Updates to the R Documentation System

Andrew Redd, R documentation Task Force

UseR!2017

History

- UseR!2016
- July 2016, R Consortium Funding
- September-December 2016
- January 2017 - Now

Design Principles

- Full class system for documentation
- Can document anything
- stored in `attr(., 'documentation')` for traditional objects;
  - functions, data.frames, etc.
- Stored as a meta object similar to class definitions for S4 and reference classes.
- Everything can be accessed or set from the `documentation` and `documentation<-` generics.
- Formats for input and output support decided
- Independent of but supports the package system

Many in - Many Out

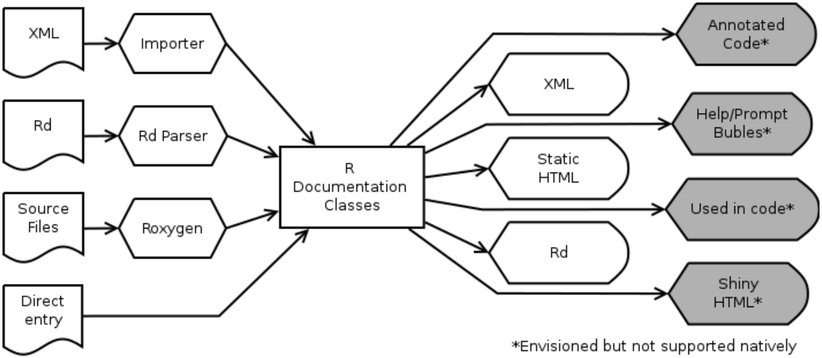

Figure 1: Many in - Many out approach.

Current State

- `parsetools`
- `documentation`
- `documentation-md`
- `documentation-yaml`

parsetools

```
library(parsetools)
options(keep.source=TRUE)
pd <- get_parse_data(parse(text="x <- rnorm(100)"))
kable(pd[c(-1,-3)])
```

	col1	col2	id	parent	token	terminal
1	1	1	1	3	SYMBOL	TRUE
3	1	1	3	14	expr	FALSE
14	1	15	14	0	expr	FALSE
2	3	4	2	14	LEFT_ASSIGN	TRUE
4	6	10	4	6	SYMBOL_FUNCTION_CALL	TRUE
6	6	10	6	12	expr	FALSE
12	6	15	12	14	expr	FALSE
5	11	11	5	12	‘(’	TRUE
7	12	14	7	8	NUM_CONST	TRUE
8	12	14	8	12	expr	FALSE
9	15	15	9	12	‘)’	TRUE

```
get_pd_assign_value(pd) %>% select(-starts_with('line')) %
```

	col1	col2	id	parent	token	terminal
4	6	10	4	6	SYMBOL_FUNCTION_CALL	TRUE
6	6	10	6	12	expr	FALSE
12	6	15	12	14	expr	FALSE
5	11	11	5	12	‘(’	TRUE
7	12	14	7	8	NUM_CONST	TRUE
8	12	14	8	12	expr	FALSE
9	15	15	9	12	‘)’	TRUE

```
get_pd_assign_variable(pd) %>% select(-starts_with('line'))
```

	col1	col2	id	parent	token	terminal	text
1	1	1	1	3	SYMBOL	TRUE	x
3	1	1	3	14	expr	FALSE	

documentation

```r
library(documentation)
x <- new('Documentation'
        , title       = "Example documentation"
        , author      = person('Andrew', 'Redd')
        , description = "This is used to demonstrate the cl
                        "system for the documentation packa
        , keywords    = "documentation"
        )
```

```r
str(x)
```

```
## Formal class 'Documentation' [package "documentation"] w
##   ..@ author     :Class 'person'  hidden list of 1
##   .. ..$ :List of 5
##   .. .. ..$ given  : chr "Andrew"
##   .. .. ..$ family : chr "Redd"
##   .. .. ..$ role   : NULL
##   .. .. ..$ email  : NULL
##   .. .. ..$ comment: NULL
##   ..@ title      : chr "Example documentation"
##   ..@ description:Formal class 'FormattedText' [package
##   .. .. ..@ .Data: chr "This is used to demonstrate the
##   ..@ references : list()
##   .. ..- attr(*, "class")= chr "bibentry"
##   ..@ seealso    :Formal class 'FormattedText' [package
##   .. .. ..@ .Data: chr(0)
##   ..@ examples   :Formal class 'Prose' [package "documen
##   .. .. ..@ .Data: list()
```

```r
toRd(x) %>% cat(sep='\n')
```

```
## \author{Andrew Redd}
## \title{Example documentation}
## \description{This is used to demonstrate the class syste
## \keyword{documentation}
```

Something a bit more interesting

```r
hw <- function(){print("hello world")}
documentation(hw) <- function_documentation('hw', title = '
documentation(hw) %>% toRd %>% unlist %>% cat(sep='\n')
```

```
## \name{hw}
## \usage{hw()}
## \value{NA}
## \title{the standard Hello world}
```

Can your documentation do this?

```
make_hello <- function(what = 'world'){
  documented(function(){
    print(paste("hello", what))
  }, name  = paste0("hello_", what)
   , title = sprintf("the dynamic hello_%s example", what)
   , description = "Prints hello" %\% what %\%"to the conso
                   "Used to show the power of dynamic docum
   )
}
hello_world <- make_hello()
hello_belgium <- make_hello('Belgium')

hello_belgium()
```

```
## [1] "hello Belgium"
```

```r
documentation(hello_belgium) %>% toRd %>% unlist %>% cat(se
```

```
## \name{hello_Belgium}
## \usage{hello_Belgium()}
## \value{NA}
## \title{the dynamic hello_Belgium example}
## \description{Prints hello Belgium to the console.
## Used to show the power of dynamic documentation.}
```

Test extraction

Example R code in “./R/hello.R”

```r
hello_belgium <- make_hello('Belgium')
if(FALSE){#@testing
  expect_output(hello_belgium(), "hello Belgium")
}
```

We run `extract_tests()` then . . .

in "./tests/testthat/test-hello.R" we will have

```
#! This file was automatically produced by documentation o
#! changes will be overwritten.
context('tests extracted from file `./R/hello.R`')
test_that("hello_belgium", {#@testing
  expect_output(hello_belgium(), "hello Belgium")
})
```

Something still to finish

Relational Tags

```
hw <-
function( greet = "Hello" #< A greeting
        , who   = "World" #< who to greet.
        ){
    #! A more complicated hello world
    print(paste(greet, who))
    #< Called for the side effect of printing,
    #^ but returns the pasted arguments invisibly.
}
```

State in summary

Documentation Classes	done..*ish*
`parsetools`	functionally Complete
test extraction	done
Relational Tags	Infrastructure in place
	Needs to integrate with Roxygen
Input Forms	Lots to do
Output Forms	
Rd	in place, could use improvement
markdown	lots to do
Others	not started

how people can help and get involved.

- Make a github request for a feature.
- Code one of the missing functionalities
  - code it up
  - test the heck out of it
  - make a pull request showing

`https://github.com/RDocTaskForce/`

- Join the mailing list

`https://lists.r-consortium.org/mailman/listinfo/rconsortium-wg-dtf`

Acknowledgements

- Special Thanks to the R Consortium for the funding for this project
- Thanks to the members and participants of the RDTF.
  - Richard Calaway
  - Sarah Goslee
  - Michael Lawrence
  - Martin Maechler
  - Duncan Murdoch
  - Kirill Muller