Package ‘textreadr’

June 17, 2020

Title Read Text Documents into R

Version 1.0.2

Maintainer Tyler Rinker <`tyler.rinker@gmail.com`>

Description A small collection of convenience tools for reading text documents into R.

Depends R (>= 3.2.2)

Suggests tesseract, testthat

Imports antiword, curl, data.table, pdftools, readxl, rvest, striprtf,
textshape, tools, utils, xml2

License GPL-2

LazyData TRUE

RoxygenNote 7.1.0

BugReports `https://github.com/trinker/textreadr/issues?state=open`

URL `https://github.com/trinker/textreadr`

NeedsCompilation no

Author Tyler Rinker [aut, cre],
Bryan Goodrich [ctb],
Dason Kurkiewicz [ctb]

Repository CRAN

Date/Publication 2020-06-17 05:50:02 UTC

R topics documented:

`as_transcript` *Coerce Text toTranscripts Into R*

Description

Coerce text into a transcript.

Usage

```
as_transcript(
  text,
  person.regex = NULL,
  col.names = c("Person", "Dialogue"),
  text.var = NULL,
  merge.broke.tot = TRUE,
  header = FALSE,
  dash = "",
  ellipsis = "...",
  quote2bracket = FALSE,
  rm.empty.rows = TRUE,
  na = "",
  sep = NULL,
  skip = 0,
  comment.char = "",
  max.person.nchar = 20,
  ...
)
```

Arguments

`text`	Character string: if file is not supplied and this is, then data are read from the value of text. Notice that a literal string can be used to include (small) data sets within R code.
`person.regex`	A capturing regex describing what is a person portion of a string.

`col.names`	A character vector specifying the column names of the transcript columns.
`text.var`	A character string specifying the name of the text variable will ensure that variable is classed as character. If `NULL` `read_transcript()` attempts to guess the text.variable (dialogue).
`merge.broke.tot`	logical. If `TRUE` and if the file being read in is .docx with broken space between a single turn of talk read_transcript will attempt to merge these into a single turn of talk.
`header`	logical. If `TRUE` the file contains the names of the variables as its first line.
`dash`	A character string to replace the en and em dashes special characters (default is to remove).
`ellipsis`	A character string to replace the ellipsis special characters.
`quote2bracket`	logical. If `TRUE` replaces curly quotes with curly braces (default is `FALSE`). If `FALSE` curly quotes are removed.
`rm.empty.rows`	logical. If `TRUE` `read_transcript()` attempts to remove empty rows.
`na`	A character string to be interpreted as an `NA` value.
`sep`	The field separator character. Values on each line of the file are separated by this character. The default of `NULL` instructs `read_transcript()` to use a separator suitable for the file type being read in.
`skip`	Integer; the number of lines of the data file to skip before beginning to read data.
`comment.char`	A character vector of length one containing a single character or an empty string. Use `""` to turn off the interpretation of comments altogether.
`max.person.nchar`	The max number of characters long names are expected to be. This information is used to warn the user if a separator appears beyond this length in the text.
`...`	Further arguments to be passed to `utils::read.table()`, `readxl::read_excel()`, or `read_doc()`.

Value

Returns a dataframe of dialogue and people.

Examples

```
## EXAMPLE 1
as_transcript("34    The New York Times reports a lot of words here.
12    Greenwire reports a lot of words.
31    Only three words.
 2    The Financial Times reports a lot of words.
 9    Greenwire short.
13    The New York Times reports a lot of words again.",
    col.names = c("NO", "ARTICLE"), sep = "    ")

## EXAMPLE 2
as_transcript("34..    The New York Times reports a lot of words here.
12..    Greenwire reports a lot of words.
31..    Only three words.
 2..    The Financial Times reports a lot of words.
 9..    Greenwire short.
13..    The New York Times reports a lot of words again.",
    col.names = c("NO", "ARTICLE"), sep = "\\.\\.")

## EXAMPLE 3
as_transcript("JAKE The New York Times reports a lot of words here.
JIM Greenwire reports a lot of words.
JILL Only three words.
GRACE The Financial Times reports a lot of words.
JIM Greenwire short.
JILL The New York Times reports a lot of words again.",
   person.regex = '(^[A-Z]{3,})'
)
```

---

browse — *Open Directories & Files*

Description

Use the operating system defaults to open directories and files.

Usage

```
browse(x = ".")
```

Arguments

x	A vector (typically of length one) of paths to directories of files.

Note

This function is operating system and setting dependent. Results may not be consistent across operating systems. Depending upon the default programs for file types the results may vary as well. Some files may not be able to be opened.

Author(s)

Dason Kurkiewicz and Tyler Rinker tyler.rinker@gmail.com.

References

http://stackoverflow.com/q/12135732/1000343

Examples

```
## Not run:
browse()

## End(Not run)
```

`download` *Download Documents*

Description

This function enables downloading documents by wrapping `curl::curl_download()`.

Usage

```
download(url, loc = tempdir(), file.out = NULL, ...)
```

Arguments

`url`	The download url(s).
`loc`	Where to put the files.
`file.out`	Option vector of names matching `url`. If this is not given `download()` will try to create a name from `url`.
`...`	Other arguments passed to `curl::curl_download()`.

Value

Places a copy of the downloaded document in location specified and returns vector of the locations as string paths.

Examples

```
## Not run:
m <- download(
c('https://cran.r-project.org/web/packages/curl/curl.pdf',
"https://github.com/trinker/textreadr/raw/master/inst/docs/rl10075oralhistoryst002.pdf"),
)

m

## End(Not run)
```

`loop_counter` *Utilities for Looping to Read In Documents*

Description

`loop_counter` - A simple loop counter for tracking the progress of reading in a batch of files.

`base_name` - Like `base::basename` but doesn't choke on long paths.

`try_limit` - Limits the amount of try that an expression can run for. This works to limit how long an attempted read-in of a document may take. Most useful in a loop with a few very long running document read-ins (e.g., .pdf files that require **tesseract** package). Note that `max.time` can not stop a `system` call (as many read-in functions are essentially utilizing, but it can limit how many `system` calls are made. This means a .pdf with multiple **tesseract**) pages will only allow the first page to read-in before returning an error result. Note that this approach does not distinguish between errors running the `expr` and time-out errors.

Usage

```
loop_counter(i, total, file, ...)

base_name(path)

try_limit(
  expr,
  max.time = Inf,
  timeout.return = NULL,
  zero.length.return = "",
  silent = TRUE,
  ...
)
```

Arguments

`i`	Iteration of the loop.
`total`	Total number of iterations.
`file`	The file name of that iteration to print out.
`...`	ignored
`path`	A character vector, containing path names.
`expr`	An expression to run.
`max.time`	Max allotted elapsed run time in seconds.
`timeout.return`	Value to return for timeouts.
`zero.length.return`	Value to return for length zero expression evaluations.
`silent`	logical. If `TRUE` report of error messages.

Value

`loop_counter` - Prints loop information.

`base_name` - Returns just the basename of the path.

Examples

```
## Not run:
files <- dir(
    system.file("docs", package = "textreadr"),
    full.names = TRUE,
    recursive = TRUE,
    pattern = '\\.(R?md|Rd?$|txt|sql|html|pdf|doc|ppt|tex)'
)

max_wait <- 30
total <- length(files)
content <- vector(mode = "list", total)

for (i in seq_along(files)){

    loop_counter(i, total, base_name(files[i]))

    content[[i]] <- try_limit(
        textreadr::read_document(files[i]),
        max.time = max_wait,
        zero.length.return = NA
    )
}


sapply(content, is.null)
sapply(content, function(x) length(x) == 1 && is.na(x))
content

## End(Not run)
```

---

peek *Data Frame Viewing*

Description

peek - Convenience function to view all the columns of the head of a truncated `base::data.frame()`. peek invisibly returns `x`. This makes its use ideal in a **dplyr/magrittr** pipeline.

unpeek - Strips out class *textreadr* so that the entire `base::data.frame()` will be printed.

Usage

```
peek(x, n = 10, width = 20, strings.left = TRUE, ...)

unpeek(x)
```

Arguments

`x`	A `base::data.frame()` object.
`n`	Number of rows to display.
`width`	The width of the columns to be displayed.
`strings.left`	logical. If `TRUE` strings will be left aligned.
`...`	For internal use.

Details

By default **dplyr** does not print all columns of a **tibble**. This makes inspection of data difficult at times, particularly with text string data. `peek()` allows the user to see a truncated head for inspection purposes.

Value

Prints a truncated head but invisibly returns `x`.

See Also

`utils::head()`

Examples

```
peek(mtcars)
peek(presidential_debates_2012)
```

`presidential_debates_2012`

2012 U.S. Presidential Debates

Description

A dataset containing a cleaned version of all three presidential debates for the 2012 election.

Usage

```
data(presidential_debates_2012)
```

Format

A data frame with 2912 rows and 4 variables

Details

- person. The speaker
- tot. Turn of talk
- dialogue. The words spoken
- time. Variable indicating which of the three debates the dialogue is from

print.textreadr *Prints a textreadr Object*

Description

Prints a textreadr object

Usage

```
## S3 method for class 'textreadr'
print(x, width = 40, ...)
```

Arguments

`x`	A [base::data.frame()] textreadr object.
`width`	The width of the columns to be displayed.
`...`	Other arguments passed to `peek()`.

read_dir *Read In Multiple Files From a Directory*

Description

Read in multiple files from a directory and create a `base::data.frame()`.

Usage

```
read_dir(
  path,
  pattern = NULL,
  doc.col = "document",
  all.files = FALSE,
  recursive = FALSE,
  ignore.case = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`path`	Path to the directory.
`pattern`	An optional regular expression. Only file names which match the regular expression will be returned.
`doc.col`	A string naming the document columns (i.e., file names sans file extension).

`all.files`	Logical. If `FALSE`, only the names of visible files are returned. If `TRUE`, all file names will be returned.
`recursive`	Logical. Should the listing recurse into directories?
`ignore.case`	logical. If `TRUE` case in the `pattern` argument will be ignored.
`verbose`	Logical. Should Each iteration of the read-in be reported.
`...`	Other arguments passed to read_document functions.

Value

Returns a `base::data.frame()` with file names as a document column and content as a text column.

Examples

```
## Not run:
read_dir(system.file("docs/Maas2011/pos", package = "textreadr"))
read_dir(system.file("docs/Maas2011", package = "textreadr"), recursive=TRUE)

## End(Not run)
```

`read_dir_transcript` *Read In Multiple Transcript Files From a Directory*

Description

Read in multiple transcript files from a directory and create a `base::data.frame()`.

Usage

```
read_dir_transcript(
  path,
  col.names = c("Document", "Person", "Dialogue"),
  pattern = NULL,
  all.files = FALSE,
  recursive = FALSE,
  skip = 0,
  merge.broke.tot = TRUE,
  header = FALSE,
  dash = "",
  ellipsis = "...",
  quote2bracket = FALSE,
  rm.empty.rows = TRUE,
  na = "",
  sep = NULL,
  comment.char = "",
  max.person.nchar = 20,
  ignore.case = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`path`	Path to the directory.
`col.names`	A character vector specifying the column names of the transcript columns (document, person, dialogue).
`pattern`	An optional regular expression. Only file names which match the regular expression will be returned.
`all.files`	Logical. If `FALSE`, only the names of visible files are returned. If `TRUE`, all file names will be returned.
`recursive`	Logical. Should the listing recurse into directories?
`skip`	Integer; the number of lines of the data file to skip before beginning to read data.
`merge.broke.tot`	logical. If `TRUE` and if the file being read in is .docx with broken space between a single turn of talk read_transcript will attempt to merge these into a single turn of talk.
`header`	logical. If `TRUE` the file contains the names of the variables as its first line.
`dash`	A character string to replace the en and em dashes special characters (default is to remove).
`ellipsis`	A character string to replace the ellipsis special characters.
`quote2bracket`	logical. If `TRUE` replaces curly quotes with curly braces (default is `FALSE`). If `FALSE` curly quotes are removed.
`rm.empty.rows`	logical. If `TRUE` `read_transcript()` attempts to remove empty rows.
`na`	A character string to be interpreted as an `NA` value.
`sep`	The field separator character. Values on each line of the file are separated by this character. The default of `NULL` instructs `read_transcript()` to use a separator suitable for the file type being read in.
`comment.char`	A character vector of length one containing a single character or an empty string. Use `""` to turn off the interpretation of comments altogether.
`max.person.nchar`	The max number of characters long names are expected to be. This information is used to warn the user if a separator appears beyond this length in the text.
`ignore.case`	logical. If `TRUE` case in the `pattern` argument will be ignored.
`verbose`	Logical. Should Each iteration of the read-in be reported.
`...`	ignored.

Value

Returns a dataframe of documents, dialogue, and people.

See Also

read_transcript

Examples

```
skips <- c(0, 1, 1, 0, 0, 1)
path <- system.file("docs/transcripts", package = 'textreadr')
textreadr::peek(read_dir_transcript(path, skip = skips), Inf)

## Not run:
## with additional  cleaning
library(tidyverse, textshape, textclean)

path %>%
    read_dir_transcript(skip = skips) %>%
    textclean::filter_row("Person", "^\\[") %>%
    mutate(
        Person = stringi::stri_replace_all_regex(Person, "(^/\\s*)|(:\\s*$)", "") %>%
            trimws(),
        Dialogue = stringi::stri_replace_all_regex(Dialogue, "(^/\\s*)", "")
    ) %>%
    peek(Inf)

## End(Not run)
```

`read_doc` *Read in .doc Content*

Description

Read in the content from a .doc file using **antiword** via the **antiword** package.

Usage

```
read_doc(file, skip = 0, remove.empty = TRUE, trim = TRUE, format = FALSE, ...)
```

Arguments

`file`	The path to the .doc file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`format`	logical. If `TRUE` the output will keep doc formatting (e.g., bold, italics, underlined). This corresponds to the `-f` flag in **antiword**.
`...`	ignored.

Value

Returns a character vector.

Examples

```
## Not run:
x <- system.file("docs/Yasmine_Interview_Transcript.doc",
    package = "textreadr")
read_doc(x)

## End(Not run)
```

`read_document` *Generic Function to Read in a Document*

Description

Generic function to read in a .pdf, .txt, .html, .rtf, .docx, or .doc file.

Usage

```
read_document(
  file,
  skip = 0,
  remove.empty = TRUE,
  trim = TRUE,
  combine = FALSE,
  format = FALSE,
  ocr = TRUE,
  ...
)
```

Arguments

`file`	The path to the a .pdf, .txt, .html, .rtf, .docx, or .doc file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`combine`	logical. If `TRUE` the vector is concatenated into a single string via `textshape::combine()`.
`format`	For .doc files only. Logical. If `TRUE` the output will keep doc formatting (e.g., bold, italics, underlined). This corresponds to the `-f` flag in **antiword**.
`ocr`	logical. If `TRUE` .pdf documents with a non-text pull using pdftools::pdf_text() will be re-run using OCR via the `tesseract::ocr()` function. This will create temporary .png files and will require a much larger compute time.
`...`	Other arguments passed to `read_pdf()`, `read_html()`, `read_docx()`, `read_doc()`, or `base::readLines()`.

Value

Returns a `base::list()` of string `base::vector()`s.

Examples

```
## .pdf
pdf_doc <- system.file("docs/rl10075oralhistoryst002.pdf",
    package = "textreadr")
read_document(pdf_doc)

## .html
html_doc <- system.file("docs/textreadr_creed.html", package = "textreadr")
read_document(html_doc)

## .docx
docx_doc <- system.file("docs/Yasmine_Interview_Transcript.docx",
    package = "textreadr")
read_document(docx_doc)

## .doc
doc_doc <- system.file("docs/Yasmine_Interview_Transcript.doc",
    package = "textreadr")
read_document(doc_doc)

## .txt
txt_doc <- system.file('docs/textreadr_creed.txt', package = "textreadr")
read_document(txt_doc)

## .pptx
pptx_doc <- system.file('docs/Hello_World.pptx', package = "textreadr")
read_document(pptx_doc)

## .rtf
## Not run:
rtf_doc <- download(
    'https://raw.githubusercontent.com/trinker/textreadr/master/inst/docs/trans7.rtf'
)
read_document(rtf_doc)

## End(Not run)

## Not run:
## URLs
read_document('http://www.talkstats.com/index.php')

## End(Not run)
```

`read_docx` *Read in .docx Content*

Description

Read in the content from a .docx file.

Usage

```
read_docx(file, skip = 0, remove.empty = TRUE, trim = TRUE, ...)
```

Arguments

`file`	The path to the .docx file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`...`	ignored.

Value

Returns a character vector.

Author(s)

Bryan Goodrich and Tyler Rinker tyler.rinker@gmail.com.

Examples

```
## Not run:
url <- "https://github.com/trinker/textreadr/raw/master/inst/docs/Yasmine_Interview_Transcript.docx"
file <- download(url)
(txt <- read_docx(file))

## End(Not run)
```

`read_html` *Read in .html Content*

Description

Read in the content from a .html file. This is generalized, reading in all body text. For finer control the user should utilize the **xml2** and **rvest** packages.

Usage

```
read_html(file, skip = 0, remove.empty = TRUE, trim = TRUE, ...)

read_xml(file, skip = 0, remove.empty = TRUE, trim = TRUE, ...)
```

Arguments

`file`	The path to the .html file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`...`	Other arguments passed to xml2::read_html().

Value

Returns a character vector.

References

The xpath is taken from Tony Breyal's response on StackOverflow: `http://stackoverflow.com/questions/3195522/is-there-a-simple-way-in-r-to-extract-only-the-text-elements-of-an-html-page/3195926#3195926`

Examples

```
html_dat <- read_html(
    system.file("docs/textreadr_creed.html", package = "textreadr")
)

## Not run:
url <- "http://www.talkstats.com/index.php"
file <- download(url)
(txt <- read_html(url))
(txt <- read_html(file))

## End(Not run)
```

---

read_pdf *Read a Portable Document Format into R*

Description

A wrapper for pdftools::pdf_text() to read PDFs into **R**.

Usage

```
read_pdf(file, skip = 0, remove.empty = TRUE, trim = TRUE, ocr = TRUE, ...)
```

Arguments

`file`	A path to a PDF file.
`skip`	Integer; the number of lines of the data file to skip before beginning to read data.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`ocr`	logical. If `TRUE` documents with a non-text pull using pdftools::pdf_text() will be re-run using OCR via the `tesseract::ocr()` function. This will create temporary .png files and will require a much larger compute time.
`...`	Other arguments passed to pdftools::pdf_text().

Value

Returns a `base::data.frame()` with the page number (`page_id`), line number (`element_id`), and the `text`.

Note

A word of caution from Carl Witthoft" "Just a warning to others who may be hoping to extract data: PDF is a container, not a format. If the original document does not contain actual text, as opposed to bitmapped images of text or possibly even uglier things than I can imagine, nothing other than OCR can help you." If the reader has OCR needs the **tesseract** package, available on CRAN (`https://CRAN.R-project.org/package=tesseract`), is an "OCR engine with Unicode (UTF-8) support" and may be of use.

Examples

```
pdf_dat <- read_pdf(
    system.file("docs/rl10075oralhistoryst002.pdf", package = "textreadr")
)

pdf_dat_b <- read_pdf(
    system.file("docs/rl10075oralhistoryst002.pdf", package = "textreadr"),
    skip = 1
)

## Not run:
library(textshape)
system.file("docs/rl10075oralhistoryst002.pdf", package = "textreadr") %>%
    read_pdf(1) %>%
    `[[`('text') %>%
    head(-1) %>%
    textshape::combine() %>%
    gsub("([A-Z])( )([A-Z])", "\\1_\\3", .) %>%
    strsplit("(-| )(?=[A-Z_]+:)", perl=TRUE) %>%
    `[[`(1) %>%
    textshape::split_transcript()

## End(Not run)
## Not run:
## An image based .pdf file returns nothing.  Using the tesseract package as
## a backend for OCR overcomes this problem.

## Non-ocr
read_pdf(
    system.file("docs/McCune2002Choi2010.pdf", package = "textreadr"),
    ocr = FALSE
)

read_pdf(
    system.file("docs/McCune2002Choi2010.pdf", package = "textreadr"),
    ocr = TRUE
)

## End(Not run)
```

`read_pptx` *Read in .pptx Content*

Description

Read in the content from a .pptx file.

Usage

```
read_pptx(
  file,
  skip = 0,
  remove.empty = TRUE,
  trim = TRUE,
  include.notes = FALSE,
  ...
)
```

Arguments

`file`	The path to the .pptx file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`include.notes`	logical. If `TRUE` then slide notes are included.
`...`	ignored.

Value

Returns a `base::data.frame()` with the slide number (`slide_id`), line number (`element_id`), and the `text`.

Examples

```
## Not run:
url <- file.path("https://www.oclc.org/content/dam/research/presentations",
    "2019/111319-godby-NISO-What-Are-Entities-Matter.pptx")
file <- download(url)
(txt <- read_pptx(file))

pptx_doc <- system.file('docs/Hello_World.pptx', package = "textreadr")
read_pptx(pptx_doc)
read_pptx(pptx_doc, include.notes = TRUE)

## End(Not run)
```

---

read_rtf *Read a Rich Text Format Content*

Description

A wrapper for `striprtf::read_rtf()` to read RTFs

Usage

```
read_rtf(file, skip = 0, remove.empty = TRUE, trim = TRUE, ...)
```

Arguments

`file`	A path to a RTF file.
`skip`	The number of lines to skip.
`remove.empty`	logical. If `TRUE` empty elements in the vector are removed.
`trim`	logical. If `TRUE` the leading/training white space is removed.
`...`	Other arguments passed to `striprtf::read_rtf()`.

Value

Returns a character vector.

See Also

`striprtf::read_rtf()`

Examples

```
## Not run:
rtf_dat <- read_rtf(
    'https://raw.githubusercontent.com/trinker/textreadr/master/inst/docs/trans7.rtf'
)

## End(Not run)
```

read_transcript *Read Transcripts Into R*

Description

Read .docx, .doc, .rtf, .csv, .xlsx, .xlsx, or .txt transcript style files into R.

Usage

```
read_transcript(
  file,
  col.names = c("Person", "Dialogue"),
  text.var = NULL,
  merge.broke.tot = TRUE,
  header = FALSE,
  dash = "",
  ellipsis = "...",
  quote2bracket = FALSE,
  rm.empty.rows = TRUE,
  na = "",
  sep = NULL,
  skip = 0,
  text,
  comment.char = "",
  max.person.nchar = 20,
  ...
)
```

Arguments

`file`	The name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, `base::getwd()`.
`col.names`	A character vector specifying the column names of the transcript columns.
`text.var`	A character string specifying the name of the text variable will ensure that variable is classed as character. If `NULL` `read_transcript()` attempts to guess the text.variable (dialogue).
`merge.broke.tot`	logical. If `TRUE` and if the file being read in is .docx with broken space between a single turn of talk read_transcript will attempt to merge these into a single turn of talk.
`header`	logical. If `TRUE` the file contains the names of the variables as its first line.
`dash`	A character string to replace the en and em dashes special characters (default is to remove).
`ellipsis`	A character string to replace the ellipsis special characters.

`quote2bracket`	logical. If `TRUE` replaces curly quotes with curly braces (default is `FALSE`). If `FALSE` curly quotes are removed.
`rm.empty.rows`	logical. If `TRUE` `read_transcript()` attempts to remove empty rows.
`na`	A character string to be interpreted as an `NA` value.
`sep`	The field separator character. Values on each line of the file are separated by this character. The default of `NULL` instructs `read_transcript()` to use a separator suitable for the file type being read in.
`skip`	Integer; the number of lines of the data file to skip before beginning to read data.
`text`	Character string: if file is not supplied and this is, then data are read from the value of text. Notice that a literal string can be used to include (small) data sets within R code.
`comment.char`	A character vector of length one containing a single character or an empty string. Use `""` to turn off the interpretation of comments altogether.
`max.person.nchar`	The max number of characters long names are expected to be. This information is used to warn the user if a separator appears beyond this length in the text.
`...`	Further arguments to be passed to `utils::read.table()`, `readxl::read_excel()`, or `read_doc()`.

Value

Returns a dataframe of dialogue and people.

Warning

`read_transcript()` may contain errors if the file being read in is .docx. The researcher should carefully investigate each transcript for errors before further parsing the data.

Note

If a transcript is a .docx file read_transcript expects two columns (generally person and dialogue) with some sort of separator (default is colon separator). .doc files must be converted to .docx before reading in.

Author(s)

Bryan Goodrich and Tyler Rinker tyler.rinker@gmail.com.

References

`https://github.com/trinker/qdap/wiki/Reading-.docx-\%5BMS-Word\%5D-Transcripts-into-R`

Examples

```
(doc1 <- system.file("docs/trans1.docx", package = "textreadr"))
(doc2 <- system.file("docs/trans2.docx", package = "textreadr"))
(doc3 <- system.file("docs/trans3.docx", package = "textreadr"))
(doc4 <- system.file("docs/trans4.xlsx", package = "textreadr"))
(doc5 <- system.file("docs/trans5.xls", package = "textreadr"))
(doc6 <- system.file("docs/trans6.doc", package = "textreadr"))

dat1 <- read_transcript(doc1)
dat2 <- read_transcript(doc1, col.names = c("person", "dialogue"))

## read_transcript(doc2) #throws an error (need skip)
dat3 <- read_transcript(doc2, skip = 1)

## read_transcript(doc3, skip = 1) #incorrect read; wrong sep
dat4 <- read_transcript(doc3, sep = "-", skip = 1)

## xlsx/xls format
dat5 <- read_transcript(doc4)
dat6 <- read_transcript(doc5)

## MS doc format
## Not run:
dat7 <- read_transcript(doc6) ## need to skip Researcher
dat8 <- read_transcript(doc6, skip = 1)

## End(Not run)

## rtf format
## Not run:
rtf_doc <- download(
    'https://raw.githubusercontent.com/trinker/textreadr/master/inst/docs/trans7.rtf'
)
dat9 <- read_transcript(rtf_doc, skip = 1)

## End(Not run)

## text string input
trans <- "sam: Computer is fun. Not too fun.
greg: No it's not, it's dumb.
teacher: What should we do?
sam: You liar, it stinks!"

read_transcript(text=trans)

## Read in text specify spaces as sep
## EXAMPLE 1
read_transcript(text="34    The New York Times reports a lot of words here.
12    Greenwire reports a lot of words.
31    Only three words.
 2    The Financial Times reports a lot of words.
 9    Greenwire short.
13    The New York Times reports a lot of words again.",
    col.names = c("NO", "ARTICLE"), sep = "   ")

## EXAMPLE 2
read_transcript(text="34..    The New York Times reports a lot of words here.
12..    Greenwire reports a lot of words.
31..    Only three words.
 2..    The Financial Times reports a lot of words.
 9..    Greenwire short.
13..    The New York Times reports a lot of words again.",
    col.names = c("NO", "ARTICLE"), sep = "\\.\\.")

## Real Example
real_dat <- read_transcript(
    system.file("docs/Yasmine_Interview_Transcript.docx", package = "textreadr"),
    skip = 19
)
```

---

textreadr *Read Text Documents into R*

Description

A small collection of convenience tools for reading text documents into R.

---

un_zip *Unzip/Unzip Files*

Description

Unzip/untar files and return the location of exit directory. This is a convenience function (wrapper for `utils::unzip()`) to make the function more pipe-able. Additionally, the location of the unzip defaults to the directory containing the zip file.

Usage

```
un_zip(file, loc = dirname(file), ...)

un_tar(file, loc = dirname(file), ...)
```

Arguments

`file`	Path to the zip file.
`loc`	The output directory location.
`...`	Other arguments passed to `utils::unzip()`.

Value

Returns the path to where the zip file was unzipped to.

See Also

`utils::unzip()`

Examples

```
## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse)

dl_loc <- 'http://www.cs.uic.edu/~liub/FBS/CustomerReviewData.zip'  %>%
    download() %>%
    un_zip()


dir(dl_loc, pattern = '[Cc]ustomer')
dir(dl_loc, pattern = 'customer', full.names = TRUE)[1] %>%
    dir()

dir(dl_loc, pattern = 'customer', full.names = TRUE)[1] %>%
    dir(pattern = '\\.txt$', full.names = TRUE)

dir(dl_loc, pattern = 'customer', full.names = TRUE)[1] %>%
    read_dir()


dir(dl_loc, pattern = 'customer', full.names = TRUE)[1] %>%
    dir(pattern = '\\.txt$', full.names = TRUE) %>%
    `[`(1) %>%
    read_document()

## End(Not run)
```

Index