Introduction to the **pandocfilters** Package

September 6, 2024

```
library("pandocfilters")

##
## Attaching package:  'pandocfilters'
## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:methods':
##
##     Math
```

The document converter pandoc is widely used in the R community. One feature of pandoc is that it can produce and consume JSON-formatted abstract syntax trees (AST). This allows to transform a given source document into JSON-formatted AST, alter it by so called filters and pass the altered JSON-formatted AST back to pandoc. This package provides functions which allow to write such filters in native R code. The package is inspired by the Python package pandocfilters.
To alter the AST, the JSON representations of the data structures building the AST have to be replicated. For this purpose, **pandocfilters** provides a set of constructors, with the goal to ease building / altering the AST.

1 Installation

Detailed information about installing pandoc, can be found at http://pandoc.org/installing.html. For the new pandoc releases there exist precompiled pandoc versions for Linux, Windows and macOS.

2 Setup

If **pandoc** is set as PATH variable

```
system2("pandoc", "--version", stdout = TRUE, stderr = TRUE)

## [1] "pandoc 3.2.1"
## [2] "Features: +server +lua"
## [3] "Scripting engine: Lua 5.4"
## [4] "User data directory: /github/home/.local/share/pandoc"
```

```
## [5] "Copyright (C) 2006-2024 John MacFarlane. Web: https://pandoc.org"
## [6] "This is free software; see the source for copying conditions. There is no"
## [7] "warranty, not even for merchantability or fitness for a particular purpose."
```

should show the version and some additional information.

2.1 Alter the pandoc version

There are several options to alter the pandoc version used by **pandocfilters**,

1. alter your PATH variable accordingly
2. set the system variable `"PANDOC_HOME"`
3. use `set_pandoc_path` after **pandocfilters** is loaded

2.1.1 Set the system variable `"PANDOC_HOME"`

To set persistent environment variables either the file `".Renviron"` or the file `".Rprofile"` can be used. More information about `".Rprofile"` and `".Renviron"` can be found in the R-Documentation. Therefore a `".Renviron"` file on Unix could look like the following

```
PANDOC_HOME=/home/florian/bin/pandoc/pandoc_292/bin/pandoc
```

or on Windows

```
PANDOC_HOME=C:/Users/Florian/AppData/Local/Pandoc/pandoc.exe
```

similarly a `".Rprofile"` file on Unix could look like the following

```
Sys.setenv(PANDOC_HOME="/home/florian/bin/pandoc/pandoc_292/bin/pandoc")
```

or on Windows

```
Sys.setenv(PANDOC_HOME="C:/Users/Florian/AppData/Local/Pandoc/pandoc.exe")
```

2.1.2 Use `set_pandoc_path`

After pandoc is loaded the used version can be altered by `set_pandoc_path`.

```
get_pandoc_version()

## [1] 2.2

set_pandoc_path("/home/florian/bin/pandoc/pandoc_221/bin/pandoc")
get_pandoc_version()

## [1] 2.2
```

3 Constructors

As mentioned before, constructors are used to replicate the pandoc AST in R. For this purpose, pandoc provides two basic types, **inline** elements and **block** elements. An extensive list can be found below.

To minimize the amount of unnecessary typing **pandocfilters** automatically converts character strings to pandoc objects of type `"Str"` if needed. Furthermore, if a single inline object is provided where a list of inline objects is needed **pandocfilters** automatically converts this inline object into a list of inline objects.

For example, the canonical way to emphasize the character string `"some text"` would be

```
Emph(list(Str("some text")))
```

Since single inline objects are automatically transformed to lists of inline objects, this is equivalent to

```
Emph(Str("some text"))
```

Since a character string is automatically transformed to an inline object, this is equivalent to

```
Emph("some text")
```

In short, whenever a list of inline objects is needed one can also use a single inline object or a character string, and therefore the following three code lines are equivalent.

```
Emph(list(Str("some text")))
Emph(Str("some text"))
Emph("some text")
```

3.1 Inline Elements

1. `Str(x)`
2. `Emph(x)`
3. `Strong(x)`
4. `Strikeout(x)`
5. `Superscript(x)`
6. `Subscript(x)`
7. `SmallCaps(x)`
8. `Quoted(x, quote_type)`
9. `Cite(citation, x)`
10. `Code(code, name, language, line_numbers, start_from)`

11. `Space()`
12. `SoftBreak()`
13. `LineBreak()`
14. `Math(x)`
15. `RawInline(format, x)`
16. `Link(target, text, title, attr)`
17. `Image(target, text, caption, attr)`
18. `Span(attr, inline)`

3.2 Block Elements

1. `Plain(x)`
2. `Para(x)`
3. `CodeBlock(attr, code)`
4. `BlockQuote(blocks)`
5. `OrderedList(lattr, lblocks)`
6. `BulletList(lblocks)`
7. `DefinitionList(x)`
8. `Header(x, level, attr)`
9. `HorizontalRule()`
10. `Table(rows, col_names, aligns, col_width, caption)`
11. `Div(blocks, attr)`
12. `Null()`

3.3 Argument Constructors

1. `Attr(identifier, classes, key_val_pairs)`
2. `Citation(suffix, id, note_num, mode, prefix, hash)`
3. `TableCell(x)`

4 Altering the AST

To read / write / alter the AST the following functions can be used.

```
pandoc_to_json

## function (file, from = "markdown")

pandoc_from_json

## function (json, to = "markdown", exchange = c("file", "arg"))

filter

## function (FUN, ..., input = stdin(), output = stdout())
```

4.1 Examples

4.1.1 Lower Case

In this example we take the first few lines from the R-FAQ Section "`2.1 What is R?`" stored in the a markdown file `"lowe_case.md"`

```
ex1_file <- system.file(package = "pandocfilters",
                        "examples", "lower_case.md")
readLines(ex1_file)

## [1] "## 2.1 What is R?"
## [2] ""
## [3] "R is a system for statistical computation and graphics.  It consists of a"
## [4] "language plus a run-time environment with graphics, a debugger, access to"
## [5] "certain system functions, and the ability to run programs stored in script"
## [6] "files."
```

and use **pandocfilters** to obtain the AST representation of this document. Since pandoc filters are typically used in the terminal the default input is the `stdin` and the default output is `stdout`, however to stay within R we will use text connections instead.
First we setup a read connection for the input and a write connection for the output.

```
icon <- textConnection(pandoc_to_json(ex1_file))
ocon <- textConnection("modified_ast", open = "w")
```

Second we define a function to alter the AST

```
lower <- function(key, value, ...) {
    if (key == "Str") Str(tolower(value)) else NULL
}
```

and apply it on the AST.

```
filter(lower, input = icon, output = ocon)
```

At the end we convert altered AST back to markdown

```
pandoc_from_json(modified_ast, to ="markdown")
```

and close the open connections.

```
close(icon)
close(ocon)
```