1]])

# GRAB THE FIXATION REPORT WITH TRIALS REMOVED
fixMessagesRemoved <- messageRemovals[[2]]

# THIS SHOWS WE HAVE NO UNCLASSIFIED FIXATIONS, GOOD!
print(organise.contingencies.descriptives(fixMessagesRemoved))

# GET A BEHAVIOURAL DATASET FOR ANALYSES AND SAVING ETC.
behavDT<- organise.behavioural.base(fixreport_df = fixMessagesRemoved,
list( 'TRIALTYPE_TEXT'), response_period_start="DISPLAY_START")

```

```
organise.checks.random_trial
```

Return a randomly selected trial for detailed checks.

Description

Return a randomly selected trial for detailed checks.

Usage

```
organise.checks.random_trial(fixreport_df)
```

Arguments

fixreport_df object Input fixation report.

Value

Single trial as a data.table, which can be printed to the console for your viewing.

Examples

```
data(fixationreport)
print(organise.checks.random_trial(fixationreport))
```

```
organise.contingencies.descriptives
  Descriptive statistics of fixation contingencies.
```

Description

Descriptive statistics of fixation contingencies.

Usage

```
organise.contingencies.descriptives(fixreport_df)
```

Arguments

fixreport_df Fixation report.

Value

Output to console.

Examples

```
data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_CHANGE")

# NOW DO ACCURACY AND RT MARKUP
```

```

fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport<-organise.message.fix_contingencies(fixationreport,
          list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

# SET UP TRUE RT
fixationreport[,TRUE_RT:=RESPONSE_TIME-DISPLAY_START,]

behaviouralData <- analyse.behavioural.data(fixationreport,
          aggregation_column_list = list('TRIALTYPE_TEXT'))

# RANDOM TRIAL TO CHECK THINGS OUT
print(organise.checks.random_trial(fixationreport))

# FIX CONTINGENCIES
print(organise.contingencies.descriptives(fixationreport))

```

```

organise.exclusions.fix_durations
          Exclude very brief and very long fixations.

```

Description

Exclude very brief and very long fixations.

Usage

```
organise.exclusions.fix_durations(fixreport_df, min = 60, max = 1200)
```

Arguments

fixreport_df	Fixation report.
min	Minimum duration of fixations.
max	Maximum duration of fixations.

Value

A data.table detailing how many trials were removed from each session, plus a data.table with the cleaned fixation report.

Examples

```

data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

```

```

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
fixreport_df = fixationreport, message="DISPLAY_CHANGE")

# NOW DO ACCURACY AND RT MARKUP
fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport<-organise.message.fix_contingencies(fixationreport,
list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

# SET UP TRUE RT
fixationreport[,TRUE_RT:=RESPONSE_TIME-DISPLAY_START,]

behaviouralData <- analyse.behavioural.data(fixationreport,
aggregation_column_list = list('TRIALTYPE_TEXT'))

# RANDOM TRIAL TO CHECK THINGS OUT
print(organise.checks.random_trial(fixationreport))

# FIX CONTINGENCIES
print(organise.contingencies.descriptives(fixationreport))

# REMOVE MISSING EVENTS - HERE, TRIALS WHICH LACKED A RESPONSE
messageRemovals <- organise.message.removals(fixreport_df=fixationreport,
required_message_list=list("DISPLAY_CHANGE", "RESPONSE_TIME"))

# LOOK AT MESSAGE REMOVALS
print(messageRemovals[[1]])

# GRAB THE FIXATION REPORT WITH TRIALS REMOVED
fixMessagesRemoved <- messageRemovals[[2]]

# THIS SHOWS WE HAVE NO UNCLASSIFIED FIXATIONS, GOOD!
print(organise.contingencies.descriptives(fixMessagesRemoved))

# GET A BEHAVIOURAL DATASET FOR ANALYSES AND SAVING ETC.
behavDT<- organise.behavioural.base(fixreport_df = fixMessagesRemoved,
list( 'TRIALTYPE_TEXT'), response_period_start="DISPLAY_START")

# REMOVALS BASED ON FIXATION DURATIONS
durationRemovals <- organise.exclusions.fix_durations(fixreport_df=fixMessagesRemoved)

durationsRemoved <- durationRemovals[[1]]

# FINAL DATASET WHICH CAN BE ANALYSED
finalDT <- durationRemovals [[2]]

```

```
organise.message.descriptives
```

Descriptive statistics for messages in message report.

Description

Descriptive statistics for messages in message report.

Usage

```
organise.message.descriptives(message_df)
```

Arguments

message_df Message report.

Value

Descriptive information relating to messages in the trials which can be printed to the console.

Examples

```
data(messagereport)
print(organise.message.descriptives(messagereport))
```

```
organise.message.fix_contingencies
```

Oganise and markup fixation contingencies.

Description

Oganise and markup fixation contingencies.

Usage

```
organise.message.fix_contingencies(fixreport_df, ordered_message_list)
```

Arguments

fixreport_df Fixation report.

ordered_message_list

List of messages to markup, in temporal order at which they occurred.

Value

Marked-up fixation report data.table.

Examples

```

data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_CHANGE")

# NOW DO ACCURACY AND RT MARKUP
fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport<-organise.message.fix_contingencies(fixationreport,
                                                  list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

```

```
organise.message.markup
```

Markup trial messages.

Description

Markup trial messages.

Usage

```

organise.message.markup(
  message_df,
  fixreport_df,
  message,
  show_working = FALSE
)

```

Arguments

message_df	Message report
fixreport_df	Fixation report
message	The message or event you want to mark up
show_working	Should eyeTrackR show more detail when calculating the output?

Value

An updated fixation report with the message marked up into each trial. If there is a difference between the number of input and output rows, there was a problem with the joining of your data. You'll have a repeated session name or trial index.

Examples

```
data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
  fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
  fixreport_df = fixationreport, message="DISPLAY_CHANGE")
```

```
organise.message.removals
```

Remove trials which fail to have all of the listed messages.

Description

Remove trials which fail to have all of the listed messages.

Usage

```
organise.message.removals(fixreport_df, required_message_list)
```

Arguments

```
fixreport_df    Fixation report.
required_message_list
                List of messages required for each trial.
```

Value

A data.table detailing how many trials were removed from each session, plus a data.table with the cleaned fixation report.

Examples

```

data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_CHANGE")

# NOW DO ACCURACY AND RT MARKUP
fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")

# NOW MARK UP FIXATION CONTINGENCIES
fixationreport<-organise.message.fix_contingencies(fixationreport,
                                                  list("DISPLAY_START", "DISPLAY_CHANGE", "RESPONSE_TIME"))

# SET UP TRUE RT
fixationreport[,TRUE_RT:=RESPONSE_TIME-DISPLAY_START,]

behaviouralData <- analyse.behavioural.data(fixationreport,
                                           aggregation_column_list = list('TRIALTYPE_TEXT'))

# RANDOM TRIAL TO CHECK THINGS OUT
print(organise.checks.random_trial(fixationreport))

# FIX CONTINGENCIES
print(organise.contingencies.descriptives(fixationreport))
messageRemovals <- organise.message.removals(fixreport_df=fixationreport,
                                             required_message_list=list("DISPLAY_CHANGE", "RESPONSE_TIME"))

# LOOK AT MESSAGE REMOVALS
print(messageRemovals[[1]])

# GRAB THE FIXATION REPORT WITH TRIALS REMOVED
fixMessagesRemoved <- messageRemovals[[2]]

# THIS SHOWS WE HAVE NO UNCLASSIFIED FIXATIONS, GOOD!
print(organise.contingencies.descriptives(fixMessagesRemoved))

```

```
organise.message.replace_spaces
```

Replace spaces in message report message with underscores.

Description

Replace spaces in message report message with underscores.

Usage

```
organise.message.replace_spaces(message_df)
```

Arguments

`message_df` A message report.

Value

An updated message report with spaces between words replaced with underscores.

Examples

```
data(messagereport)
messagereport <- organise.message.replace_spaces(messagereport)
```

```
organise.message.return_specific
```

Return trials where a specific message is found.

Description

Return trials where a specific message is found.

Usage

```
organise.message.return_specific(
  message_df,
  fixreport_df,
  message,
  show_working = FALSE
)
```

Arguments

`message_df` Message report.
`fixreport_df` Fixation report.
`message` The message you want to search for.
`show_working` Should eyeTrackR show more detail when calculating the output?

Value

Data.table of marked up fixation report.

Examples

```
# HERE, 'SYNCTIME' STARTS A TRIAL
data(messagereport)
data(fixationreport)

print(organise.message.return_specific(messagereport, fixationreport, 'DISPLAY_START'))
```

```
organise.responses.markup
      Mark up responses into a fixation report.
```

Description

Mark up responses into a fixation report.

Usage

```
organise.responses.markup(fixreport_df, correct_answer_column)
```

Arguments

```
fixreport_df    Fixation report
correct_answer_column
                 The column in the fixation report containing the correct button response number
                 (1-7).
```

Value

Updated fixation report as a data.table.

Examples

```
data(fixationreport)
data(messagereport)

# REPLACE SPACES IN MESSAGES
messagereport <- organise.message.replace_spaces(messagereport)

# TAKE A LOOK
print(organise.message.descriptives(messagereport))

# MARKUP
fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_START")

fixationreport <- organise.message.markup(message_df=messagereport,
                                         fixreport_df = fixationreport, message="DISPLAY_CHANGE")
```

```
# NOW DO ACCURACY AND RT MARKUP  
fixationreport <- organise.responses.markup(fixationreport, "CORRECT_RESPONSE")
```

Index

*Topic **datasets**

- fixationreport, [10](#)
- messagereport, [10](#)

- analyse.behavioural.data, [2](#)
- analyse.calculate.means, [3](#)
- analyse.fix.count, [4](#)
- analyse.fix.duration, [5](#)
- analyse.fix.first_duration, [6](#)
- analyse.fix.totaltime, [7](#)
- analyse.sac.amplitude, [8](#)
- analyse.visit.count, [9](#)

- fixationreport, [10](#)

- messagereport, [10](#)

- organise.behavioural.base, [11](#)
- organise.checks.random_trial, [12](#)
- organise.contingencies.descriptives, [13](#)
- organise.exclusions.fix_durations, [14](#)
- organise.message.descriptives, [16](#)
- organise.message.fix_contingencies, [16](#)
- organise.message.markup, [17](#)
- organise.message.removals, [18](#)
- organise.message.replace_spaces, [19](#)
- organise.message.return_specific, [20](#)
- organise.responses.markup, [21](#)