

# Package: pandocfilters (via r-universe)

September 6, 2024

**Title** Pandoc Filters for R

**Version** 0.1-6

**Description** The document converter 'pandoc' <<https://pandoc.org/>> 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. Although this package is inspired by the Python package 'pandocfilters' <<https://github.com/jgm/pandocfilters/>>, it provides additional convenience functions which make it simple to use the 'pandocfilters' package as a report generator. Since 'pandocfilters' inherits most of it's functionality from 'pandoc' it can create documents in many formats (for more information see <<https://pandoc.org/>>) but is also bound to the same limitations as 'pandoc'.

**URL** <https://pandoc.org/>, <https://github.com/jgm/pandocfilters/>

**Depends** R (>= 3.0.0)

**Imports** jsonlite, utils

**Suggests** knitr

**VignetteBuilder** knitr

**SystemRequirements** pandoc (> 1.12)

**License** GPL-3

**RoxygenNote** 7.2.1

**NeedsCompilation** no

**Author** Florian Schwendinger [aut, cre], Kurt Hornik [aut], Andrie de Vries [ctb]

**Maintainer** Florian Schwendinger <FlorianSchwendinger@gmx.at>

**Repository** CRAN

**Date/Publication** 2022-08-11 21:40:02 UTC

## Contents

as.block . . . . .	3
as.inline . . . . .	3
astrapply . . . . .	4
Attr . . . . .	4
BlockQuote . . . . .	5
BulletList . . . . .	5
c.block . . . . .	6
c.inline . . . . .	6
Citation . . . . .	7
Cite . . . . .	7
Code . . . . .	8
CodeBlock . . . . .	8
Definition . . . . .	9
DefinitionList . . . . .	9
Div . . . . .	10
document . . . . .	10
Emph . . . . .	12
filter . . . . .	13
get_pandoc_path . . . . .	13
get_pandoc_types_version . . . . .	14
get_pandoc_version . . . . .	14
Header . . . . .	15
HorizontalRule . . . . .	15
Image . . . . .	16
is.block . . . . .	16
is.inline . . . . .	17
LineBreak . . . . .	17
Link . . . . .	18
ListAttributes . . . . .	18
Math . . . . .	19
Note . . . . .	19
Null . . . . .	20
OrderedList . . . . .	20
pandoc_to_json . . . . .	21
Para . . . . .	21
Plain . . . . .	22
Quoted . . . . .	22
RawInline . . . . .	23
set_pandoc_path . . . . .	23
SmallCaps . . . . .	24
SoftBreak . . . . .	24
Space . . . . .	24
Span . . . . .	25
Str . . . . .	25
Strikeout . . . . .	26
Strong . . . . .	26

`as.block` 3

Subscript . . . . .	27
Superscript . . . . .	27
Table . . . . .	28
TableCell . . . . .	28
write.pandoc . . . . .	29

**Index** **30**

---

`as.block` *Block Objects*

---

**Description**

In pandoc "block" objects are used as container for "inline" objects and to give them specific roles. Objects of the classes "NULL" and "character" can be coerced to "block".

**Usage**

```
as.block(x)
```

**Arguments**

x                    an object of type "NULL" or "character" or "block".

**Value**

an object of class "block".

**Examples**

```
as.block("some text")
as.block(NULL)
```

---

`as.inline` *Inline Objects*

---

**Description**

Objects of the classes "NULL" and "character" can be coerced to "inline".

**Usage**

```
as.inline(x)
```

**Arguments**

x                    an object of type "NULL", "character" or "inline".

**Value**

an object of class "inline".

**Examples**

```
as.inline("some text")
as.inline(NULL)
```

---

astrapply	<i>Apply a Function on a AST</i>
-----------	----------------------------------

---

**Description**

Apply the function FUN on the abstract syntax tree (AST) obtained from pandoc.

**Usage**

```
astrapply(x, FUN, ...)
```

**Arguments**

x	a list representing the AST obtained from pandoc.
FUN	the function to be applied to the AST.
...	optional arguments to FUN.

**Value**

A list containing the modified AST.

---

Attr	<i>Attributes</i>
------	-------------------

---

**Description**

A constructor for pandoc attributes.

**Usage**

```
Attr(identifier = "", classes = character(), key_val_pairs = list())
```

**Arguments**

identifier	a character string
classes	a character giving the classes
key_val_pairs	a list of tuple of type "character"

**Examples**

```
Attr("A", c("B", "C"), list(c("D", "E")))
```

---

BlockQuote

*Block Quote*

---

**Description**

Constructs a block object of type "BlockQuote".

**Usage**

```
BlockQuote(blocks)
```

**Arguments**

blocks            a block object or list of block objects

**Examples**

```
BlockQuote(Plain("Hello R!"))
```

---

BulletList

*Bullet List*

---

**Description**

Constructs a block object of type "BulletList".

**Usage**

```
BulletList(llblocks)
```

**Arguments**

llblocks            a list of lists of blocks

**Examples**

```
bullet_1 <- Plain("A")
bullet_2 <- Plain(Str("B"))
bullet_3 <- list(Plain(list(Str("C"))))
BulletList(list(bullet_1, bullet_2, bullet_3))
```

c.block

*Combine Block Objects*

---

**Description**

Objects of class "block" can be combined by using the generic default method "c" (combine).

**Usage**

```
## S3 method for class 'block'  
c(...)
```

**Arguments**

... objects to be concatenated.

**Value**

an list of "block" objects.

**Examples**

```
c(Header( "R Basics" ), Header("What is R?", level=2),  
Plain(c(Emph("R"), Space(), "is a system for ", Strong("statistical computation"))))
```

---

c.inline

*Combine Inline Objects*

---

**Description**

Objects of class "inline" can be combined by using the generic default method "c" (combine).

**Usage**

```
## S3 method for class 'inline'  
c(...)
```

**Arguments**

... objects to be concatenated.

**Value**

an list of "inline" objects.

**Examples**

```
c(Str("some"), Strong("text"))
```

---

Citation

*Citation*


---

**Description**

Constructs an object of type "Citation".

**Usage**

```
Citation(
    suffix,
    id,
    note_num = 0L,
    mode = "AuthorInText",
    prefix = list(),
    hash = 0L
)
```

**Arguments**

suffix	a inline object or list of inline objects
id	a character string (not visible in the text)
note_num	an integer
mode	a character string giving the citation mode, possible values are "AuthorInText", "SuppressAuthor" and "NormalCitation".
prefix	a inline object or list of inline objects
hash	an integer

---

Cite

*Citation*


---

**Description**

Constructs an inline object of type "Cite".

**Usage**

```
Cite(citation, x)
```

**Arguments**

citation	an object of type "Citation"
x	a inline object or a list of inline objects

**Examples**

```
ci <- Citation(suffix=list(Str("Suffix_1")),
               id="Citation_ID_1", prefix=list(Str("Prefix_1")))
Cite(ci, Str("some text"))
```

---

Code

*Inline Code*


---

**Description**

Constructs an inline object of type "Code".

**Usage**

```
Code(code, name = "", language = NULL, line_numbers = FALSE, start_from = 1)
```

**Arguments**

code	a character string giving the inline code
name	an optional character string giving the name of the inline code chunk
language	an optional character string giving the programming language
line_numbers	a logical which controls if line numbers should be used
start_from	an integer giving the first line number

**Examples**

```
Code("lm(hello ~ world)", "my_r_inline_code", "R", TRUE, 0)
Code("lm(hello ~ world)")
```

---

CodeBlock

*Code Block*


---

**Description**

Constructs a block object of type "CodeBlock".

**Usage**

```
CodeBlock(attr, code)
```

**Arguments**

attr	an object of type "Attr"
code	a character string containing the source code.



**Examples**

```
attr <- Attr("id", "Programming Language", list(c("key", "value")))
code <- "x <- 3\nprint('Hello R!')"
```

```
CodeBlock(attr, code)
```

---

 Definition

*Definition*


---

**Description**

Constructs a Definition which can be used as an element of a "DefinitionList".

**Usage**

```
Definition(key, value)
```

**Arguments**

key	a inline object or list of inline objects
value	a block object or list of block objects

**Examples**

```
Definition("some key", Plain("some value"))
```

---

 DefinitionList

*Definition List*


---

**Description**

Constructs a block object of type "DefinitionList".

**Usage**

```
DefinitionList(x)
```

**Arguments**

x	a list of key value pairs, the key is a list of "inline" objects and the values are a list of lists of objects of type "block".
---	---

**Details**

In the pandoc API <https://johnmacfarlane.net/BayHac2014/doc/pandoc-types/Text-Pandoc-Definition.html> the DefinitionList is described as follows, each list item is a pair consisting of a term (a list of "inline" objects) and one or more definitions (each a list of blocks).

**Examples**

```
key <- list(Str("key"))
value <- list(list(Plain(list(Str("value")))))
DefinitionList(list(list(key, value), Definition("some key", Plain("some value"))))
```

---

**Div***Generic Block Container with Attributes*

---

**Description**

Constructs a block object of type "Div".

**Usage**

```
Div(blocks, attr = Attr())
```

**Arguments**

blocks	a block object or list of block objects
attr	an object of type "Attr"

**Examples**

```
blocks <- Plain("Hello R!")
Div(blocks)
```

---

**document***Create a new Document*

---

**Description**

Constructs an object of type "document".

**Usage**

```
document()
```

**Details**

Each document has the following methods:

```
to_json()
```

**Description**

Returns the JSON representation of the document.

```
write(con, format = "markdown", writer = write.pandoc)
```

**Description**

Write the JSON-formatted AST to a connection.

**Arguments**

<code>con</code>	a connection object or a character string to which the document is written.
<code>format</code>	a character string giving the format (e.g. "latex", "html").
<code>writer</code>	an optional writer function, see <a href="#">write.pandoc</a> .

**Note**

Any function with the three arguments `x`, `con` and `format` can be used as writer function.

```
append(x)
```

**Description**

Append a new block to the document.

**Arguments**

`x` block object or list of block objects.

```
append_plain(x)
```

**Description**

For more information about the arguments see [Plain](#).

```
append_para(x)
```

**Description**

For more information about the arguments see [Para](#).

```
append_code_block(attr, code)
```

**Description**

For more information about the arguments see [CodeBlock](#).

```
append_block_quote(blocks)
```

**Description**

For more information about the arguments see [BlockQuote](#).

```
append_ordered_list(lattr, lblocks)
```

**Description**

For more information about the arguments see [OrderedList](#).

```
append_bullet_list(lblocks)
```

**Description**

For more information about the arguments see [BulletList](#).

`append_definition_list(x)`

**Description**

For more information about the arguments see [DefinitionList](#).

`append_header(x, level=1L, attr=Attr())`

**Description**

For more information about the arguments see [Header](#).

`append_horizontal_rule()`

**Description**

For more information about the arguments see [HorizontalRule](#).

`append_table(rows, col_names=NULL, aligns=NULL, col_width=NULL, caption=list())`

**Description**

For more information about the arguments see [Table](#).

`append_div(blocks, attr)`

**Description**

For more information about the arguments see [Div](#).

`append_null()`

**Description**

For more information about the arguments see [Null](#).

---

Emph

*Emphasized Text*

---

**Description**

Constructs an inline object of type "Emph".

**Usage**

`Emph(x)`

**Arguments**

`x` a inline object or a list of inline objects

**Examples**

`Emph("emphasize")`

---

filter	<i>Filter JSON-formatted AST.</i>
--------	-----------------------------------

---

**Description**

Apply a filter on the JSON-formatted abstract syntax tree (AST).

**Usage**

```
filter(FUN, ..., input = stdin(), output = stdout())
```

**Arguments**

FUN	the function to be applied on the AST.
...	optional arguments to FUN.
input	a connection object or a character string from which the JSON-formatted AST is read.
output	a connection object or a character string to which the JSON-formatted AST is written.

---

get_pandoc_path	<i>Get Pandoc Path</i>
-----------------	------------------------

---

**Description**

Get the path of pandoc.

**Usage**

```
get_pandoc_path()
```

---

`get_pandoc_types_version`*Get Pandoc-Types Version*

---

**Description**

Get the version of pandoc-types.

**Usage**

```
get_pandoc_types_version(type = c("numeric", "character"))
```

**Arguments**

`type` a character giving the type of the return value.

**Examples**

```
get_pandoc_types_version()
```

---

`get_pandoc_version` *Get Pandoc Version*

---

**Description**

Get the version of pandoc.

**Usage**

```
get_pandoc_version(type = c("numeric", "character"))
```

**Arguments**

`type` a character giving the type of the return value.

**Examples**

```
get_pandoc_version()
```

---

Header

*Header*

---

**Description**

Constructs a block object of type "Header".

**Usage**

```
Header(x, level = 1L, attr = Attr())
```

**Arguments**

x	a inline object or a list of inline objects
level	an integer giving the level
attr	an object of type "Attr"

**Examples**

```
Header("My Header")
```

---

HorizontalRule

*Horizontal Rule*

---

**Description**

Constructs a block object of type "HorizontalRule".

**Usage**

```
HorizontalRule()
```

**Examples**

```
HorizontalRule()
```

---

 Image
 

---

*Image***Description**

Constructs an inline object of type "Image".

**Usage**

```
Image(target, text, caption = "", attr = Attr())
```

**Arguments**

target	a character string giving the target (hyper reference)
text	a inline object or a list of inline objects giving the visible part
caption	a character string describing the picture
attr	an optional object of type "Attr"

**Details**

Further Usage examples can be found in the README.

**Examples**

```
Image("https://Rlogo.jpg", "some_text", "fig:some_caption")
```

---

is.block

*Block Objects***Description**

Tests if an object has the class attribute "block".

**Usage**

```
is.block(x)
```

**Arguments**

x	an object to be tested.
---	-------------------------

**Value**

a logical indicating if the provided object is of type "block".

**Examples**

```
is.block(as.block(NULL))
```



---

`is.inline`

*Inline Objects*

---

**Description**

Tests if an object has the class attribute "inline".

**Usage**

```
is.inline(x)
```

**Arguments**

x                    an object to be tested.

**Value**

a logical indicating if the provided object is of type "inline".

**Examples**

```
is.inline(as.inline(NULL))
```

---

`LineBreak`

*Hard Line Break*

---

**Description**

Constructs an inline object of type "LineBreak".

**Usage**

```
LineBreak()
```

**Examples**

```
LineBreak()
```

---

Link	<i>Hyperlink</i>
------	------------------

---

**Description**

Constructs an inline object of type "Link".

**Usage**

```
Link(target, text, title = "", attr = Attr())
```

**Arguments**

target	a character string giving the target (hyper reference)
text	a inline object or a list of inline objects giving the visible part
title	an optional character string giving the title
attr	an optional object of type "Attr"

**Details**

Further Usage examples can be found in the README.

**Examples**

```
Link("https://cran.r-project.org/", "Text_Shown", "some title")
```

---

ListAttributes	<i>ListAttributes</i>
----------------	-----------------------

---

**Description**

A constructor for pandoc list attributes.

**Usage**

```
ListAttributes(
  first_number = 1L,
  style = "DefaultStyle",
  delim = "DefaultDelim"
)
```

**Arguments**

first_number	an integer giving the first number of the list
style	a character string giving the style, possible values are "DefaultStyle", "Example", "Decimal", "LowerRoman", "UpperRoman", "LowerAlpha" and "UpperAlpha".
delim	a character string giving the delimiter, possible values are "DefaultDelim", "Period", "OneParen" and "TwoParens".

---

 Math

---

*TeX Math*


---

**Description**

Constructs an inline object of type "Math".

**Usage**

Math(x)

**Arguments**

x                    a character string

**Examples**

```
Math("3*x^2")
```

---

 Note

---

*Note*


---

**Description**

Constructs an inline object of type "Note".

**Usage**

Note(x)

**Arguments**

x                    a pandoc block object or a list of pandoc block objects

**Examples**

```
block <- Plain("x")
Note(block)
```

---

Null	<i>Nothing</i>
------	----------------

---

**Description**

Constructs a block object of type "Null".

**Usage**

```
Null()
```

**Examples**

```
Null()
```

---

OrderedList	<i>Ordered List</i>
-------------	---------------------

---

**Description**

Constructs a block object of type "OrderedList".

**Usage**

```
OrderedList(lattr, llblocks)
```

**Arguments**

lattr	a list of attributes
llblocks	a list of lists of blocks

**Examples**

```
ordered_1 <- Plain("A")
ordered_2 <- list(Plain(Str("B")))
ordered_3 <- list(Plain(list(Str("C"))))
OrderedList(ListAttributes(), ordered_1)
OrderedList(ListAttributes(), list(ordered_1, ordered_2, ordered_3))
```

---

pandoc\_to\_json            *Utility functions for testing filters*

---

**Description**

Utility functions for testing filters

**Usage**

```
pandoc_to_json(file, from = "markdown")
```

```
pandoc_from_json(json, to = "markdown", exchange = c("file", "arg"))
```

**Arguments**

file	file name
from	markdown, html, latex or native
json	a JSON representation of the AST to be passed to pandoc
to	markdown, html, latex or native
exchange	a character string

---

Para                    *Paragraph*

---

**Description**

Constructs a block object of type "Para".

**Usage**

```
Para(x)
```

**Arguments**

x	a inline object or list of inline objects
---	---

**Examples**

```
Para("x")
```

---

Plain

*Plain Text*

---

**Description**

Constructs a block object of type "Plain", a plain paragraph.

**Usage**

```
Plain(x)
```

**Arguments**

x                    a inline object or list of inline objects

**Examples**

```
Plain("x")
```

---

Quoted

*Quoted Text*

---

**Description**

Constructs an inline object of type "Quoted".

**Usage**

```
Quoted(x, quote_type = "DoubleQuote")
```

**Arguments**

x                    a inline object or a list of inline objects  
quote\_type          a character giving the quote type, valid types are "SingleQuote" and "DoubleQuote"

**Examples**

```
Quoted("some text", quote_type="SingleQuote")  
Quoted("some text", quote_type="DoubleQuote")
```

---

RawInline	<i>Raw Inline</i>
-----------	-------------------

---

**Description**

Constructs an inline object of type "RawInline".

**Usage**

```
RawInline(format, x)
```

**Arguments**

format	a character string giving the format (e.g. "latex", "html")
x	a character string giving the inline

**Examples**

```
RawInline("latex", "some RawInline")
```

---

set_pandoc_path	<i>Set Pandoc Path</i>
-----------------	------------------------

---

**Description**

Set the path to pandoc.

**Usage**

```
set_pandoc_path(path = "pandoc")
```

**Arguments**

path	a character giving the location of pandoc (default is "pandoc" which uses the pandoc set in the system path).
------	---

---

SmallCaps	<i>Small Caps Text</i>
-----------	------------------------

---

**Description**

Constructs an inline object of type "SmallCaps".

**Usage**

```
SmallCaps(x)
```

**Arguments**

x                    a inline object or a list of inline objects

**Examples**

```
SmallCaps("The latex command for 'small caps' is 'textsc'!")
```

---

SoftBreak	<i>Soft Line Break</i>
-----------	------------------------

---

**Description**

Constructs an inline object of type "SoftBreak".

**Usage**

```
SoftBreak()
```

**Examples**

```
SoftBreak()
```

---

Space	<i>Inter-word space</i>
-------	-------------------------

---

**Description**

Constructs an inline object of type "Space".

**Usage**

```
Space()
```

**Examples**

```
Space()
```



---

Span	<i>Generic Inline Container with Attributes</i>
------	---

---

**Description**

Constructs an inline object of type "Span".

**Usage**

```
Span(attr, inline)
```

**Arguments**

attr	an object of type "Attr"
inline	a inline object or a list of inline objects which will be shown

**Examples**

```
attr <- Attr("A", "B", list(c("C", "D")))
Span(attr, "some inline string")
```

---

Str	<i>Text (String)</i>
-----	----------------------

---

**Description**

Constructs an inline object of type "Str".

**Usage**

```
Str(x)
```

**Arguments**

x	a character string
---	--------------------

**Details**

To minimize the amount of unnecessary typing, pandoc filters 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 string")`. In short, whenever a list of inline objects is needed one can also use a single inline object or a character string.

**Examples**

```
Str("SomeString")
```

---

Strikeout

*Strikeout Text*

---

**Description**

Constructs an inline object of type "Strikeout".

**Usage**

```
Strikeout(x)
```

**Arguments**

x                    a inline object or a list of inline objects

**Examples**

```
Strikeout("strikeout")
```

---

Strong

*Strongly Emphasized Text*

---

**Description**

Constructs an inline object of type "Strong".

**Usage**

```
Strong(x)
```

**Arguments**

x                    a inline object or a list of inline objects

**Examples**

```
Strong("strong")
```

---

Subscript

*Subscripted Text*

---

**Description**

Constructs an inline object of type "Subscript".

**Usage**

Subscript(x)

**Arguments**

x                    a inline object or a list of inline objects

**Examples**

Subscript("some text written in superscript")

---

Superscript

*Superscripted Text*

---

**Description**

Constructs an inline object of type "Superscript".

**Usage**

Superscript(x)

**Arguments**

x                    a inline object or a list of inline objects

**Examples**

Superscript("some text written in superscript")

---

Table	<i>Table</i>
-------	--------------

---

**Description**

Constructs a block object of type "Table".

**Usage**

```
Table(
  rows,
  col_names = NULL,
  aligns = NULL,
  col_width = NULL,
  caption = list()
)
```

**Arguments**

rows	an object of class "matrix", "data.frame", "table" or a list of lists of pandoc objects of type "TableCell"
col_names	a list of objects of type "TableCell"
aligns	a character vector of alignments, possible values are "l" for left, "r" for right, "c" for center and "d" for default.
col_width	a numeric vector
caption	a inline object or a list of inline objects giving the caption

**Details**

Table, with caption, column alignments (required), relative column widths (0 = default), column headers (each a list of blocks), and rows (each a list of lists of blocks)

---

TableCell	<i>Table Cell</i>
-----------	-------------------

---

**Description**

Table cells is a constructor for plain table cells.

**Usage**

```
TableCell(x)
```

**Arguments**

x	a character string giving the content of the table cell
---	---

**Details**

In general table cells are a list of block elements, the constructor TableCell creates a plain table cell.

**Examples**

```
TableCell("Cell 1")
```

---

```
write.pandoc
```

*Write the JSON-formatted AST to a connection*

---

**Description**

Write the JSON-formatted AST to a connection.

**Usage**

```
write.pandoc(json, file, format, exchange = c("arg", "file"))
```

**Arguments**

json	a JSON representation of the AST to be written out
file	a connection object or a character string to which the JSON-formatted AST is written
format	a character string giving the format (e.g. "latex", "html")
exchange	a character string

**Details**

If you want to apply a filter to the document before it get's written out, or your pandoc installation is not registered in the PATH it can be favorable to provide your own writer function to the document class.

# Index

`as.block`, 3  
`as.inline`, 3  
`astrapply`, 4  
`Attr`, 4

`BlockQuote`, 5, 11  
`BulletList`, 5, 11

`c.block`, 6  
`c.inline`, 6  
`Citation`, 7  
`Cite`, 7  
`Code`, 8  
`CodeBlock`, 8, 11

`Definition`, 9  
`DefinitionList`, 9, 12  
`Div`, 10, 12  
`document`, 10

`Emph`, 12

`filter`, 13

`get_pandoc_path`, 13  
`get_pandoc_types_version`, 14  
`get_pandoc_version`, 14

`Header`, 12, 15  
`HorizontalRule`, 12, 15

`Image`, 16  
`is.block`, 16  
`is.inline`, 17

`LineBreak`, 17  
`Link`, 18  
`ListAttributes`, 18

`Math`, 19

`Note`, 19

`Null`, 12, 20

`OrderedList`, 11, 20

`pandoc_from_json (pandoc_to_json)`, 21  
`pandoc_to_json`, 21  
`Para`, 11, 21  
`Plain`, 11, 22

`Quoted`, 22

`RawInline`, 23

`set_pandoc_path`, 23  
`SmallCaps`, 24  
`SoftBreak`, 24  
`Space`, 24  
`Span`, 25  
`Str`, 25  
`Strikeout`, 26  
`Strong`, 26  
`Subscript`, 27  
`Superscript`, 27

`Table`, 12, 28  
`TableCell`, 28

`write.pandoc`, 11, 29