Package 'justifier'

June 3, 2019

Title Human and Machine-Readable Justifications and Justified Decisions Based on 'YAML'

Version 0.1.0

Maintainer Gjalt-Jorn Ygram Peters <gjalt-jorn@behaviorchange.eu>

Description Leverages the 'yum' package to implement a 'YAML' ('YAML Ain't Markup Language', a human friendly standard for data serialization; see <https:yaml.org>) standard for documenting justifications, such as for decisions taken during the planning, execution and analysis of a study or during the development of a behavior change intervention as illustrated by Marques & Peters (2019) <doi:10.17605/osf.io/ndxha>. These justifications are both human- and machine-readable, facilitating efficient extraction and organisation.

License GPL (>= 2)

Encoding UTF-8

LazyData true

URL https://r-packages.gitlab.io/justifier

BugReports https://gitlab.com/r-packages/justifier/issues

Suggests covr, knitr, rmarkdown, testthat

Imports data.tree (>= 0.7.8), DiagrammeR (>= 1.0.0), purrr (>= 0.3.0), ufs (>= 0.2.0), yum (>= 0.0.1)

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Gjalt-Jorn Ygram Peters [aut, cre]

Repository CRAN

Date/Publication 2019-06-03 12:30:18 UTC

R topics documented:

`apply_graph_theme` *Apply multiple DiagrammeR global graph attributes*

Description

Apply multiple DiagrammeR global graph attributes

Usage

```
apply_graph_theme(graph, ...)
```

Arguments

`graph`	The DiagrammeR::DiagrammeR graph to apply the attributes to.
`...`	One or more character vectors of length three, where the first element is the attribute, the second the value, and the third, the attribute type (`graph`, `node`, or `edge`).

Value

The DiagrammeR::DiagrammeR graph.

Examples

```
exampleJustifier <- '
---
assertion:
  -
    id: assertion_id
    label: "An assertion"
decision:
  -
    id: decision_id
    label: "A decision"
    justification:
      -
        id: justification_id
        label: "A justification"
        assertion:
          -
```

```
            id: assertion_id
            description: "A description of an assertion"
            source:
              -
                id: source1_id
                label: "First source"
              -
                id: source2_id
                label: "second source"
---
';
justifications <-
  load_justifications(text=exampleJustifier);
miniGraph_original <-
  justifications$decisionGraphs[[1]];
miniGraph <-
  apply_graph_theme(miniGraph_original,
                    c("color", "#0000AA", "node"),
                    c("shape", "triangle", "node"),
                    c("fontcolor", "#FF0000", "node"));
### This line should be run when executing this example as test, because
### rendering a DiagrammeR graph takes quite long
## Not run:
DiagrammeR::render_graph(miniGraph);

## End(Not run)
```

`load_justifications` *Load Justifications from a file or multiple files*

Description

These function load justifications from the YAML fragments in one (`load_justifications`) or multiple files (`load_justifications_dir`).

Usage

```
load_justifications(text, file, delimiterRegEx = "^---$",
  justificationContainer = c("justifier", "justification", "decision",
  "assertion", "source"), ignoreOddDelimiters = FALSE,
  encoding = "UTF-8", silent = TRUE)

load_justifications_dir(path, recursive = TRUE, extension = "jmd",
  regex, justificationContainer = c("justifier", "justification",
  "decision", "assertion", "source"), delimiterRegEx = "^---$",
  ignoreOddDelimiters = FALSE, encoding = "UTF-8", silent = TRUE)
```

Arguments

text, file	As text or file, you can specify a file to read with encoding encoding, which will then be read using base::readLines(). If the argument is named text, whether it is the path to an existing file is checked first, and if it is, that file is read. If the argument is named file, and it does not point to an existing file, an error is produced (useful if calling from other functions). A text should be a character vector where every element is a line of the original source (like provided by base::readLines()); although if a character vector of one element *and* including at least one newline character (\n) is provided as text, it is split at the newline characters using base::strsplit(). Basically, this behavior means that the first argument can be either a character vector or the path to a file; and if you're specifying a file and you want to be certain that an error is thrown if it doesn't exist, make sure to name it file.
delimiterRegEx	The regular expression used to locate YAML fragments
justificationContainer	The container of the justifications in the YAML fragments. Because only justifications are read that are stored in this container, the files can contain YAML fragments with other data, too, without interfering with the parsing of the justifications.
ignoreOddDelimiters	Whether to throw an error (FALSE) or delete the last delimiter (TRUE) if an odd number of delimiters is encountered.
encoding	The encoding to use when calling readLines(). Set to NULL to let readLines() guess.
silent	Whether to be silent (TRUE) or informative (FALSE).
path	The path containing the files to read.
recursive	Whether to also process subdirectories (TRUE) or not (FALSE).
extension	The extension of the files to read; files with other extensions will be ignored. Multiple extensions can be separated by a pipe (|).
regex	Instead of specifing an extension, it's also possible to specify a regular expression; only files matching this regular expression are read. If specified, regex takes precedece over extension,

Details

load_justifications_dir simply identifies all files and then calls load_justifications for each of them. load_justifications loads the YAML fragments containing the justifications using yum::load_yaml_fragments() and then parses the justifications into a visual representation as a ggplot2::ggplot graph and Markdown documents with overviews.

Value

An object with the ggplot2::ggplot graph stored in output$graph and the overview in output$overview.

Examples

```
exampleMinutes <- 'This is an example of minutes that include
a source, an assertion, and a justification. For example, in
the meeting, we can discuss the assertion that sleep deprivation
affects decision making. We could quickly enter this assertion in
a machine-readable way in this manner:

---
assertion:
  -
    id: assertion_SD_decision
    label: Sleep deprivation affects the decision making proces.
    source:
      id: source_Harrison
---

Because it is important to refer to sources, we cite a source as well.
We have maybe specified that source elsewhere, for example in the
minutes of our last meeting. That specification may have looked
like this:

---
source:
  -
    id: source_Harrison
   label: "Harrison & Horne (2000) The impact of sleep deprivation on decision making: A review."
    xdoi: "doi:10.1037/1076-898x.6.3.236"
    type: "Journal article"
---

We can now refer to these two specifications later on, for
example to justify decisions we take.
';

load_justifications(text=exampleMinutes);

### To load a directory with justifications
examplePath <-
  file.path(system.file(package="justifier"),
            'extdata');
load_justifications_dir(path=examplePath);
```

`parse_justifications` *Parsing justifications*

Description

This function is normally called by `load_justifications()`; however, sometimes it may be desirable to parse justifications embedded in more complex objects, for example as provided by `yum::load_and_simplify()`. Therefore, this function can also be called directly.

Usage

```
parse_justifications(x)
```

Arguments

x	An object resulting from a call to yum::load_and_simplify().

Details

While there is some flexibility in how justifications can be specified, they are most easily processed further if they all follow the same conventions. This function ensures this. The convention is as follows:

- all specifications are provided in four 'flat' lists, named after the types of elements they contain;
- all elements have a unique identifier
- all references to other elements are indeed only references to the other elements' id's in these 'flat lists'

Value

The parsed `justifier` object

Examples

```
### Specify an example text
exampleFile <-
  system.file("extdata",
              "simple-example.jmd",
              package="justifier");

### Show contents
cat(readLines(exampleFile), sep="\n");

### Load it with yum::load_and_simplify()
loadedMinutes <- yum::load_and_simplify(exampleFile);

### Show contents
names(loadedMinutes);

### Parse 'manually'
parsedJustifications <- justifier::parse_justifications(loadedMinutes);

### Show contents
names(parsedJustifications);
```

sanitize_for_DiagrammeR *Sanitize for DiagrammeR*

Description

Basically a wrapper for `gsub()` to sanitize a string for DiagrammeR

Usage

```
sanitize_for_DiagrammeR(x,
  regExReplacements = list(c("\\\"", "`"), c("\\'", "`"), c("\\\\", "/")))
```

Arguments

`x`	The string or vector
`regExReplacements`	A list of two-element character vectors; first element should be the elemnet to search, and the second element, the replacement.

Value

The sanitized character vector

Examples

```
justifier::sanitize_for_DiagrammeR("This is or isn't problematic");
```

to_specList *Producing a list of specifications*

Description

This function is for internal use, but has been exported in case it's useful for people working 'manually' with lists of justifications.

Usage

```
to_specList(x, types, type)
```

Arguments

`x`	The list to parse.
`types`	The class to assign to the specification list (the `justifierSpecList` object to return).
`type`	The class to assign to each specification (in addition to `justifierSpec`).

Value

A list of classes `c("justifierSpecList", types)` where each element is a specification of class `c("justifierSpec", type)`.

Examples

```
### Specify an example text
exampleFile <-
  system.file("extdata",
              "simple-example.jmd",
              package="justifier");

### Show contents
cat(readLines(exampleFile), sep="\n");

### Load it with yum::load_and_simplify()
loadedMinutes <- yum::load_and_simplify(exampleFile);

### Show contents
names(loadedMinutes);

### Show classes
class(loadedMinutes["assertion"]);

### Convert to specification list
res <- to_specList(loadedMinutes["assertion"],
                   type="assertion",
                   types="assertions");
### Show classes
class(res);

### Show original and parsed objects
loadedMinutes["assertion"];
res;
```

Index