How to tweak Rmarkdown output programmatically: Part 1 of a series . . .

Richard Careaga

2015-11-11

RMarkdown and knitr encourage a kind of arms race towards improved presentation of R results. In the face of console-style output, such as

```
##
##                     Disputed Firearm Knife
##   Death in custody       0.0    0.00  0.00
##   Gunshot                0.3   47.92 14.24
##   Struck by vehicle      0.0    0.15  0.00
##   Taser                  0.0    0.00  0.15
##   Unknown                0.0    0.00  0.00
##   Sum                    0.3   48.07 14.39
##
##                         No
##   Death in custody   3.86
##   Gunshot            9.50
##   Struck by vehicle  3.26
##   Taser              4.60
##   Unknown            0.45
##   Sum               21.66

##
##                     Non-lethal firearm Other
##   Death in custody                0.00  0.15
##   Gunshot                         3.26  5.19
##   Struck by vehicle               0.00  0.00
##   Taser                           0.00  0.30
##   Unknown                         0.00  0.00
##   Sum                             3.26  5.64
##
##                     Unknown
##   Death in custody     0.15
##   Gunshot              2.97
##   Struck by vehicle    0.00
##   Taser                0.00
##   Unknown              0.00
##   Sum                  3.12
```

we demand to know: *Where's the glory in **that***?

No, we aim for *publication quality* tables, for which we turn to LaTeX, and for that we are blessed with **xtable**.

Table 1: 2

	Disputed	Firearm	Knife	No
Death in custody	0.00	0.00	0.00	3.86
Gunshot	0.30	47.92	14.24	9.50
Struck by vehicle	0.00	0.15	0.00	3.26
Taser	0.00	0.00	0.15	4.60
Unknown	0.00	0.00	0.00	0.45
Sum	0.30	48.07	14.39	21.66

Close, but no cigar.[1]

In the first place, although we have the **zero.print** = "." in **addmargins** to surpress all the *0.00* entries, we don"t in **xtables**. While a **Wickham** may be able to dash off a package before breakfast to fix this, mere mortals cannot. It is also an imposition on our fellow mere mortals to ask them to accompany our attempts at literate programming with new **devtool:install** instructions for our R code to fix this.[2] Additionally, we would like percentage signs"%", which must be escaped for LaTeX with a backslash,"\%."

Doing this by hand is both tedious and error prone.[3] Doing post-processing of an intermediate **md** file is possible but offends the workflow god. Enter **pandoc**. This is a Haskell package that does the behind-the-scenes heavy lifting for RMarkdown. It has been called the Swiss Army Knife of format conversion.

Buried in **help(rmarkdown)** lies the secret sauce to pass to **pandoc** the command line arguments that will sneak an invisible filter through which will tickle its imput to pummel the raw output of **xtable** into presentable LaTeX and, thus, ultimately, into polished tables that don't require hand tweaking.

In the forepart *yaml*, add

```
output:
  rmarkdown::tufte_handout:
    pandoc_args: [
      "--filter", "/Users/rc/bin/style1"
         ]
```

where *style1* is a program that reads from *stdin* and writes to *stdout*, parsing the output of *pandoc* and feeding it back into *pandoc* for further rendering.

In a Platonic world, however, we would want to take the edit changes needed to transform the ***first*** table below into the ***second***

[1] When, please, may we expect the embargo on genuine Cuban cigars to be lifted, please, Mr. President?

[2] R has many virtues, string manipulation not necessarily included.

[3] Why not just go back to Excel?

from the metadata in the document itself.

	Disputed	Firearm	Knife	No
Death in custody	0.00	0.00	0.00	3.86
Gunshot	0.30	47.92	14.24	9.50
Struck by vehicle	0.00	0.15	0.00	3.26
Taser	0.00	0.00	0.15	4.60
Unknown	0.00	0.00	0.00	0.45
Sum	0.30	48.07	14.39	21.66

Table 2: Causes of 674 civilian deaths by type of weapon carried by civilian, part 1

	Disputed (%)	Firearm (%)	Knife (%)	None (%)
Death in Custody	.	.	.	3.86
Gunshot	0.30	47.92	14.24	9.50
Struck by Vehicle	.	0.15	.	3.26
Taser	.	.	0.15	4.60
Unknown	.	.	.	0.45
Sum	0.30	48.07	14.39	21.66

Table 3: Causes of 674 civilian deaths by type of weapon carried by civilian, part 1

This proves not to be as straightforward as might be desired. While *pandoc*'s API exposes a serialization of the abstract syntax tree (AST) that it uses to internally represent the source document for transformation into the target document and allows free access to anything in the *yaml* formatter for use in a document template, the functionality to directly access the *yaml* fields **for anything else** is supressed. *Unless you want to fork the pandoc project* and refactor the code (in Haskell) to permit this, keeping your programmatic edits in the formatter is not in the cards, at least to the dealer at this table.

Besides, it finally penetrated my thick skull,

> Filtering through a programmatic process is a noble ideal when all of the kinks have been worked out, especially when you are cranking the handle on a production process. **But** that's not the world we usually inhabit. Grab, grab, scrub, scrub, tinker, tinker, output, output, tinker, tinker. Repeat.

Being brutally honest with myself, I don't embark upon any sort of the types of data analyses that I routinely do with the resolve of testing all the edge cases for my output format. *Type, type, render, render, tweak, tweak* is what I do. Only if I knew that I had a huge number of chunks to output and process consistently would I do that.

But what about the intermediate case between the one off and the factory method where you have sufficient iterations to make one-off

hand editing problematic but too few to make a full fledged bot a good use of your time?

For this, we need to bring the prompt back into our workflow and use the hierloom legacy of standard command line or bespoke tools to bring the power of *stdin | stdout* to bear.

For **R**, and, therefore, for **Rmd**, we have

```
> system()
```

which allows us to send **R** objects off to be processed at less heartache and brought back into the workflow.

AND THAT IS WHY, the second thought that permeated my skull was

> The closer to the source and to the eyes on the output, the better off you are.

A word, however, on an important assumption: The code has been already composed and tested and the analysis completed before the author sets out to prepare an Rmd file for draft and final rendering. *Premature pretty printing* is the sin that tempts the otherwise enlightened to ape the office application substitution of decoration for thought.[4]

[4] In the days of greenbar, no one worried about how snazzy a thick stack of folded printout looked. There were sensible defaults, hard to override, and you tended to worry more about substance. When I left the Fortune 50 world, it seemed that most reports were far more concerned with appearance than substance.

SO, HERE'S HOW I USED SYSTEM() to tweak my **xtable** rendering.

First, I adapted Haskell code that I had been working on as a **pandoc** filter. It ain't what you could call elegant, and there is no special magic to the language. You can substitute **sed, awk, Perl, Python, Ruby, C, C++, Go** or any other language or combination of languages and command line utilities you like.[5]

[5] Windows users: I am unable to answer any questions related to how this could be done without a *nix. It's been a decade since I've owned a box under Windows.

The **only** thing to be said in favor of the style of this bit of code by a rank beginner is that *it works notwithstanding the deficit of elegance.*

```
{- --style1.hs 2015-11-09 23:12 v 1.0 by Richard Careaga
   --adjust column alignment for row names, title case required words in row names
   --replace columns with revised names and additional header row giving units
   --NB: plauge of blackslashes resolved by inspection
   --zero.print = TRUE to replace "0.00" with 0.0
   --usage: cat FILENAME | ./style1
   --R: see discussion in accompanying text
   --TODO: replace lambdas, map it to a structure of k, v pairs, get k, v pairs from a
   --      metadata block in the source file and otherwise prevent milk from coming out the
   --      noses of real Haskell programmers
-}

import Text.Regex
```

```
custody :: String
custody = "custody"

uCustody :: String
uCustody = "Custody"

vehicle :: String
vehicle = "vehicle"

uVehicle :: String
uVehicle = "Vehicle"

numZeros :: String
numZeros = " 0.00"

dotZeros :: String
dotZeros = "."

rightJustify :: String
rightJustify = "{r"

leftJustify :: String
leftJustify = "{l"

noLabel :: String
noLabel = ".label.2."

emptyString :: String
emptyString = ""

oldHeader :: String
oldHeader = " & Disputed & Firearm & Knife & No \\\\"

newHeader :: String
newHeader = " & Disputed & Firearm & Knife & None \\\\\\\\\n&(\\\\%)&(\\\\%)&(\\\\%)&(\\\\%)\\\\"

main :: IO()
main = interact ((\ f -> subRegex (mkRegex rightJustify) f leftJustify) .
                 (\ f -> subRegex (mkRegex oldHeader)    f newHeader)   .
                 (\ f -> subRegex (mkRegex custody)      f uCustody)    .
                 (\ f -> subRegex (mkRegex vehicle)      f uVehicle)    .
                 (\ f -> subRegex (mkRegex numZeros)     f dotZeros)    .
                 (\ f -> subRegex (mkRegex noLabel)      f emptyString))
```

```
{-- stdin
% latex table generated in R 3.2.1 by xtable 1.7-4 package
% Mon Nov  9 21:01:01 2015
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
  \hline
 & Disputed & Firearm & Knife & No \\
  \hline
Death in custody & 0.00 & 0.00 & 0.00 & 3.86 \\
  Gunshot & 0.30 & 47.92 & 14.24 & 9.50 \\
  Struck by vehicle & 0.00 & 0.15 & 0.00 & 3.26 \\
  Taser & 0.00 & 0.00 & 0.15 & 4.60 \\
  Unknown & 0.00 & 0.00 & 0.00 & 0.45 \\
  Sum & 0.30 & 48.07 & 14.39 & 21.66 \\
   \hline
\end{tabular}
\caption{Causes of 674 civilian deaths by type of weapon carried by civilian, part 1}
\label{2}
\end{table}

-- stdout
% latex table generated in R 3.2.1 by xtable 1.7-4 package
% Mon Nov  9 21:01:01 2015
\begin{table}[ht]
\centering
\begin{tabular}{lrrrr}
  \hline
 & Disputed & Firearm & Knife & None \\
 &  (\%)    & (\%)    &  (\%) & (\%) \\
  \hline
Death in Custody &. &. &. & 3.86 \\
  Gunshot & 0.30 & 47.92 & 14.24 & 9.50 \\
  Struck by Vehicle &. & 0.15 &. & 3.26 \\
  Taser &. &. & 0.15 & 4.60 \\
  Unknown &. &. &. & 0.45 \\
  Sum & 0.30 & 48.07 & 14.39 & 21.66 \\
   \hline
\end{tabular}
\caption{Causes of 674 civilian deaths by type of weapon carried by civilian, part 1}

\end{table}
--}
```

IT SHOULD BE MAINLY OBVIOUS what this code does. The plague of backslashes is due to the use of that character as an escape character in both \LaTex\ and R.

The way that I am now using this is to modify a copy of the output of **table** through a **system()** command, thusly

```
library(xtable)
options(xtable.comment = FALSE)
options(xtable.booktabs = TRUE)
load("cod.Rda") # previously saved data frame
sumcod <- cod[6,9] # n observations
round(cod[,1:4]/sumcod*100,2) # first part of the wide table
{r, results="hide", echo=FALSE} # in Rmd chunk to surpress but capture table
table <- print(xtable(cod[,1:4]/sumcod*100,2, caption = paste("Causes of", sumcod,
+ "civilian deaths by type of weapon carried by civilian, part 1")))
{r, results="asis", echo=FALSE} # in Rmd chunk to filter output of xtable
table <- cat(system("~/bin/style1", input =  table, intern = TRUE))
```