Package 'WhatsR'

July 14, 2025

Type Package

Title Parsing, Anonymizing and Visualizing Exported 'WhatsApp' Chat Logs

Version 1.0.6

Date 2025-07-14

Description Imports 'WhatsApp' chat logs and parses them into a usable dataframe object. The parser works on chats exported from Android or iOS phones and on Linux, macOS and Windows. The parser has multiple options for extracting smileys and emojis from the messages, extracting URLs and domains from the messages, extracting names and types of sent media files from the messages, extracting timestamps from messages, extracting and anonymizing author names from messages. Can be used to create anonymized versions of data.

License GPL-3

Imports stringi, qdapRegex, readr, tokenizers, data.table, ggplot2, anytime, mgsub, stats, qdap, ggwordcloud, dplyr, ragg, checkmate, visNetwork, lubridate, methods, leaflet

Encoding UTF-8

RoxygenNote 7.3.2

Suggests extrafont, tm, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://gesiscss.github.io/WhatsR/, https://github.com/gesiscss/WhatsR

BugReports https://github.com/gesiscss/WhatsR/issues

NeedsCompilation no

Author Julian Kohne [aut, cre]

Maintainer Julian Kohne <julian.kohne@gesis.org>

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2025-07-14 14:40:02 UTC

Contents

`create_chatlog` *Creating test data in the structure of 'WhatsApp' chat logs*

Description

Creates a .txt file in the working directory that has the same structure as chat logs exported from 'WhatsApp'. Messages have a timestamp, sender name and message body containing lorem ipsum, emoji, links, smilies, location, omitted media files, linebreaks, self-deleting photos, and 'WhatsApp' system messages. Timestamps are formatted according to specified phone operating system and time format settings. 'WhatsApp' system messages are formatted according to specified phone operating system and language.

Usage

```
create_chatlog(
  n_messages = 150,
  n_chatters = 2,
  n_emoji = 50,
  n_diff_emoji = 20,
  n_links = 20,
  n_locations = 5,
  n_smilies = 20,
  n_diff_smilies = 15,
```

```
  n_media = 10,
  media_excluded = TRUE,
  n_sdp = 3,
  n_deleted = 5,
  startdate = "01.01.2019",
  enddate = "31.12.2022",
  language = "german",
  time_format = "24h",
  os = "android",
  path = getwd(),
  chatname = "Simulated_WhatsR_chatlog"
)
```

Arguments

n_messages	Number of messages that are contained in the created .txt file.
n_chatters	Number of different chatters present in the created .txt file.
n_emoji	Number of messages that contain emoji. Must be smaller or equal to n_messages.
n_diff_emoji	Number of different emoji that are used in the simulated chat.
n_links	Number of messages that contain links. Must be smaller or equal to n_messages.
n_locations	Number of messages that contain locations. Must be smaller or equal to n_messages.
n_smilies	Number of messages that contain smilies. Must be smaller or equal to n_messages.
n_diff_smilies	Number of different smilies that are used in the simulated chat.
n_media	Number of messages that contain media files. Must be smaller or equal to n_messages.
media_excluded	Whether media files were excluded in simulated export or not. Default is TRUE.
n_sdp	Number of messages that contain self-deleting photos. Must be smaller or equal to n_messages.
n_deleted	Number of messages that contain deleted messages. Must be smaller or equal to n_messages.
startdate	Earliest possible date for messages. Format is 'dd.mm.yyyy'. Timestamps for messages are created automatically between startdate and enddate. Input is interpreted as UTC
enddate	Latest possible date for messages. Format is 'dd.mm.yyyy'. Timestamps for messages are created automatically between startdate and enddate. Input is interpreted as UTC
language	Parameter for the language setting of the exporting phone. Influences structure of system messages
time_format	Parameter for the time format setting of the exporting phone (am/pm vs. 24h). Influences the structure of timestamps.
os	Parameter for the operating system setting of the exporting phone. Influences the structure of timestamps and 'WhatsApp' system messages.
path	Character string for indicating the file path of where to save the file. Can be NA to not save a file. Default is getwd()
chatname	Name for the created .txt file.

Value

A .txt file with a simulated 'WhatsApp' chat containing lorem ipsum but all structural properties of actual chats.

Examples

```
SimulatedChat <- create_chatlog(path = NA)
```

`download_emoji` *Scraping a dictionary of emoji from https://www.unicode.org/*

Description

Scrapes a dictionary of emoji from https://www.unicode.org/, assuming that the website is available and its structure does not change. Can be used to update the emoji dictionary contained in this package by replacing the file and recompiling the package. The dictionary is ordered according to the length of the emojis' byte representation (longer ones first) to prevent partial matching of shorter strings when iterating through the data frame.

Usage

```
download_emoji(
  unicode_page = "https://www.unicode.org/Public/emoji/15.1/emoji-test.txt",
  delete_header = 32,
  nlines = -1L
)
```

Arguments

`unicode_page`	URL to the unicode page containing the emoji dictionary.
`delete_header`	Number of lines to delete from the top of the file.
`nlines`	Number of lines to read from the file. Passed to `readLines` as n. Negative Integers will read all lines.

Value

A data frame containing:
1) The native representation (glyphs) of all emoji in R
2) A textual description of what the emoji is displaying
3) The hexadecimal codepoints of the emoji
4) The status of the emoji (e.g. "fully-qualified" or "component")
5) Original order of the .txt file that the emoji were fetched from

Examples

```
Emoji_dictionary <- download_emoji(nlines = 50)
```

parse_android *Parsing raw 'WhatsApp' chat logs according to Android text structure*

Description

Creates a data frame from an exported 'WhatsApp' chat log containing one row per message and a column for DateTime when the message was sent, name of the sender and body of the message. Only works as an intermediary function called from within parse_chat

Usage

```
parse_android(
  chatlog,
  newline_indicator = "\n",
  media_omitted = "<media omitted>",
  media_indicator = "(file attached)",
  sent_location = paste0("location: (?=https:\\/\\/maps\\.google\\.com\\/",
    "\\?q=\\d\\d.\\d{6}\\,\\d\\.\\d{6})"),
  live_location = "^live location shared$",
  datetime_indicator = paste("(?!^)(?=((\\d{2}\\.\\d{2}\\.\\d{2})|(\\d{1,2}",
   "\\/\\d{1,2}\\/\\d{2})),\\s\\d{2}\\:\\d{2}((\\s\\-)|(\\s(?i:(am|pm))\\s\\-)))",
    sep = ""),
  newline_replace = " start_newline ",
  media_replace = " media_omitted ",
  foursquare_loc = "^.*: https://foursquare.com/v/.*$"
)
```

Arguments

chatlog	'WhatsApp' chat preprocessed by parse_chat
newline_indicator	character string defining character for newline indicators. Default is a Unicode newline.
media_omitted	character string inserted by 'WhatsApp' instead of file names when not exporting media.
media_indicator	character string for detecting media and file attachments.
sent_location	Regex for detecting auto generated messages for locations shared via chat.
live_location	Regex for detecting auto generated messages for live locations shared via chat.
datetime_indicator	Regex for detecting the DateTime indicator at the beginning of each message.
newline_replace	replacement string for a newline character in parsed message. Default is " start_newline ".
media_replace	replacement string for omitted media files. Default is " media_omitted ".
foursquare_loc	Regex for detecting sent Locations as FourSquare Links.

Value

A data frame containing the timestamp, name of the sender and message body

Examples

```
ParsedChat <- parse_android("29.01.18, 23:33 - Alice: Hi?\n 29.01.18, 23:45 - Bob: Hi\n")
```

`parse_chat` *Parsing exported 'WhatsApp' chat logs as a dataframe*

Description

Creates a data frame from an exported 'WhatsApp' chat log containing one row per message. Some columns are saved as lists using the I() function so that multiple elements can be stored per message while still maintaining the general structure of one row per message. These columns should be treated as lists or unlisted first.

Usage

```
parse_chat(
  path,
  os = "auto",
  language = "auto",
  anonymize = "add",
  consent = NA,
  emoji_dictionary = "internal",
  smilie_dictionary = "wikipedia",
  rpnl = " start_newline ",
  verbose = FALSE
)
```

Arguments

`path`	Character string containing the file path to the exported 'WhatsApp' chat log as a .txt file.
`os`	Operating system of the phone the chat was exported from. Default "auto" tries to automatically detect the OS. Also supports "android" or "iOS".
`language`	Indicates the language setting of the phone with which the messages were exported. Default is "auto" trying to match either 'English' or 'German'. More languages might be supported in the future.
`anonymize`	TRUE results in the vector of sender names being anonymized and columns containing personal identifiable information to be deleted or restricted, FALSE displays the actual names and all content, "add" adds anonomized columns to the full info columns. Do not blindly trust this and always double check.

consent
: String containing a consent message. All messages from chatters who have not posted this *exact* message into the chat will be deleted. Default is NA, no deleting anything.

emoji_dictionary
: Dictionary for emoji matching. Can use a version included in this package when set to "internal" or an updated data frame created by download_emoji passed as a character string containing the path to the file.

smilie_dictionary
: Value "emoticons" uses ex_emoticon to extract smilies, "wikipedia" uses a more inclusive custom list of smilies containing all mentions from https://de.wiktionary.org/w/index.php?t and manually added ones.

rpnl
: Replace newline. A character string for replacing line breaks within messages for the parsed message for better readability. Default is " start_newline ".

verbose
: Prints progress messages for parse_chat() to the console if TRUE, default is FALSE.

Value

A dataframe containing one row per message and 11,15, or 19 columns, depending on the setting of the anonymize parameter

Examples

```
data <- parse_chat(system.file("englishandroid24h.txt", package = "WhatsR"))
```

---

parse_ios *Parsing raw 'WhatsApp' chat log according to iOs text structure*

Description

Creates a data frame from an exported 'WhatsApp' chat log containing one row per message and a column for DateTime when the message was send, name of the sender and body of the message. Only works as an intermediary function called from within parse_chat

Usage

```
parse_ios(
  chatlog,
  newline_indicator = "\n",
  media_omitted = "<media omitted>",
  media_indicator = "^<attached:\\s(.)*?\\.(.)*?>$",
  sent_location = paste0("location: (?=https:\\/\\/maps\\.google\\.com\\/",
    "\\?q=\\d\\d.\\d{6}\\,\\d\\.\\d{6})"),
  live_location = "^live location shared$",
  datetime_indicator = paste("(?!^)(?=\\[((\\d{2}\\.\\d{2}\\.\\d{2})|",
    "(\\d{1,2}\\/\\d{1,2}\\/\\d{2})),\\s\\d{1,2}\\:\\d{2}((\\:\\d{2}\\",
```

```
    "s(?i:(pm|am)))|(\\s(?i:(pm|am)))|(\\:\\d{2}\\])|(\\:\\d{2})|(\\s))\\])",
    sep = ""),
  newline_replace = " start_newline ",
  media_replace = " media_omitted ",
  foursquare_loc = "^.*: https://foursquare.com/v/.*$"
)
```

Arguments

chatlog	'WhatsApp' chat preprocessed by parse_chat
newline_indicator	Character string defining character for newline indicators. Default is a Unicode newline.
media_omitted	Character string inserted by 'WhatsApp' instead of file names when not exporting media.
media_indicator	Character string for detecting media and file attachments.
sent_location	Regex for detecting auto generated messages for locations shared via chat.
live_location	Regex for detecting auto generated messages for locations shared via chat.
datetime_indicator	Regex for detecting the DateTime indicator at the beginning of each message.
newline_replace	Replacement string for a newline character in parsed message. Default is " start_newline ".
media_replace	Replacement string for omitted media files. Default is " media_omitted ".
foursquare_loc	Regex for detecting sent Locations as FourSquare Links.

Value

A data frame containing the timestamp, name of the sender and message body

Examples

```
ParsedChat <- parse_ios("[29.01.18, 23:33:00] Alice: Hello?\\n [29.01.18, 23:45:01] Bob: Hello")
```

---

plot_emoji *Plotting emoji distributions in 'WhatsApp' chat logs*

Description

Plots four different types of graphs for the emoji contained in a parsed 'WhatsApp' chat log. Returns dataframe used for plotting if desired.

Usage

```
plot_emoji(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  min_occur = 1,
  return_data = FALSE,
  emoji_vec = "all",
  plot = "bar",
  emoji_size = 10,
  font_family = "Noto Color Emoji",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with anytime. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with anytime. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
min_occur	Minimum number of occurrences for emoji to be included in the plots. Default is 1.
return_data	If TRUE, returns the subsetted data frame used for plotting. Default is FALSE.
emoji_vec	A vector of emoji that the visualizations and data will be restricted to.
plot	The type of plot that should be returned. Options are "heatmap", "cumsum", "bar" and "splitbar".
emoji_size	Determines the size of the emoji displayed on top of the bars for "bar" and "splitbar", default is 10.
font_family	Character string for indicating font family used to plot_emoji. Fonts might need to be installed manually, see font_import.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

Plots and/or the subset data frame based on author names, datetime and emoji occurrence

Examples

```
# importing data
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))

# opening AGG graphics device from the ragg package (replace tempfile with filepath)
ragg::agg_png(tempfile(), width = 800, height = 600, res = 150)

# plotting emoji
plot_emoji(data,font_family="Times", exclude_sm = TRUE) #font_family = "Noto Color Emoji" on Linux

# Close the AGG device
dev.off()
```

`plot_lexical_dispersion` *Lexical disperson plots for keywords in 'WhatsApp' chat logs*

Description

Visualizes the occurrence of specific keywords within the chat. Requires the raw message content to be contained in the preprocessed data

Usage

```
plot_lexical_dispersion(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  keywords = c("hello", "world"),
  return_data = FALSE,
  exclude_sm = FALSE,
  ...
)
```

Arguments

`data`	A 'WhatsApp' chatlog that was parsed with `parse_chat` using anonymize = FALSE or anonymize = "add".
`names`	A vector of author names that the plots will be restricted to.
`starttime`	Datetime that is used as the minimum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
`endtime`	Datetime that is used as the maximum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.

keywords	A vector of keywords to be displayed, default is c("hello","world").
return_data	Default is FALSE, returns data frame used for plotting when TRUE.
exclude_sm	If TRUE, excludes the 'WhatsApp' System Messages from the descriptive statistics. Default is FALSE.
...	Further arguments passed down to dispersion_plot.

Value

Lexical Dispersion plots for specified keywords

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_lexical_dispersion(data, keywords = c("auch"))
```

---

plot_links *Visualizing links in 'WhatsApp' chat logs*

Description

Visualizes the occurrence of links in a 'WhatsApp' chatlog

Usage

```
plot_links(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  use_domains = TRUE,
  exclude_long = 50,
  min_occur = 1,
  return_data = FALSE,
  link_vec = "all",
  plot = "bar",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chatlog that was parsed with parse_chat.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.

endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
use_domains	If TRUE, links are shortened to domains. This includes the inputs in link_vec. Default is TRUE.
exclude_long	Either NA or a numeric value. If numeric value is provided, removes all links/domains longer than x characters. Default is 50.
min_occur	The minimum number of occurrences a link has to have to be included in the visualization. Default is 1.
return_data	If TRUE, returns the subset data frame. Default is FALSE.
link_vec	A vector of links that the visualizations will be restricted to.
plot	The type of plot that should be returned Options are "heatmap", "cumsum", "bar" and "splitbar".
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

Plots and/or the subset data frame based on author names, datetime and emoji occurrence

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_links(data)
```

---

plot_locations *Plotting locations sent in 'WhatsApp' chat logs on maps*

Description

Plots the location data that is sent in the 'WhatsApp' chatlog on an auto-scaled map. Requires unanonymized 'Location' column in data

Usage

```
plot_locations(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  mapzoom = 5,
  return_data = FALSE,
  jitter_value = NA,
  jitter_seed = 12345,
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chatlog that was parsed with parse_chatwith anonymize= FALSE or anonymize = "add".
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
mapzoom	Value for zoom into the map passed down to get_map. Default value is 5. Higher zoom will auto-download more map files which can take a while.
return_data	If TRUE, returns a data frame of LatLon coordinates extracted from the chat for more elaborate plotting. Default is FALSE.
jitter_value	Amount of random jitter to add to the geolocations to hide exact locations. Default value is 0.01. Can be NA for exact locations.
jitter_seed	Seed for adding random jitter to coordinates. Passed to set.seed
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

Plots for geolocation and/or a data frame of latitude and longitude coordinates

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_locations(data)
```

---

plot_media *Visualizing media files in 'WhatsApp' chat logs if chats were exported with media files*

Description

Creates summary data frames or visualizations of sent media files or file types

Usage

```
plot_media(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
```

```
  use_filetype = TRUE,
  min_occur = 1,
  return_data = FALSE,
  media_vec = "all",
  plot = "bar",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chatlog that was parsed with parse_chat and was exported usng the "with media" option.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
use_filetype	If TRUE, shortens sent file attachments to file types.
min_occur	The minimum number of occurrences a media (type) has to have to be included in the visualization. Default is 1.
return_data	If TRUE, returns the subset data frame. Default is FALSE.
media_vec	A vector of media (types) that the visualizations will be restricted to.
plot	The type of plot that should be returned Options include "heatmap", "cumsum", "bar" and "splitbar".
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

Plots and/or the subset data frame based on author names, datetime and media (type) occurrence

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_media(data, plot = "heatmap")
```

plot_messages *Visualizing the number of sent messages per person in 'WhatsApp' chat logs*

Description

Plots summarizing the amount of messages per person

Usage

```
plot_messages(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  plot = "bar",
  return_data = FALSE,
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
plot	Type of plot to be returned, options are "bar", "cumsum", "heatmap" and "pie". Default is "bar".
return_data	If TRUE, returns the subset data frame. Default is FALSE.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

Plots summarizing the number of messages per person

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_messages(data)
```

plot_network *Visualizing the network of consecutive replies in 'WhatsApp' chat logs*

Description

Plots a network for replies between authors in chat logs. Each message is evaluated as a reply to the previous one.

Usage

```
plot_network(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  return_data = FALSE,
  collapse_sessions = FALSE,
  edgetype = "n",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chatlog that was parsed with parse_chat.
names	A vector of author names that the visualization will be restricted to. Non-listed authors will be removed.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
return_data	If TRUE, returns a data frame of subsequent interactions with senders and recipients. Default is FALSE.
collapse_sessions	Whether multiple subsequent messages by the same sender should be collapsed into one row. Default is FALSE.
edgetype	What type of content is displayed as an edge. Must be one of "TokCount","EmojiCount","SmilieCount","L or "n".
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

A network visualization of senders in 'WhatsApp' chat logs where each subsequent message is considered a reply to the previous one. Input will be ordered by TimeOrder column.

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_network(data)
```

`plot_replytimes` *Visualizing replytimes in 'WhatsApp' chat logs*

Description

Visualizes the reply times and reaction times to messages per author

Usage

```
plot_replytimes(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  return_data = FALSE,
  aggregate_sessions = TRUE,
  plot = "box",
  type = "replytime",
  exclude_sm = FALSE
)
```

Arguments

`data`	A 'WhatsApp' chat log that was parsed with `parse_chat`.
`names`	A vector of author names that the plots will be restricted to.
`starttime`	Datetime that is used as the minimum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
`endtime`	Datetime that is used as the maximum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
`return_data`	If TRUE, returns a data frame of response times extracted from the chat for more elaborate plotting. Default is FALSE.
`aggregate_sessions`	If TRUE, concurrent messages of the same author are aggregated into one session. Default is TRUE.
`plot`	Type of plot to be returned, options are "box" and "heatmap".
`type`	If "replytime", plots display how much time it takes authors to reply to previous message, if "reactiontime", plots display how much time it takes for authors to get responded to.
`exclude_sm`	If TRUE, excludes the 'WhatsApp' system messages from the data. Default is FALSE.

Value

Plots for Replytimes or Reactiontimes of authors. Input will be ordered by TimeOrder column.

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_replytimes(data)
```

`plot_smilies` *Visualize smilies used in 'WhatsApp' chat logs*

Description

Plots the smilies used in 'WhatsApp' chat logs by sender

Usage

```
plot_smilies(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  min_occur = 1,
  return_data = FALSE,
  smilie_vec = "all",
  plot = "bar",
  exclude_sm = FALSE
)
```

Arguments

`data`	A 'WhatsApp' chat log that was parsed with `parse_chat`.
`names`	A vector of author names that the plots will be restricted to.
`starttime`	Datetime that is used as the minimum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
`endtime`	Datetime that is used as the maximum boundary for exclusion. Is parsed with `as.POSIXct`. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
`min_occur`	The minimum number of occurrences a smiley has to have to be included in the visualization. Default is 1.
`return_data`	If TRUE, returns a data frame of smilies extracted from the chat for more elaborate plotting. Default is FALSE.
`smilie_vec`	A vector of smilies that the visualizations will be restricted to.

plot	The type of plot that should be returned. Options are "heatmap", "cumsum", "bar" and "splitbar".
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the data. Default is FALSE.

Value

Plots for distribution of smilies in 'WhatsApp' chats

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_smilies(data)
```

---

plot_tokens *Visualizing token distribution per person*

Description

Visualizing token distribution per person

Usage

```
plot_tokens(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  plot = "bar",
  return_data = FALSE,
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chatlog that was parsed with parse_chat.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
plot	The type of plot to be used. Options include "bar","box","violin" and "cumsum". Default is "bar". NA values will be removed before plotting. For "violin", Senders with less than 2 messages are removed.

return_data	If TRUE, returns the subsetted data frame. Default is FALSE.
exclude_sm	If TRUE, excludes the 'WhatsApp' System Messages from the descriptive statistics. Default is FALSE.

Value

Plots showcasing the distribution of tokens per person

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_tokens(data)
```

---

plot_tokens_over_time *Distribution of Tokens over time*

Description

Summarizes the distribution of user-generated tokens over time

Usage

```
plot_tokens_over_time(
  data,
  names = "all",
  names_col = "Sender",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  plot = "alltime",
  return_data = FALSE,
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat with parameters anonymize = FALSE or anonymize = "add".
names	A vector of author names that the plots will be restricted to.
names_col	A column indicated by a string that should be accessed to determine the names. Only needs to be changed when parse_chat used the parameter anon = "add" and the column "Anonymous" should be used. Default is "Sender".
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.

plot	Type of plot to be returned. Options are "year", "day", "hour", "heatmap" and "alltime". Default is "alltime".
return_data	If TRUE, returns the subset data frame. Default is FALSE.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

A summary of tokens over time. Input will be ordered by TimeOrder column.

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_tokens_over_time(data)
```

---

plot_wordcloud *Wordclouds for 'WhatsApp' chat logs*

Description

Creates a wordcloud by author for 'WhatsApp' chat logs. Requires raw message text to be present in data.

Usage

```
plot_wordcloud(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  remove_stops = TRUE,
  stop = "english",
  comparison = FALSE,
  return_data = FALSE,
  font_size = 10,
  min_occur = 5,
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat and anonymize = FALSE or anonymize = "add"
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.

endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
remove_stops	Either TRUE or FALSE, default is TRUE. Configures whether stopwords from stopwords are removed from the text strings.
stop	The language for stopword removal. Stopwords are taken from stopwords. Options are "english" and "german".
comparison	Must be TRUE or FALSE. If TRUE, will split up wordcloud by sender. Default is FALSE.
return_data	Will return the data frame used to create the plot if TRUE. Default is FALSE.
font_size	Size of the words in the wordcloud, passed to scale_size_area. Default is 10, a good starting value is 0.0125 * number of messages in data frame.
min_occur	Sets the minimum frequency a token must occur in the chat for it to be included in the plot. Default is 5.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from word clouds. Default is FALSE.

Value

A wordcloud plot per author for 'WhatsApp' chat logs

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
plot_wordcloud(data, comparison = TRUE, min_occur = 6)
```

---

summarize_chat *Basic 'WhatsApp' chat log Statistics*

Description

Creates a list of basic information about a single 'WhatsApp' chat log

Usage

```
summarize_chat(data, exclude_sm = FALSE)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

A list containing:

1) The number of messages in the chat
2) The number of tokens in the chat
3) The number of participants in the chat
4) The date of the first message
6) The date of the last message
7) The total duration of the chat
8) The number of system messages in the chat
9) The number of emoji in the chat
10) The number of smilies in the chat
11) The number of links in the chat
12) The number of media in the chat
12) The number of locations in the chat

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
summarize_chat(data)
```

---

summarize_tokens_per_person

Token Distributions for sent messages

Description

Summarizing the distribution of tokens for sent messages

Usage

```
summarize_tokens_per_person(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat.
names	A vector of author names that the plots will be restricted to.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.

endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed withas.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

A summary of tokens per message distribution per author

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
summarize_tokens_per_person(data)
```

---

tailor_chat *Restricting chat logs to certain authors or timeframes.*

Description

Excluding parts of the chat by senders or timestamps

Usage

```
tailor_chat(
  data,
  names = "all",
  starttime = "1960-01-01 00:00",
  endtime = "2200-01-01 00:00",
  exclude_sm = FALSE
)
```

Arguments

data	A 'WhatsApp' chat log that was parsed with parse_chat.
names	A vector of names that the output is restricted to. Messages from other non-contained authors are excluded.
starttime	Datetime that is used as the minimum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
endtime	Datetime that is used as the maximum boundary for exclusion. Is parsed with as.POSIXct. Standard format is "yyyy-mm-dd hh:mm". Is interpreted as UTC to be compatible with 'WhatsApp' timestamps.
exclude_sm	If TRUE, excludes the 'WhatsApp' system messages from the descriptive statistics. Default is FALSE.

Value

A dataframe that is restricted to the specified timeframe and authors

Examples

```
data <- readRDS(system.file("ParsedWhatsAppChat.rds", package = "WhatsR"))
tailor_chat(data, names = c("Mallory", "Alice"))
```

Index