How to Create Your Own Chunk Options in R Markdown

Ulrik Lyngs

ABSTRACT

ACM Reference Format:
Ulrik Lyngs. 2019. How to Create Your Own Chunk Options in R Markdown. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

For the 2019 ACM CHI conference I wanted to go all in on reproducibility and write our paper submission completely in R Markdown, and just tweak the ACM's LaTeX template so that I could use it as a template for R Markdown to use when outputting to PDF. It worked in the end (find PDF preprint and .Rmd source file for our paper here, an R package that makes it easy here, and a blog post about the process here), but along the way I found that I was missing some chunk options to be able to keep my paper reproducible.

For example, in an update to their LaTeX template, the ACM now wanted all figures to include a description to improve accessibility for visually impaired readers. In LaTeX, this was supposed to be accomplished by adding `\Description{This is a figure description}` inside the relevant figure environment. How should I handle this while staying within my R Markdown-based workflow?

One option would be to add these descriptions manually in LaTeX as the last step before finishing the paper - that is, I could add `keep_tex: true` in my YAML header, then manually adjust the **.tex** file for our paper and then re-generate a PDF from this LaTeX file. This would work but it was also a bad option that would be error-prone: if I discovered some mistake that would need fixing in the R Markdown source file and require me to recompile and re-create the **.tex** file, then I would have to add all the descriptions again…

The much better solution would be if there simply existed a chunk option 'description' that I could set directly in the R Markdown source file with something like ```` ```{r my_figure, description="This is a figure description"} ````, and which would then automatically add `\Description{This is a figure description}` when knitting to PDF. Fortunately, it is possible to just create new chunk options yourself!

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 GETTING A GRIP ON KNIT OUTPUT HOOKS

Knitr, which does the first part of R Markdown's work under the hood, provides 'hooks' which are "customisable functions to run before/after a code chunk, tweak the output, and manipulate chunk options". We will work with 'output hooks' which are used to customise and polish *raw* output from chunks. There are 8 different kinds of output hooks that can grab different types of output; for our purpose we will modify the `chunk` output hook which grab all the output of a chunk.

A chunk output hook takes the form `function (x, options)` where `x` is a character string of the output and `options` is a list of the chunk options.

To modify them, do this:

```
knit_hooks$set(chunk = function(x, options) {
  # some code to modify chunk output hooks here
})
```

The stupidest possible thing to do

The stupidest possible thing we might do do would be to get all chunks to output "Hello, world!".

Here is a random plot:

```
plot(pressure)
```

Then we modify the chunk output hook to return 'hello world': First, let's store the current configuration of the chunk output hook, and have a look at it as well:

```
## function (x, options)
## {
##   x = gsub(paste0("[\n]{2,}(", fence, "|   )"), "\n\n\\1",
##         x)
##     x = gsub("[\n]+$", "", x)
##     x = gsub("^[\n]+", "\n", x)
##     if (isTRUE(options$collapse)) {
##      x = gsub(paste0("\n([", fence_char, "]{3,})\n+\\1(",
##         tolower(options$engine), ")?\n"), "\n", x)
```

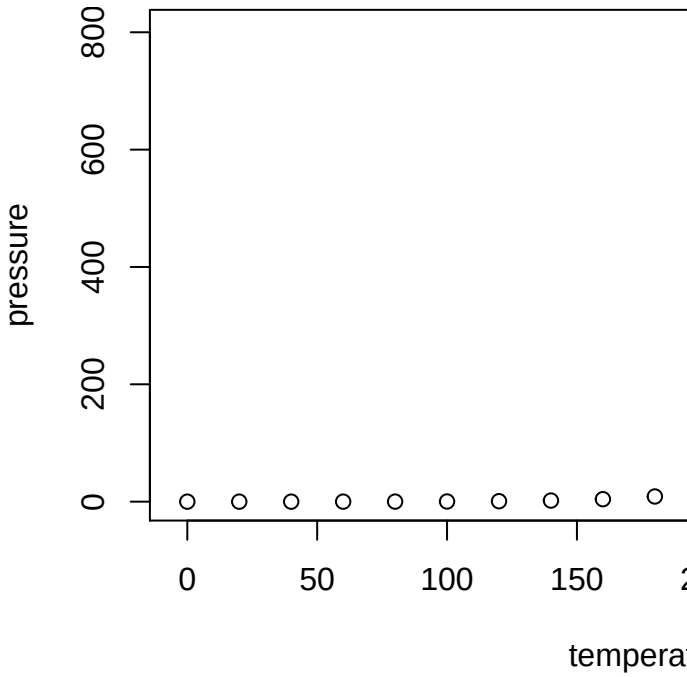


Figure 1: A random plot

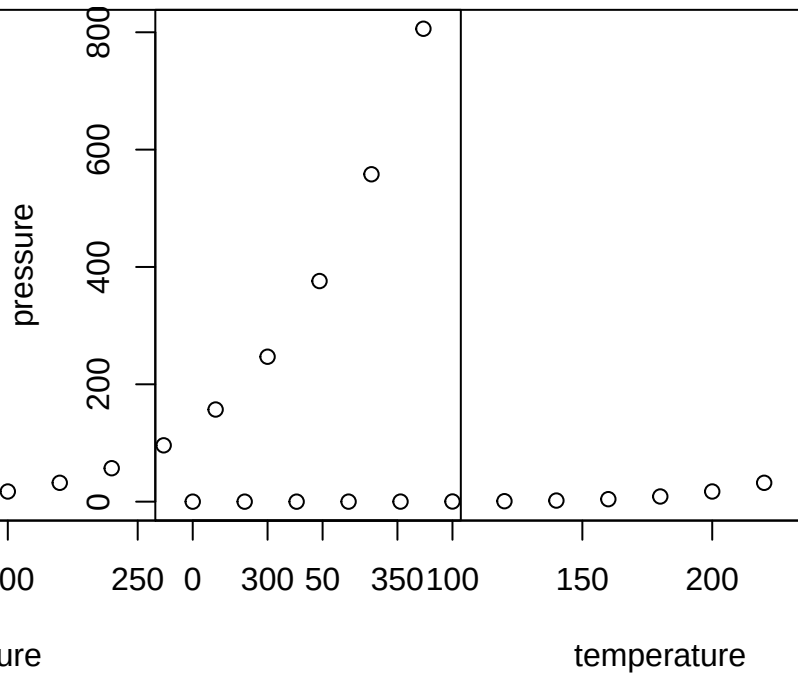


Figure 2: A random plot

```
##     }
##     if (is.null(s <- options$indent))
##         return(x)
##     line_prompt(x, prompt = s, continue = s)
## }
## <bytecode: 0x7f88d16720e8>
## <environment: 0x7f88d16842e0>
```

Then modify it:

```
knit_hooks$set(chunk = function(x, options) {
  return("Hello, world!")
})
```

Now if we try do draw the random plot again we simply get:

Hello, world!

Creating a new chunk option

To make this minimally more useful, imagine if this only happened if we set a chunk option `hello`.

Let's first set the output hook back to what it was:

```
knit_hooks$set(chunk = hook_chunk)
```

Now let's modify it again:

```
knit_hooks$set(chunk = function(x, options) {
  if (!is.null(options$hello)) {
    return("Hello, world!")
  } else {
    return(hook_chunk(x, options))
  }
})
```

Let's check if this works. Here's our random plot again (Figure 2):

```
plot(pressure)
```

And here it is with chunk option `hello=TRUE`: Hello, world!

2 EXAMPLE USE: ADDING \DESCRIPTION{} TO LATEX FIGURE OUTPUT

Finally, let's create an actually useful chunk option: the option to add a `\Description{}` to figures in PDF output, which is what I needed. For good measure, let's start by resetting the chunk output hook to its original state:

```
knit_hooks$set(chunk = hook_chunk)
```

The problem we're trying to solve is this: If we knit to PDF and set `keep_tex = TRUE` in the YAML header, we see that our random plot included in the **.tex** file in this way:

```
\begin{figure}
\centering
\includegraphics{how-to-create-your-own-chunk-options-in-r-
\caption{A random plot}
\end{figure}
```

What we want in our **.tex** file is this:

```
\begin{figure}
\centering
\includegraphics{how-to-create-your-own-chunk-options-in-r-markdown_files/figure-latex/random-plot-1.pdf}
\Description{A scatter plot of an exponentially growing curve}
\caption{A random plot}
\end{figure}
```

We would like this to be easily done via a chunk option `description`. What we want to do is search through the usual LaTeX output with a regular expression and insert `\Description{A scatter plot of an exponentially growing curve}` after the call to `\includegraphics`.

Note that if you want to follow along with this example, you must output to PDF via an ACM LaTeX template in which `Description` has been defined as a control sequence. If you go to github.com/ulyngs/chi-proc-rmd-template you will find the relevant files in **chi-proc-rmd-template/inst/rmarkdown/templates/acm_chi_proc/skeleton/**.

This should work:

```
# store the usual chunk output function
hook_chunk = knit_hooks$get('chunk')

knit_hooks$set(chunk = function(x, options) {
  regular_output = hook_chunk(x, options)

  # if there is a description
  if (!is.null(options$description)) {
    # include the following LaTeX - \\1 refers to the regex grouping
    latex_include <- paste0("\\1\\\\Description\\{", options$description, "\\}")

    # search and replace in the output
    gsub('(\\\\includegraphics[^}]+})', latex_include, regular_output)

  } else {

    # if there isn't a description just return unmodified
    return(regular_output)  # pass to default hook

  }
})
```

So now let's try with these chunk options for Figure 3:

```
```{r my-description, echo=TRUE, fig.cap="A random plot", description="A scatter plot of an exponentially growing curve"}

plot(pressure)
```

Uh-oh it actually doesn't work, our LaTeX output still shows like this:

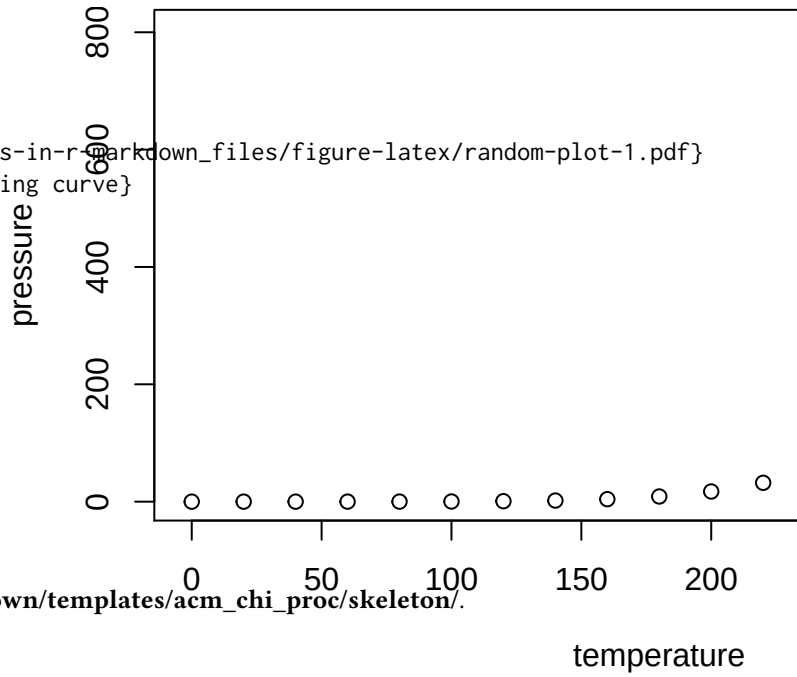


Figure 3: A random plot

Turns out that for this to work we have to be explicit with **knitr** that the figure output is intended to be treated as LaTeX when we're trying to modify output (see Yihui Xie's explanation for why here: github.com/yihui/knitr/issues/1464).

So when we want to modify LaTeX output in our R Markdown document, we need to add this in out setup chunk:

```
if (knitr::is_latex_output()) knitr::knit_hooks$set(plot =
```

Let's try again:

```
plot(pressure)
```

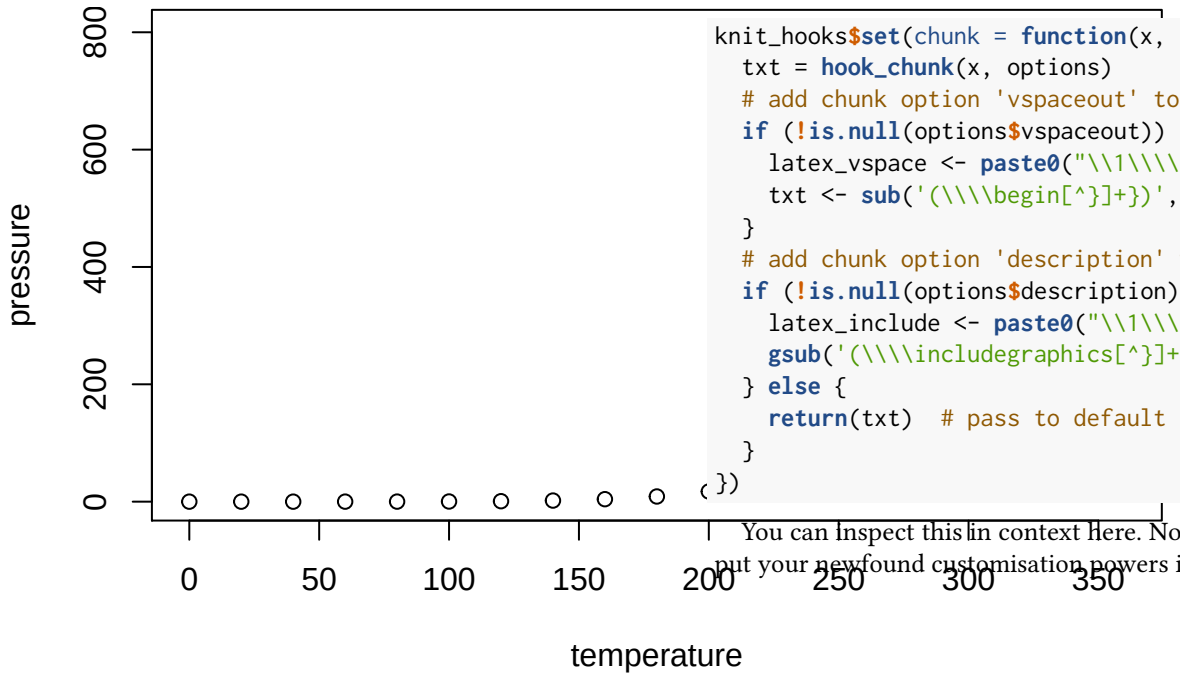
Yup the LaTeX generated for Figure 4 now looks as we wanted:

```
\begin{figure}
\includegraphics{how-to-create-your-own-chunk-options-in-r-
\end{figure}
```

3 CONCLUSION

It is super powerful to be able to define your own chunk options. In the R Markdown template for CHI proceedings, I also create a chunk option that allows chunks to be positioned vertically in PDF output by inserting the LaTeX commmand `\vspace`, so that the relevant part of the initial setup chunk looks like this:

```
# create additional chunk options
hook_chunk = knit_hooks$get('chunk')
```
```




```
knit_hooks$set(chunk = function(x, options) {
  txt = hook_chunk(x, options)
  # add chunk option 'vspaceout' to position chunks vertica
  if (!is.null(options$vspaceout)) {
    latex_vspace <- paste0("\\1\\\\vspace\\{", options$vspa
    txt <- sub('(\\\\begin[^}]+})', latex_vspace, txt)
  }
  # add chunk option 'description' which adds \Description{
  if (!is.null(options$description)) {
    latex_include <- paste0("\\1\\\\Description\\{", option
    gsub('(\\\\includegraphics[^}]+})', latex_include, txt)
  } else {
    return(txt)  # pass to default hook
  }
})
```

You can Inspect this in context here. Now go ahead and put your newfound customisation powers into practice!

Figure 4: A random plot