Using HTML Comments to Facilitate Colaboration with Git

R. Mark Sharp

2021-02-16

Introduction

Retention of collaborative interactions that appear as back and forth comments within source documents of a project is a valuable source of history and subsequent education and process improvement. However, it is often the case that at the end of the development phase those interactions need to be removed before sending the source files to further review or production. This tutorial provides one method and a set of tools to both collaborate via HTML comments and remove comments selectively when needed.

Collaboration

HTML comments can be inserted into an RMarkdown document as a means of communicating with others or making a note to yourself. The workflow illustrated herein uses a text label immediately after the beginning characters of the HTML comment (i.e., `<!--`). White space, including blank lines are ignored. Thus, any of the following will be seen as a comment.

```
`<!-- This is a comment without a useful label -->`
`<!-- RMS This is a one line comment that has my intitials as its label.-->`
`<!--
RMS Comments can span multiple lines and later can be entirely removed by
either selecting to remove all comments or only those comments that have
labels you provide as a character vector to the "label" parameter
-->`
```

Collaboration will usually involve suggestions and perhaps corrections.

```
  xt_print(all_unusual_wts_sex,
           caption = stri_c(common_name, ":  ", animal_count, " ", sex_str,
                            " with weights outside the ",
                            signif(conf_int * 100, 3),
                            "\\% predicted range."),
           label = stri_c("tbl:", arc_species_code, "-", sex),
           format.args = list(big.mark = ",", decimal.mark = "."),
           size = size)

`<!-- RMS I added some formating code (format.args) in the call to xt_print
     please look at the output to see whether or not you prefer that formating
     decision.-->`
```

The collaborator (TJH) can add his own comment or add to the original to inform RMS that the change was considered.

```
`<!-- RMS I added some formating code (format.args) in the call to xt_print
     please look at the output to see whether or not you prefer that formating
     decision.
```

```
    *** TJH I like the change as some of the numbers are over 10^5
 -->`
```

Alternatively, TJH may decide to keep his response separate from the original comment by adding his own separate comment.

```
`<!-- RMS I added some formating code (format.args) in the call to xt_print
     please look at the output to see whether or not you prefer that formating
     decision.
 -->`
`<!-- TJH I like the change as some of the numbers are over 10^5
 -->`
```

We recommend the first convention as it keeps related comments together and still allows identification of participants within the conversation.

Later, reasons for selecting one convention over the other will be discussed again when removal of comments is described.

Review

HTML comments can also be used as a part of a review process. This is not conceptually different that other forms of collaboration but the need to indicate acceptance or rejection of suggestions and final approval is demonstrated in the following versions of text.

```
  xt_print(bad_sample_dates_df, caption = "Bad Sample Dates",
           label = "tbl:bad-sample-dates", type = type, ...)


`<!-- RMS The caption reads like a title. Consider using a caption that allows
the table to be understood without forcing the reader to go find the
narrative that describes the table contents.
-->`
```

The document author can respond to the request by simply editing the code. However, editing to comment allows the review to quickly see that the concern was acknowledged and addressed.

```
    if (nrow(bad_sample_dates_df) == 1) {
      caption <- stri_c(
        "There was 1 bad sample date identified where the animal was not
        present on the date indicated.")
    } else {
      caption <- stri_c(
        "There were ", nrow(bad_sample_dates_df), " bad sample dates identified
        where the animals were not present on the dates indicated.")
    }

    xt_print(bad_sample_dates_df, caption = caption,
             label = "tbl:bad-sample-dates", type = type, ...)



`<!-- RMS The caption reads like a title. Consider using a caption that allows
the table to be understood without forcing the reader to go find the
narrative that describes the table contents.
*** TJH I added a more helpful dynamically generated caption so that the text
```

```
reflects the number of bad dates shown.
-->`
```

The reviewer can then note that the change was seen and accepted.

```
`<!-- RMS The caption reads like a title. Consider using a caption that allows
the table to be understood without forcing the reader to go find the
narrative that describes the table contents.
*** TJH I added a more helpful dynamically generated caption so that the text
reflects the number of bad dates shown.
*** RMS Nicely done. I like the dynamically generated caption.
accepted 20210215
-->`
```

Locating Comments

Retention of comments during the document development phase is helpful so that decisions made earlier are not forgotten with the result of time being wasted rethinking earlier topics of discussion.

There are several functions that can be used to produce various inventories of HTML comments within RMarkdown source files.

- **get_html_comment_text_lines_and_labels_from_files**

Takes a vector of files[1] and labels to retrieve and returns a dataframe with full file paths, the base file names, starting line number of each comment, the end line number of each comment, and the identifying labels. This dataframe is ordered by path, label, and starting line number of the comment.

The default value of the `label` argument is `""` when no definition is provided.

```
files = system.file("testdata","find_html_comment_test_file.Rmd",
                        package = "rmsutilityr")
html_comment_lines_and_labels <-
  get_html_comment_text_lines_and_labels_from_files(files)

caption <-
  knitr:::escape_latex(stri_c("Output of the ",
         "get_html_comment_text_lines_and_labels_from_files ",
         "function includes all comments when no 'label' parameter is ",
         "provided."))


 xtable(html_comment_lines_and_labels[ , c("file", "comment_label",
                                           "comment_start_line",
                                           "comment_end_line")],
     booktabs = TRUE, caption = caption) %>%
   xtable2kable() %>%
   kable_styling(latex_options = "striped")
```

This same function can be used to review the text of comments from selected collaborators.

```
files = system.file("testdata","find_html_comment_test_file.Rmd",
                        package = "rmsutilityr")
html_comment_lines_and_labels <-
  get_html_comment_text_lines_and_labels_from_files(files, label = "RMS")
```

[1]This example uses a single file but a character vector with fully qualified file names is expected.

	file	comment_label	comment_start_line	comment_end_line
4	find_html_comment_test_file.Rmd	I	26	26
1	find_html_comment_test_file.Rmd	RMS	16	18
5	find_html_comment_test_file.Rmd	RMS	30	32
6	find_html_comment_test_file.Rmd	RMS	34	35
3	find_html_comment_test_file.Rmd	SRR	25	25
2	find_html_comment_test_file.Rmd	TJH	22	23

Table 1: Output of the get_html_comment_text_lines_and_labels_from_files function includes all comments when no 'label' parameter is provided.

```
caption <-
  knitr:::escape_latex(stri_c("Output of the ",
         "get_html_comment_text_lines_and_labels_from_files ",
         "function includes text of comments from selected ",
         "comment labels."))


 xtable(html_comment_lines_and_labels[ , c("file", "comment_label",
                                           "comment_text")],
     booktabs = TRUE, caption = caption) %>%
   xtable2kable() %>%
   kable_styling(latex_options = "striped", font_size = 9) %>%
   column_spec(4, width = "20em")
```

	file	comment_label	comment_text
1	find_html_comment_test_file.Rmd	RMS	This is a comment that I want to notice later. To help others see it, I decided to write more words than are needed.
2	find_html_comment_test_file.Rmd	RMS	Let's count this one
3	find_html_comment_test_file.Rmd	RMS	I am counting this too.

Table 2: Output of the get_html_comment_text_lines_and_labels_from_files function includes text of comments from selected comment labels.

Deleting Comments

As stated earlier, in some workflows it is an advantage to remove collaborators' and reviewers' comments from the final document to clean up the presentation and to prevent unintended influence on subsequent readers of the source RMarkdown document.

This can be done by providing a character vector of full or relative path names and a directory to place the edited files in to using the `write_files_after_deleting_selected_comments` function.

```
files = system.file("testdata","find_html_comment_test_file.Rmd",
                    package = "rmsutilityr")
new_files <-
  write_files_after_deleting_selected_comments(new_path = tempdir(),
                                               files, label = "RMS")
new_files
```

```
## [1] "/var/folders/y3/5z1skj6s5tq0pktmsq6x80v80000gn/T//RtmpVechFl/find_html_comment_test_file.Rmd"
```

You can check for comments in your files quickly with this helper function[2]

```
count_selected_comments <- function(files, label = "") {
  comment_count <- 0
  for (file in files) {
    lines <- readLines(file)
    lines_and_labels <- return_html_comment_text_lines_and_labels(lines, label)
    comment_count <- comment_count + length(lines_and_labels[[1]])
  }
  comment_count
}

count_selected_comments(files, label = "RMS")
```

```
## [1] 3
```

```
count_selected_comments(new_files, label = "RMS")
```

```
## [1] 0
```

```
count_selected_comments(new_files, label = "")
```

```
## [1] 3
```

You can see the differences in the before and after versions of a file with the `Rdiff` and `diffr` functions.

```
# There is only one file in `files` and in `new_files` so the subsetting is
# unnecessary.
tools::Rdiff(from = files[1], to = new_files[1], useDiff = TRUE, Log = TRUE)
```

```
## $status
## [1] 1
##
## $out
##  [1] "16,18d15"
##  [2] "< <!-- RMS This is a comment that I want to notice later. To help others see it,"
##  [3] "<  I decided to write more words than are needed."
##  [4] "< -->"
##  [5] "30,32d26"
##  [6] "< <!-- RMS "
##  [7] "< Let's count this one"
##  [8] "< -->"
##  [9] "34,35d27"
## [10] "< <!--"
## [11] "< RMS I am counting this too. -->"
```

I prefer the HTML output of the `diffr` package. Its use is very similar[3].

```
library(diffr)
diffr(files[1], new_files[1])
```

[2] A commented version of the same helper function is included in this package.
[3] HTML output is not shown.