Package ‘textshape’

May 28, 2021

Title Tools for Reshaping Text
Version 1.7.3
Maintainer Tyler Rinker <tyler.rinker@gmail.com>
Description Tools that can be used to reshape and restructure text data.
Depends R (>= 3.4.0)
Imports data.table, slam, stats, stringi, utils
Suggests testthat
License GPL-2
LazyData TRUE
RoxygenNote 7.1.1
URL https://github.com/trinker/textshape
BugReports https://github.com/trinker/textshape/issues
Collate 'bind_list.R' 'bind_table.R' 'bind_vector.R' 'change_index.R'
  'cluster_matrix.R' 'column_to_rownames.R' 'combine.R'
  'duration.R' 'flatten.R' 'from_to.R' 'grab_index.R'
  'grab_match.R' 'intabulate.R' 'split_index.R' 'split_match.R'
  'split_match_regex_to_transcript.R' 'split_portion.R'
  'split_run.R' 'split_sentence.R' 'split_sentence_token.R'
  'split_speaker.R' 'split_token.R' 'split_transcript.R'
  'split_word.R' 'textshape-package.R' 'tidy_colol_dtm.R'
  'utils.R' 'tidy_dtm.R' 'tidy_list.R' 'tidy_matrix.R'
  'tidy_table.R' 'tidy_vector.R' 'unique_pairs.R' 'unnest_text.R'
NeedsCompilation no
Author Tyler Rinker [aut, cre],
  Joran Elias [ctb],
  Matthew Flickinger [ctb],
  Paul Foster [ctb]
Repository CRAN
Date/Publication 2021-05-28 17:50:03 UTC
R topics documented:

bind_list ........................................... 3
bind_table ........................................... 4
bind_vector ........................................... 4
change_index ......................................... 5
cluster_matrix ........................................ 6
column_to_rownames .................................. 7
combine ............................................... 8
DATA .................................................. 9
duration ............................................... 10
flatten ............................................... 11
from_to .............................................. 13
golden_rules ......................................... 14
grab_index ............................................ 15
grab_match ............................................ 16
hamlet ................................................. 17
mtabulate ............................................. 18
simple_dtm .......................................... 19
split_index .......................................... 20
split_match .......................................... 21
split_match_regex_to_transcript ....................... 22
split_portion ........................................ 24
split_run ............................................. 25
split_sentence ....................................... 26
split_sentence_token .................................. 27
split_speaker ........................................ 28
split_token .......................................... 29
split_transcript ...................................... 30
split_word ........................................... 31
textshape ............................................ 32
tidy_colo_tdm ........................................ 32
tidy_dtm ............................................. 34
tidy_list ............................................. 35
tidy_matrix .......................................... 36
tidy_table ........................................... 38
tidy_vector .......................................... 38
unique_pairs ......................................... 39
unnest_text .......................................... 40

Index 42
Row Bind a List of Named Dataframes or Vectors

Description

Deprecated, use tidy_list instead.

Usage

bind_list(x, id.name = "id", content.name = "content", ...)

Arguments

x          A named list of data.frames or vector.
id.name    The name to use for the column created from the list.
content.name The name to use for the column created from the list of vectors (only used if x is vector).
...        ignored.

Value

Returns a data.table with the names from the list as an id column.

Examples

## Not run:
bind_list(list(p=1:500, r=letters))
bind_list(list(p=mtcars, r=mtcars, z=mtcars, d=mtcars))

## 2015 Vice-Presidential Debates Example
if (!require("pacman")) install.packages("pacman")
pacman::p_load(rvest, magrittr, xml2)

debates <- c(
    wisconsin = "110908",
    boulder = "110906",
    california = "110756",
    ohio = "110489"
)

lapply(debates, function(x){
    xml2::read_html(paste0("http://www.presidency.ucsb.edu/ws/index.php?pid=", x)) %>%
    rvest::html_nodes("p") %>%
    rvest::html_text() %>%
    textshape::split_index(grep("^[A-Z]+:", .)) %>%
    textshape::combine() %>%
    textshape::split_transcript() %>%
    textshape::split_sentence())
bind_table

Column Bind a Table’s Values with Its Names

Description

Deprecated, use tidy_table instead.

Usage

bind_table(x, id.name = "id", content.name = "content", ...)

Arguments

x A table.

id.name The name to use for the column created from the table names.

content.name The name to use for the column created from the table values.

... ignored.

Value

Returns a data.table with the names from the table as an id column.

Examples

## Not run:

x <- table(sample(LETTERS[1:6], 1000, TRUE))

bind_table(x)

## End(Not run)

bind_vector

Column Bind an Atomic Vector’s Values with Its Names

Description

Deprecated, use tidy_vector instead.

Usage

bind_vector(x, id.name = "id", content.name = "content", ...)

**change_index**

### Arguments

- `x`: A named atomic `vector`.
- `id.name`: The name to use for the column created from the `vector names`.
- `content.name`: The name to use for the column created from the `vector` values.
- `...`: ignored.

### Value

Returns a `data.table` with the `names` from the `vector` as an id column.

### Examples

```r
## Not run:
x <- setNames(sample(LETTERS[1:6], 1000, TRUE), sample(state.name[1:5], 1000, TRUE))
bind_vector(x)
## End(Not run)
```

-----

## change_index

### Indexing of Changes in Runs

#### Description

Find the indices of changes in runs in a vector. This function pairs well with `split_index` and is the default for the indices in all `split_index` functions that act on atomic vectors.

#### Usage

```r
change_index(x, ...)
```

#### Arguments

- `x`: A vector.
- `...`: ignored.

#### Value

Returns a vector of integer indices of where a vector initially changes.

#### See Also

`split_index`
Examples

```r
set.seed(10)
(x <- sample(0:1, 20, TRUE))
change_index(x)
split_index(x, change_index(x))

(p_chng <- change_index(CO2[["Plant"]]))
split_index(CO2[["Plant"]], p_chng)
```

cluster_matrix

Reorder a Matrix Based on Hierarchical Clustering

Description

Reorder matrix rows, columns, or both via hierarchical clustering.

Usage

```r
cluster_matrix(x, dim = "both", method = "ward.D2", ...)
```

Arguments

- `x`: A matrix.
- `dim`: The dimension to reorder (cluster); must be set to "row", "col", or "both".
- `method`: The agglomeration method to be used (see `hclust`).
- `...`: ignored.

Value

Returns a reordered matrix.

See Also

- `hclust`

Examples

```r
cluster_matrix(mtcars)
cluster_matrix(mtcars, dim = 'row')
cluster_matrix(mtcars, dim = 'col')
```

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

```r
## plot heatmap w/o clustering
wo <- mtcars %>%
```
cor() %>%
tidy_matrix('car', 'var') %>%
ggplot(aes(var, car, fill = value)) +
  geom_tile() +
  scale_fill_viridis(name = expression(r[xy])) +
  theme(
    axis.text.y = element_text(size = 8) ,
    axis.text.x = element_text(
      size = 8,
      hjust = 1,
      vjust = 1,
      angle = 45
    ),
    legend.position = 'bottom',
    legend.key.height = grid::unit(.1, 'cm'),
    legend.key.width = grid::unit(.5, 'cm')
  ) +
  labs(subtitle = "With Out Clustering")

## plot heatmap w clustering
w <- mtcars %>%
cor() %>%
cluster_matrix() %>%
tidy_matrix('car', 'var') %>%
mutate(
  var = factor(var, levels = unique(var)),
  car = factor(car, levels = unique(car))
) %>%
group_by(var) %>%
ggplot(aes(var, car, fill = value)) +
  geom_tile() +
  scale_fill_viridis(name = expression(r[xy])) +
  theme(
    axis.text.y = element_text(size = 8) ,
    axis.text.x = element_text(
      size = 8,
      hjust = 1,
      vjust = 1,
      angle = 45
    ),
    legend.position = 'bottom',
    legend.key.height = grid::unit(.1, 'cm'),
    legend.key.width = grid::unit(.5, 'cm')
  ) +
  labs(subtitle = "With Clustering")

gridExtra::grid.arrange(wo, w, ncol = 2)

## End(Not run)
Description

Takes an existing column and uses it as rownames instead. This is useful when turning a `data.frame` into a `matrix`. Inspired by the `tibble` package’s `column_to_row` which is now deprecated if done on a `tibble` object. By coercing to a `data.frame` this problem is avoided.

Usage

```r
column_to_rownames(x, loc = 1)
```

Arguments

- `x`: An object that can be coerced to a `data.frame`.
- `loc`: The column location as either an integer or string index location. Must be unique row names.

Value

Returns a `data.frame` with the specified column moved to rownames.

Examples

```r
state_dat <- data.frame(state.name, state.area, state.center, state.division)
column_to_rownames(state_dat)
column_to_rownames(state_dat, 'state.name')
```

---

**combine**

*Combine Elements*

Description

Combine (`paste`) elements (`vectors`, `lists`, or `data.frames`) together with `collapse = TRUE`.

Usage

```r
combine(x, ...)
```

# Default S3 method:
```r
combine(x, fix.punctuation = TRUE, ...)
```

# S3 method for class 'data.frame'
```r
combine(x, text.var = TRUE, ...)
```
Arguments

- **x**: A `data.frame` or character vector with runs.
- `fix.punctuation`: logical. If TRUE, spaces before/after punctuation that should not be are removed (regex used: "(\s+?=[,.?!;:%-])|(?=\s+[$-])\s+").
- `text.var`: The name of the text variable.
- **...**: Ignored.

Value

Returns a vector (if given a list/vector) or an expanded `data.table` with elements pasted together.

Examples

```r
(x <- split_token(data[["state"]][1], FALSE))
combine(x)

(x2 <- split_token(data[["state"]], FALSE))
combine(x2)

(x3 <- split_sentence(data))
## without dropping the non-group variable column
combine(x3)

## Dropping the non-group variable column
combine(x3[, 1:5, with=FALSE])
```

Description

A fictitious dataset useful for small demonstrations.

Usage

```r
data(DATA)
```

Format

A data frame with 11 rows and 5 variables
Details

- person. Speaker
- sex. Gender
- adult. Dummy coded adult (0-no; 1-yes)
- state. Statement (dialogue)
- code. Dialogue coding scheme

---

**duration**

*Duration of Turns of Talk*

Description

duration - Calculate duration (start and end times) for duration of turns of talk measured in words.
startss - Calculate start times from a numeric vector.
ends - Calculate end times from a numeric vector.

Usage

duration(x, ...)

## Default S3 method:
duration(x, grouping.var = NULL, ...)

## S3 method for class 'data.frame'
duration(x, text.var = TRUE, ...)

## S3 method for class 'numeric'
duration(x, ...)

starts(x, ...)

ends(x, ...)

Arguments

- **x**
  A data.frame or character vector with a text variable or a numeric vector.
- **grouping.var**
  The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.
- **text.var**
  The name of the text variable. If TRUE duration tries to detect the text column.
- **...**
  Ignored.

Value

Returns a vector or data frame of starts and/or ends.
Examples

```r
(x <- c(
  "Mr. Brown comes! He says hello. I give him coffee.",
  "I'll go at 5 p.m. eastern time. Or somewhere in between!",
  "go there"
))
duration(x)
group <- c("A", "B", "A")
duration(x, group)

groups <- list(group1 = c("A", "B", "A"), group2 = c("red", "red", "green"))
duration(x, groups)

data(DATA)
duration(DATA)

## Larger data set
duration(hamlet)

## Integer values
x <- sample(1:10, 10)
duration(x)
starts(x)
ends(x)
```

---

**flatten**  
**Flatten a Nested List of Vectors Into a Single Tier List of Vectors**

**Description**

Flatten a named, nested list of atomic vectors to a single level using the concatenated list/atomic vector names as the names of the single tiered list.

**Usage**

```r
flatten(x, sep = "_", ...)  
```

**Arguments**

- `x`  
  A nested, named list of vectors.

- `sep`  
  A separator to use for the concatenation of the names from the nested list.

- `...`  
  Ignored.

**Value**

Returns a flattened list.
**Note**

The order of the list is sorted alphabetically. Pull requests for the option to return the original order would be appreciated.

**Author(s)**

StackOverflow user @Michael and Paul Foster and Tyler Rinker <tyler.rinker@gmail.com>.

**References**

https://stackoverflow.com/a/41882883/1000343
https://stackoverflow.com/a/48357114/1000343

**Examples**

```r
x <- list(
  urban = list(
    cars = c('volvo', 'ford'),
    food.dining = list(
      local.business = c('carls'),
      chain.business = c('dennys', 'panera')
    )
  ),
  rural = list(
    land.use = list(
      farming = list(
        dairy = c('cows'),
        vegie.plan = c('carrots')
      )
    ),
    social.rec = list(
      community.center = c('town.square')
    ),
    people.type = c('good', 'bad', 'in.between')
  ),
  other.locales = c('suburban'),
  missing = list(
    unknown = c(),
    known = c()
  ),
  end = c('wow')
)

x

flatten(x)
flatten(x, ' -> ')
```

```r
## Not run:
## dictionary from quanteda
require(quanteda); library(textreadr); library(dplyr)
mydict <- https://provalisresearch.com/Download/LaverGarry.zip" %>%
```
textreadr::download("") %>%
  unzip(exdir = tempdir()) %>%
  `\(\text{\texttt{\textbackslash{}}}\)`(1) %>%
  quanteda::dictionary(file = .)

mydict
flatten(as.list(mydict))

## End(Not run)

from_to

Prepare Discourse Data for Network Plotting

Description

`from_to` - Add the next speaker as the from variable in a to/from network data structure. Assumes that the flow of discourse is coming from person A to person B, or at the very least the talk is taken up by person B. Works by taking the vector of speakers and shifting everything down one and then adding a closing element.

`from_to_summarize` - A wrapper for `from_to.data.frame` that adds a `word.count` column and then combines duplicate rows.

Usage

```r
from_to(x, final = "End", ...) # Default S3 method:
from_to(x, final = "End", ...) # S3 method for class 'character'
from_to(x, final = "End", ...) # S3 method for class 'numeric'
from_to(x, from.var, final = "End", ...) # S3 method for class 'data.frame'
from_to_summarize(x, from.var, id.vars = NULL, text.var = TRUE, ...)
```

Arguments

- `x` A data form vector or `data.frame`.
- `final` The name of the closing element or node.
- `from.var` A character string naming the column to be considered the origin of the talk.
- `id.vars` The variables that correspond to the speaker or are attributes of the speaker (from variable).
text.var The name of the text variable. If TRUE duration tries to detect the text column.

... Ignored.

Value

Returns a vector (if given a vector) or an augmented data.table.

Examples

```r
from_to(DATA, 'person')
from_to_summarize(DATA, 'person')
from_to_summarize(DATA, 'person', c('sex', 'adult'))
## Not run:
if (!require("pacman")) install.packages("pacman"); library(pacman)
p_load(dplyr, geomnet, qdap, stringi, scales)
p_load_current_gh('trinker/textsahep')

dat <- from_to_summarize(DATA, 'person', c('sex', 'adult')) %>%
  mutate(words = rescale(word.count, c(.5, 1.5)))

dat %>%
  ggplot(aes(from_id = from, to_id = to)) +
  geom_net(
    aes(linewidth = words),
    layout.alg = "fruchtermanreingold",
    directed = TRUE,
    labelon = TRUE,
    size = 1,
    labelcolour = 'black',
    ecolour = "grey70",
    arrowsize = 1,
    curvature = .1
  ) +
  theme_net() +
  xlim(c(-0.05, 1.05))
## End(Not run)
```

---

golden_rules Sentence Boundary Disambiguation Edge Cases

Description

A slightly filtered dataset containing Dias’s sentence boundary disambiguation edge cases. This is a nested data set with the outcome column as a nested list of desired splits. The non-ASCII cases and spaced ellipsis examples have been removed.

Usage

data(golden_rules)
grab_index

Format

A data frame with 45 rows and 3 variables

Details

• Rule. The name of the rule to test
• Text. The testing text
• Outcome. The desired outcome of the sentence disambiguation

References


---

get_elements_matching_between_2_points

Description

Use regexes to get all the elements between two points.

Usage

```
grab_index(x, from = NULL, to = NULL, ...)
```

## S3 method for class 'character'
```
grab_index(x, from = NULL, to = NULL, ...)
```

## Default S3 method:
```
grab_index(x, from = NULL, to = NULL, ...)
```

## S3 method for class 'list'
```
grab_index(x, from = NULL, to = NULL, ...)
```

## S3 method for class 'data.frame'
```
grab_index(x, from = NULL, to = NULL, ...)
```

Arguments

- **x**
  A character vector, `data.frame`, or list.
- **from**
  An integer to start from (if NULL defaults to the first element/row).
- **to**
  A integer to get up to (if NULL defaults to the last element/row).
- **...**
  Ignored.
**grab_match**

Get Elements Matching Between 2 Points

**Description**

Use regexes to get all the elements between two points.

**Usage**

```r
grab_match(x, from = NULL, to = NULL, from.n = 1, to.n = 1, ...)
```

```r
## S3 method for class 'character'
grab_match(x, from = NULL, to = NULL, from.n = 1, to.n = 1, ...)
```

```r
## S3 method for class 'list'
grab_match(x, from = NULL, to = NULL, from.n = 1, to.n = 1, ...)
```

```r
## S3 method for class 'data.frame'
grab_match(
  x,
  from = NULL,
  to = NULL,
  from.n = 1,
  to.n = 1,
  text.var = TRUE,
  ...
)
```

**Arguments**

- **x**
  - A character vector, data.frame, or list.
- **from**
  - A regex to start getting from (if NULL defaults to the first element/row).
- **to**
  - A regex to get up to (if NULL defaults to the last element/row).
- **from.n**
  - If more than one element matches from this dictates which one should be used. Must be an integer up to the number of possible matches, 'first' (equal to 1), 'last' (the last match possible), or 'n' (the same as 'last').
hamlet

If more than one element matches to this dictates which one should be used. Must be an integer up to the number of possible matches, 'first' (equal to 1), 'last' (the last match possible), or 'n' (the same as 'last').

text.var The name of the text variable with matches. If TRUE grab_match tries to detect the text column.

... Other arguments passed to grep, most notable is ignore.case.

Value

Returns a subset of the original data set.

Examples

```
grab_match(DATA$state, from = 'dumb', to = 'liar')
grab_match(DATA$state, from = 'dumb')
grab_match(DATA$state, to = 'liar')
grab_match(DATA$state, from = 'no', to = 'the', ignore.case = TRUE)
grab_match(DATA$state, from = 'no', to = 'the', ignore.case = TRUE, from.n = 'first', to.n = 'last')
grab_match(as.list(DATA$state), from = 'dumb', to = 'liar')

## Data.frame: attempts to find text.var
grab_match(DATA, from = 'dumb', to = 'liar')
```

hamlet

Hamlet (Complete & Split by Sentence)

Description

A dataset containing the complete dialogue of Hamlet with turns of talk split into sentences.

Usage

```
data(hamlet)
```

Format

A data frame with 2007 rows and 7 variables

Details

- act. The act (akin to repeated measures)
- tot. The turn of talk
- scene. The scene (nested within an act)
- location. Location of the scene
- person. Character in the play
- died. Logical coded death variable if yes the character dies in the play
- dialogue. The spoken dialogue
mtabulate

Tabulate Frequency Counts for Multiple Vectors

Description
mtabulate - Similar to tabulate that works on multiple vectors.

as_list - Convert a count matrix to a named list of elements. The semantic inverse of mtabulate.

Usage
mtabulate(vects)
as_list(mat, nm = rownames(mat))

Arguments
vects A vector, list, or data.frame of named/unnamed vectors.
mat A matrix of counts.
nm A character vector of names to assign to the list.

Value
mtabulate - Returns a data.frame with columns equal to number of unique elements and the
number of rows equal to the the original length of the vector, list, or data.frame (length equals
number of columns in data.frame). If list of vectors is named these will be the rownames of the
dataframe.
as_list - Returns a list of elements.

Author(s)
Joran Elias and Tyler Rinker <tyler.rinker@gmail.com>.

References
https://stackoverflow.com/a/9961324/1000343

See Also
tabulate
Examples

```r
mtabulate(list(w=letters[1:10], x=letters[1:5], z=letters))
mtabulate(list(mtcars$cyl[1:10]))

## Dummy coding
mtabulate(mtcars$cyl[1:10])
mtabulate(CO2[, "Plant"])

dat <- data.frame(matrix(sample(c("A", "B"), 30, TRUE), ncol=3))
mtabulate(dat)
as_list(mtabulate(dat))
t(mtabulate(dat))
as_list(t(mtabulate(dat)))
```

---

**simple_dtm**  
*Simple DocumentTermMatrix*

---

**Description**

A dataset containing a simple `DocumentTermMatrix`.

**Usage**

```r
data(simple_dtm)
```

**Format**

A list with 6 elements

**Details**

- **i** The document locations
- **j** The term locations
- **v** The count of terms for that particular element position
- **nrow** The number of rows
- **ncol** The number of columns
- **dimnames** document and terms
split_index

Split Data Forms at Specified Indices

Description

Split data forms at specified integer indices.

Usage

```r
split_index(
  x,
  indices = if (is.atomic(x)) { NULL } else { change_index(x) },
  names = NULL,
  ...
)
```

## S3 method for class 'list'
```r
split_index(x, indices, names = NULL, ...)
```

## S3 method for class 'data.frame'
```r
split_index(x, indices, names = NULL, ...)
```

## S3 method for class 'matrix'
```r
split_index(x, indices, names = NULL, ...)
```

## S3 method for class 'numeric'
```r
split_index(x, indices = change_index(x), names = NULL, ...)
```

## S3 method for class 'factor'
```r
split_index(x, indices = change_index(x), names = NULL, ...)
```

## S3 method for class 'character'
```r
split_index(x, indices = change_index(x), names = NULL, ...)
```

## Default S3 method:
```r
split_index(x, indices = change_index(x), names = NULL, ...)
```

Arguments

- `x` A data form (list, vector, data.frame, matrix).
- `indices` A vector of integer indices to split at. If `indices` contains the index 1, it will be silently dropped. The default value when `x` evaluates to TRUE for `is.atomic` is to use `change_index(x)`.
- `names` Optional vector of names to give to the list elements.
- `...` Ignored.
split_match

Value

Returns of list of data forms broken at the indices.

Note

Two dimensional object will retain dimension (i.e., drop = FALSE is used).

See Also

change_index

Examples

## character

```r
split_index(LETTERS, c(4, 10, 16))
split_index(LETTERS, c(4, 10, 16), c("dog", "cat", "chicken", "rabbit"))
```

## numeric

```r
split_index(1:100, c(33, 66))
```

## factor

```r
(p_chng <- change_index(CO2[["Plant"]]))
split_index(CO2[["Plant"]], p_chng)
#'change_index' was unnecessary as it is the default of atomic vectors
split_index(CO2[["Plant"]])
```

## list

```r
split_index(as.list(LETTERS), c(4, 10, 16))
```

## data.frame

```r
(vs_change <- change_index(mtcars[["vs"]]))
split_index(mtcars, vs_change)
```

## matrix

```r
(mat <- matrix(1:50, nrow=10))
split_index(mat, c(3, 6, 10))
```

---

split_match | Split a Vector By Split Points

Description

split_match - Splits a vector into a list of vectors based on split points.

split_match_regex - split_match with regex = TRUE.

Usage

```r
split_match(x, split = ",", include = FALSE, regex = FALSE, ...)

split_match_regex(x, split = ",", include = FALSE, ...)
```
split_match_regex_to_transcript

Arguments

- **x**: A vector with split points.
- **split**: A vector of places (elements) to split on or a regular expression if regex argument is TRUE.
- **include**: An integer of 1 (split character(s) are not included in the output), 2 (split character(s) are included at the beginning of the output), or 3 (split character(s) are included at the end of the output).
- **regex**: logical. If TRUE regular expressions will be enabled for split argument.
- ... other arguments passed to grep and grepl.

Value

Returns a list of vectors.

Author(s)

Matthew Flickinger and Tyler Rinker <tyler.rinker@gmail.com>.

References

https://stackoverflow.com/a/24319217/1000343

Examples

```r
set.seed(15)
x <- sample(c("", LETTERS[1:10]), 25, TRUE, prob=c(.2, rep(.08, 10)))

split_match(x)
split_match(x, "C")
split_match(x, c("", "C"))

split_match(x, include = 0)
split_match(x, include = 1)
split_match(x, include = 2)

set.seed(15)
x <- sample(1:11, 25, TRUE, prob=c(.2, rep(.08, 10)))
split_match(x, 1)
```

---

split_match_regex_to_transcript

*Split Text by Regex Into a Transcript*

Description

A wrapper for `split_match_regex` and `textreadr`'s `as_transcript` to detect person variable, split the text into turns of talk, and convert to a data.frame with person and dialogue variables. There is a bit of cleansing that is closer to `as_transcript` than `split_transcript`. 
Usage

split_match_regex_to_transcript(
  x,
  person.regex = "^[A-Z]{3,}",
  col.names = c("Person", "Dialogue"),
  dash = "",
  ellipsis = "...",
  quote2bracket = FALSE,
  rm.empty.rows = TRUE,
  skip = 0,
  ...
)

Arguments

  x          A vector with split points.
  person.regex A vector of places (elements) to split on or a regular expression if regex argument is TRUE.
  col.names  A character vector specifying the column names of the transcript columns.
  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.
  skip       Integer; the number of lines of the data file to skip before beginning to read data.
  ...        ignored.

Value

Returns a data.frame of dialogue and people.

Examples

```r
# Not run:
system.file(
  "docs/Simpsons_Roasting_on_an_Open_Fire_Script.pdf",
  package = "textshape"
) %>%
textrad::read_document() %>%
  split_match_regex_to_transcript("^[A-Z]{3,}", skip = 2)
```

# End(Not run)
**split_portion**  
*Break Text Into Ordered Word Chunks*

**Description**  
Some visualizations and algorithms require text to be broken into chunks of ordered words. `split_portion` breaks text, optionally by grouping variables, into equal chunks. The chunk size can be specified by giving number of words to be in each chunk or the number of chunks.

**Usage**

```r
split_portion(
  text.var,  
  grouping.var = NULL,  
  n.words,  
  n.chunks,  
  as.string = TRUE,  
  rm.unequal = FALSE,  
  as.table = TRUE,  
  ...  
)
```

**Arguments**

- `text.var`  
  The text variable

- `grouping.var`  
  The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.

- `n.words`  
  An integer specifying the number of words in each chunk (must specify n.chunks or n.words).

- `n.chunks`  
  An integer specifying the number of chunks (must specify n.chunks or n.words).

- `as.string`  
  logical. If TRUE the chunks are returned as a single string. If FALSE the chunks are returned as a vector of single words.

- `rm.unequal`  
  logical. If TRUE final chunks that are unequal in length to the other chunks are removed.

- `as.table`  
  logical. If TRUE the list output is coerced to `data.table` or `tibble`.

- `...`  
  Ignored.

**Value**

Returns a list or `data.table` of text chunks.
Examples

with(DATA, split_portion(state, n.chunks = 10))
with(DATA, split_portion(state, n.words = 10))
with(DATA, split_portion(state, n.chunks = 10, as.string=FALSE))
with(DATA, split_portion(state, n.chunks = 10, rm.unequal=TRUE))
with(DATA, split_portion(state, person, n.chunks = 10))
with(DATA, split_portion(state, list(sex, adult), n.words = 10))
with(DATA, split_portion(state, person, n.words = 10, rm.unequal=TRUE))

## Bigger data
with(hamlet, split_portion(dialogue, person, n.chunks = 10))
with(hamlet, split_portion(dialogue, list(act, scene, person), n.chunks = 10))
with(hamlet, split_portion(dialogue, person, n.words = 300))
with(hamlet, split_portion(dialogue, list(act, scene, person), n.words = 300))

split_run

Split Runs

Description

Split runs of consecutive characters.

Usage

split_run(x, ...)

## Default S3 method:
split_run(x, ...)

## S3 method for class 'data.frame'
split_run(x, text.var = TRUE, ...)

Arguments

x  A data.frame or character vector with runs.

text.var  The name of the text variable with runs. If TRUE split_word tries to detect the text column with runs.

...  Ignored.

Value

Returns a list of vectors of runs or an expanded data.table with runs split apart.
Examples

```r
x1 <- c(
  "1233444555666666", 
  NA, 
  "abbccdddeeeefffff", 
  "sddfg", 
  "1112222333"
)

x <- c(rep(x1, 2), ">>???,,,,....::::;["

split_run(x)
```

```r
DATA["run.col"] <- x
split_run(DATA, "run.col")
```

---

### split_sentence

**Split Sentences**

**Description**

Split sentences.

**Usage**

```r
split_sentence(x, ...)
```

---

**Arguments**

- `x` A `data.frame` or character vector with sentences.
- `text.var` The name of the text variable. If `TRUE` `split_sentence` tries to detect the column with sentences.
- `...` Ignored.

**Value**

Returns a list of vectors of sentences or a expanded `data.frame` with sentences split apart.
Examples

(x <- c(paste0(
    "Mr. Brown comes! He says hello. I give him coffee. I will ",
    "go at 5 p.m. eastern time. Or somewhere in between! go there"
),
    paste0(
    "Marvin K. Mooney Will You Please Go Now!", "The time has come."
    "The time has come. The time is now. Just go. Go. GO!",
    "I don't care how."
)))
split_sentence(x)

data(DATA)
split_sentence(DATA)

## Not run:
## Kevin S. Dias' sentence boundary disambiguation test set
data(golden_rules)
library(magrittr)
golden_rules
    split_sentence(Text)

## End(Not run)

split_sentence_token Split Sentences & Tokens

Description

Split sentences and tokens.

Usage

split_sentence_token(x, ...)

## Default S3 method:
split_sentence_token(x, lower = TRUE, ...)

## S3 method for class 'data.frame'
split_sentence_token(x, text.var = TRUE, lower = TRUE, ...)

Arguments

x A data.frame or character vector with sentences.
lower logical. If TRUE the words are converted to lower case.
text.var The name of the text variable. If TRUE split_sentence_token tries to detect the column with sentences.
... Ignored.
split_speaker

Value

Returns a list of vectors of sentences or a expanded data.frame with sentences split apart.

Examples

(x <- c(paste0(
    "Mr. Brown comes! He says hello. i give him coffee. i will ",
    "go at 5 p. m. eastern time. Or somewhere in between!go there"
),
paste0(
    "Marvin K. Mooney Will You Please Go Now!", "The time has come."
    "The time has come. The time is now. Just go. Go. GO!",
    "I don't care how."
)))

split_sentence_token(x)

data(DATA)

split_sentence_token(DATA)

## Not run:
## Kevin S. Dias' sentence boundary disambiguation test set
data(golden_rules)
library(magrittr)

golden_rules %$%
    split_sentence_token(Text)

## End(Not run)

split_speaker

Break and Stretch if Multiple Persons per Cell

Description

Look for cells with multiple people and create separate rows for each person.

Usage

split_speaker(dataframe, speaker.var = 1, sep = c("and", ",", ",", ",", )

Arguments

dataframe A dataframe that contains the person variable.
speaker.var The person variable to be stretched.
sep The separator(s) to search for and break on. Default is: c("and", ",", ",", )
... Ignored.
Value

Returns an expanded dataframe with person variable stretched and accompanying rows repeated.

Examples

```r
## Not run:
DATA$person <- as.character(DATA$person)
DATA$person[c(1, 4, 6)] <- c("greg, sally, & sam",
  "greg, sally", "sam and sally")

split_speaker(DATA)

DATA$person[c(1, 4, 6)] <- c("greg_sally_sam",
  "greg.sally", "sam; sally")

split_speaker(DATA, sep = c(",", ",", ";"))

DATA <- textshape::DATA #reset DATA

## End(Not run)
```

Description

Split tokens.

Usage

`split_token(x, ...)`

## Default S3 method:
`split_token(x, lower = TRUE, ...)`

## S3 method for class 'data.frame'
`split_token(x, text.var = TRUE, lower = TRUE, ...)`

Arguments

- `x`: A `data.frame` or character vector with tokens.
- `lower`: logical. If TRUE the words are converted to lower case.
- `text.var`: The name of the text variable. If TRUE `split_token` tries to detect the text column with tokens.
- `...`: Ignored.
Value

Returns a list of vectors of tokens or an expanded \texttt{data.table} with tokens split apart.

Examples

```r
(x <- c(
    "Mr. Brown comes! He says hello. I give him coffee.",
    "I'll go at 5 p.m. eastern time. Or somewhere in between!",
    "go there"
))
split_token(x)
split_token(x, lower=FALSE)
```

```r
data(DATA)
split_token(DATA)
split_token(DATA, lower=FALSE)
```

```r
## Larger data set
split_token(hamlet)
```

---

**split_transcript**

\textit{Split a Transcript Style Vector on Delimiter & Coerce to Dataframe}

Description

Split a transcript style vector (e.g., c("greg: Who me","sarah: yes you!")) into a name and dialogue vector that is coerced to a \texttt{data.table}. Leading/trailing white space in the columns is stripped out.

Usage

```r
split_transcript(
  x,
  delim = ":",
  colnames = c("person", "dialogue"),
  max.delim = 15,
  ...)
```

Arguments

- \texttt{x} A transcript style vector (e.g., c("greg: Who me","sarah: yes you!")).
- \texttt{delim} The delimiter to split on.
- \texttt{colnames} The column names to use for the \texttt{data.table} output.
- \texttt{max.delim} An integer stating how many characters may come before a delimiter is found. This is useful for the case when a colon is the delimiter but time stamps are also found in the text.
- \texttt{...} Ignored.
Value

Returns a 2 column data.table.

Examples

split_transcript(c("greg: Who me", "sarah: yes you!"))

## Not run:
## 2015 Vice-Presidential Debates Example
if (!require("pacman")) install.packages("pacman")
pacman::p_load(rvest, magrittr, xml2)

debates <- c(
    wisconsin = "110908",
    boulder = "110906",
    california = "110756",
    ohio = "110489"
)
lapply(debates, function(x){
    xml2::read_html(paste0("http://www.presidency.ucsb.edu/ws/index.php?pid=", x)) %>%
    rvest::html_nodes("p") %>%
    rvest::html_text() %>%
    textshape::split_index(grep("^[A-Z]+:", .)) %>%
    textshape::combine() %>%
    textshape::split_transcript() %>%
    textshape::split_sentence()
})

## End(Not run)

---

split_word

Split Words

Description

Split words.

Usage

split_word(x, ...)

## Default S3 method:
.split_word(x, lower = TRUE, ...)

## S3 method for class 'data.frame'
.split_word(x, text.var = TRUE, lower = TRUE, ...)
Arguments

x A data.frame or character vector with words.
lower logical. If TRUE the words are converted to lower case.
text.var The name of the text variable. If TRUE split_word tries to detect the text column with words.
... Ignored.

Value

Returns a list of vectors of words or an expanded data.table with words split apart.

Examples

(x <- c("Mr. Brown comes! He says hello. I give him coffee.", "I'll go at 5 p.m. eastern time. Or somewhere in between!", "go there")
split_word(x)
split_word(x, lower=FALSE)

data(DATA)
split_word(DATA)
split_word(DATA, lower=FALSE)

## Larger data set
split_word(hamlet)

---

**textshape**

Tools for Reshaping Text

Description

Tools that can be used to reshape and restructure text data.

---

**tidy_colo_tdm**

Convert a DocumentTermMatrix/TermDocumentMatrix into Collocating Words in Tidy Form

Description

Converts non-zero elements of a DocumentTermMatrix/TermDocumentMatrix into a tidy data set made of collocating words.
tidy_colo_tdm

Usage

tidy_colo_tdm(x, ...)
tidy_colo_dtm(x, ...)

Arguments

x
A DocumentTermMatrix/TermDocumentMatrix.
...
Ignored.

Value

Returns a tidied data.frame.

See Also

unique_pairs

Examples

data(simple_dtm)
tidied <- tidy_colo_dtm(simple_dtm)
tidied
unique_pairs(tidied)

## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load_current_gh("trinker/gofastr", "trinker/lexicon")
pacman::p_load(tidyverse, magrittr, ggstance)

my_dtm <- with(
  presidential_debates_2012,
  q_dtm(dialogue, paste(time, tot, sep = ",")))
tidy_colo_dtm(my_dtm) %>%
tbl_df() %>%
  filter(!term_1 %in% c('i', lexicon::sw_onix) &
    !term_2 %in% lexicon::sw_onix
  ) %>%
  filter(term_1 != term_2) %>%
  unique_pairs() %>%
  filter(n > 15) %>%
  complete(term_1, term_2, fill = list(n = 0)) %>%
  ggplot(aes(x = term_1, y = term_2, fill = n)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

## End(Not run)
tidy_dtm

Convert a DocumentTermMatrix/TermDocumentMatrix into Tidy Form

Description

Converts non-zero elements of a DocumentTermMatrix/TermDocumentMatrix into a tidy data set.

Usage

```
tidy_dtm(x, ...)
tidy_tdm(x, ...)
```

Arguments

- `x` A DocumentTermMatrix/TermDocumentMatrix.
- `...` ignored.

Value

Returns a tidied data.frame.

Examples

```
data(simple_dtm)
tidy_dtm(simple_dtm)
```

```
## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load_current_gh('trinker/gofastr')
pacman::p_load(tidyverse, magrittr, ggstance)

my_dtm <- with(
  presidential_debates_2012,
  q_dtm(dialogue, paste(time, tot, sep = "_"))
)

 tidy_dtm(my_dtm) %>%
  tidyr::extract(
    col = doc,
    into = c("time", "turn", "sentence"),
    regex = "([^d]+)(\d+)/(\d+)(\d+)"
  ) %>%
  mutate(
    time = as.numeric(time),
    turn = as.numeric(turn),
    sentence = as.numeric(sentence)
)```
tidy_list

) %>%
  tbl_df() %>%
  print() %>%
  group_by(time, term) %>%
  summarize(n = sum(n)) %>%
  group_by(time) %>%
  arrange(desc(n)) %>%
  slice(1:10) %>%
  ungroup() %>%
  mutate(
    term = factor(paste(term, time, sep = "__"),
      levels = rev(paste(term, time, sep = "__")))
  ) %>%
  ggplot(aes(x = n, y = term)) +
  geom_barh(stat='identity') +
  facet_wrap(~time, ncol=2, scales = 'free_y') +
  scale_y_discrete(labels = function(x) gsub("__.+$", ",", x))
## End(Not run)

tidy_list

Tidy a List of Named Dataframes or Named Vectors or Vectors

Description

rbind a named list of data.frames or vectors to output a single data.frame with the names from the list as an id column.

Usage

tidy_list(
  x,
  id.name = "id",
  content.name = "content",
  content.attribute.name = "attribute",
  ...
)

Arguments

x
  A named list of data.frames or vector.

id.name
  The name to use for the column created from the list.

content.name
  The name to use for the column created from the list of vectors (only used if x is vector).

content.attribute.name
  The name to use for the column created from the list of names given to the vectors (only used if x is named vector).

... Ignored.
Value

Returns a `data.table` with the names from the list as an id column.

Examples

```r
 tidy_list(list(p=1:500, r=letters))
 tidy_list(list(p=mtcars, r=mtcars, z=mtcars, d=mtcars))

 x <- list(
   a = setNames(c(1:4), LETTERS[1:4]),
   b = setNames(c(7:9), LETTERS[7:9]),
   c = setNames(c(10:15), LETTERS[10:15]),
   d = c(x=4, y=6, 4),
   e = setNames(1:10, sample(state.abb, 10, TRUE)),
   f = setNames(1:10, sample(month.abb, 10, TRUE))
 )
 tidy_list(x)

## Not run:
## 2015 Vice-Presidential Debates Example
if (!require("pacman")) install.packages("pacman")
pacman::p_load(rvest, magrittr, xml2)

debates <- c(
   wisconsin = "110908",
   boulder = "110906",
   california = "110756",
   ohio = "110489"
)

lapply(debates, function(x){
   paste0("http://www.presidency.ucsb.edu/ws/index.php?pid=", x) %>%
xml2::read_html() %>%
rvest::html_nodes("p") %>%
rvest::html_text() %>%
textshape::split_index(grep("^[A-Z]+:", .)) %>%
textshape::combine() %>%
textshape::split_transcript() %>%
textshape::split_sentence()
}) %>%
textshape::tidy_list("location")

## End(Not run)
```
tidy_matrix

Description

tidy_matrix - Converts matrices into a tidy data set. Essentially, a stacking of the matrix columns and repeating row/column names as necessary.

tidy_adjacency_matrix - A wrapper for tidy_matrix with the row.name, col.name, & value.name all set to "from", "to", & "n", assuming preparation for network analysis.

Usage

tidy_matrix(x, row.name = "row", col.name = "col", value.name = "value", ...)
tidy_adjacency_matrix(x, ...)

Arguments

x A matrix.
row.name A string to use for the row names that are now a column.
col.name A string to use for the column names that are now a column.
value.name A string to use for the values that are now a column.
... ignored.

Value

Returns a tidied data.frame.

Examples

mat <- matrix(1:16, nrow = 4,
  dimnames = list(LETTERS[1:4], LETTERS[23:26]))

mat
tidy_matrix(mat)

data(simple_dtm)
tidy_matrix(as.matrix(simple_dtm), 'doc', 'term', 'n')

X <- as.matrix(simple_dtm[1:10, 1:10])
tidy_adjacency_matrix(crossprod(X))
tidy_adjacency_matrix(crossprod(t(X)))
tidy_table

**Tidy a Table: Bind Its Values with Its Names**

**Description**

`cbind` a table's values with its names to form id (from the names) and content columns.

**Usage**

```
tidy_table(x, id.name = "id", content.name = "content", ...)  
```

**Arguments**

- **x**: A table.
- **id.name**: The name to use for the column created from the table names.
- **content.name**: The name to use for the column created from the table values.
- **...**: ignored.

**Value**

Returns a data.table with the names from the table as an id column.

**Examples**

```
x <- table(sample(LETTERS[1:6], 1000, TRUE))
tidy_table(x)  
```

tidy_vector

**Tidy a Named Atomic Vector: Bind Its Values with Its Names**

**Description**

`cbind` a named atomic vector's values with its names to form id (from the names) and content columns.

**Usage**

```
tidy_vector(x, id.name = "id", content.name = "content", ...)  
```

**Arguments**

- **x**: A named atomic vector.
- **id.name**: The name to use for the column created from the vector names.
- **content.name**: The name to use for the column created from the vector values.
- **...**: ignored.
**unique_pairs**

**Value**

Returns a data.table with the names from the vector as an id column.

**Examples**

```r
x <- setNames(sample(LETTERS[1:6], 1000, TRUE), sample(state.name[1:5], 1000, TRUE))
tidy_vector(x)
```

---

**unique_pairs**  
*Extract Only Unique Pairs of Collocating Words in tidy_colo_dtm*

**Description**

tidy_colo_dtm utilizes the entire matrix to generate the tidied data.frame. This means that the upper and lower triangles are used redundantly. This function eliminates this redundancy by dropping one set of the pairs from a tidied data.frame.

**Usage**

```r
unique_pairs(x, col1 = "term_1", col2 = "term_2", ...)
```

## Default S3 method:

```r
unique_pairs(x, col1 = "term_1", col2 = "term_2", ...)
```

## S3 method for class 'data.table'

```r
unique_pairs(x, col1 = "term_1", col2 = "term_2", ...)
```

**Arguments**

- `x` A data.frame with two columns that contain redundant pairs.
- `col1` A string naming column 1.
- `col2` A string naming column 2.
- `...` ignored.

**Value**

Returns a filtered data.frame.

**See Also**

tidy_colo_dtm
Examples

```r
dat <- data.frame(
  term_1 = LETTERS[1:10],
  term_2 = LETTERS[10:1],
  stringsAsFactors = FALSE
)

unique_pairs(dat)
```

---

**unnest_text**

*Un-nest Nested Text Columns*

**Description**

Un-nest nested text columns in a data.frame. Attempts to locate the nested text column without specifying.

**Usage**

```r
unnest_text(dataframe, column, integer.rownames = TRUE, ...)
```

**Arguments**

- `dataframe` A dataframe object.
- `column` Column name to search for markers/terms.
- `integer.rownames` logical. If TRUE then the rownames are numbered 1 through number of rows, otherwise the original row number is retained followed by a period and the element number from the list.
- `...` ignored.

**Value**

Returns an un-nested data.frame.

**Examples**

```r
dat <- DATA

## Add a nested/list text column
dat$splitted <- lapply(dat$state, function(x) {
  unlist(strsplit(x, '\s+'), perl = TRUE))
}

unnest_text(dat)
unnest_text(dat, integer.rownames = FALSE)

## Add a second nested integer column`
dat$d <- lapply(dat$split, nchar)
## Not run:
unnest_text(dat) # causes error, must supply column explicitly

## End(Not run)
unnest_text(dat, 'split')

## As a data.table
library(data.table)
dt_dat <- data.table::as.data.table(data.table::copy(dat))
unnest_text(dt_dat, 'split')
## Not run:
unnest_text(dt_dat, 'd')

## End(Not run)

## Not run:
## As a tibble
library(tibble)
t_dat <- tibble::as_tibble(dat)
unnest_text(t_dat, 'split')

## End(Not run)
Index

* chunks
  split_portion, 24
* datasets
  DATA, 9
  golden_rules, 14
  hamlet, 17
  simple_dtm, 19
* frequency
  mtabulate, 18
* group
  split_portion, 24
* tabulate
  mtabulate, 18
* text
  split_portion, 24

as_list (mtabulate), 18
bind_list, 3
bind_table, 4
bind_vector, 4
cbind, 38
change_index, 5, 20, 21
cluster_matrix, 6
column_to_rownames, 7
combine, 8

DATA, 9
data.frame, 3, 8–10, 15, 16, 18, 25–29, 32, 35, 39
data.table, 3–5, 9, 14, 24, 25, 30–32, 36, 38, 39
DocumentTermMatrix, 19, 32–34
duration, 10

ends (duration), 10
flatten, 11
from_to, 13
from_to_summarize (from_to), 13
golden_rules, 14
grab_index, 15
golden_rules, 14
hamlet, 17
hclust, 6
is.atomic, 20
list, 3, 8, 18, 35, 36
matrix, 8
mtabulate, 18
names, 3–5, 35, 36, 38, 39

package_textshape (textshape), 32
paste, 8
rbind, 35
read_transcript, 23
simple_dtm, 19
split_index, 5, 20
split_match, 21
split_match_regex, 22
split_match_regex (split_match), 21
split_match_regex_to_transcript, 22
split_portion, 24
split_run, 25
split_sentence, 26
split_sentence_token, 27
split_speaker, 28
split_token, 29
split_transcript, 22, 30
split_word, 31
starts (duration), 10
table, 4, 38
INDEX

tabulate, 18
TermDocumentMatrix, 32–34
textshape, 32
tidy_adjacency_matrix (tidy_matrix), 36
tidy_colo_dtm, 39
tidy_colo_dtm (tidy_colo_tdm), 32
tidy_colo_tdm, 32
tidy_dtm, 34
tidy_list, 3, 35
tidy_matrix, 36
tidy_table, 4, 38
tidy_tdm (tidy_dtm), 34
tidy_vector, 4, 38
unique_pairs, 33, 39
unnest_text, 40

vector, 3, 5, 8, 18, 35, 38, 39