Using `modelsummary` in an Rmarkdown document

This file illustrates how to use `modelsummary` to produce PDF, HTML, and RTF (Microsoft Word-compatible) files from a single Rmarkdown document.

Header

At the top of every Rmarkdown document, there is a "header" enclosed between two "—" lines, and written in YAML markup syntax. The first important thing to note is that in the current document, we have including the following code in the header:

```
header-includes:
- \usepackage{booktabs}
- \usepackage{xcolor}
```

These lines are required for PDF documents with `modelsummary` tables. Under the hood, Rmarkdown uses LaTeX to create PDF documents. All the tables produced by `modelsummary` require the LaTeX `booktabs` package. In addition, features like colors require loading additional LaTeX packages `xcolor`. The `kableExtra` documentation explains which LaTeX packages may be required for different use cases.

A first table

To begin we estimate three linear regression models using the `mtcars` data:

```
library(modelsummary)
library(kableExtra)
library(gt)
models <- list()
models[[1]] <- lm(mpg ~ cyl, mtcars)
models[[2]] <- lm(mpg ~ cyl + drat, mtcars)
models[[3]] <- lm(hp ~ cyl + drat, mtcars)
```

Then, we load the `modelsummary` package and call the `msummary` function:

```
msummary(models,
         title = 'Determinants of Car Features',
         statistic = 'conf.int')
```

Without any modification, this code will produce a nice table, whether the document is compiled to PDF, HTML, or RTF.[1] In fact, tables produced using *any* of the `modelsummary` built-in options and arguments should compile correctly in all three formats.

One major benefit of `modelsummary` is that regression tables can be customized using the powerful `gt` and `kableExtra` libraries. To achieve this, we *post-process* the output of `msummary()` using functions from the `gt` or `kableExtra` packages. Achieving this in an Rmarkdown file requires some slight, but very easy, tweaks to our code. The next two sections explain what those tweaks are.

[1] The Rmarkdown output in the header must be set to either "pdf_document", "html_document", or "rtf_document".

Table 1: Determinants of Car Features

	Model 1	Model 2	Model 3
(Intercept)	37.885	28.725	-215.768
	[33.649, 42.120]	[13.197, 44.252]	[-396.489, -35.048]
cyl	-2.876	-2.484	39.012
	[-3.534, -2.217]	[-3.398, -1.569]	[28.367, 49.657]
drat		1.872	33.662
		[-1.183, 4.927]	[-1.893, 69.218]
Num.Obs.	32	32	32
R2	0.726	0.740	0.728
Adj.R2	0.717	0.722	0.709
AIC	169.3	169.6	326.7
BIC	173.7	175.5	332.6
Log.Lik.	-81.653	-80.809	-159.348

Output formats: `gt` vs. `kableExtra`

The two external packages that `modelsummary` uses to create tables are `gt` and `kableExtra`. These packages are fantastic, but they each have (minor) disadvantages. In particular,

- `gt`'s LaTeX support is not yet mature, and creating PDF files often breaks in Rmarkdown.
- `kableExtra` does not support RTF output.

In the rest of this example file, we will demonstrate how to use `gt` and `kableExtra` functions to customize tables. However, we will not create tables using `gt` in PDF documents, and we will not create tables using `kableExtra` in RTF documents. To achieve this, we detect the Rmarkdown output format and create of two boolean variables:

```
is_rtf <- knitr::opts_knit$get("rmarkdown.pandoc.to") == 'rtf'
is_latex <- knitr::opts_knit$get("rmarkdown.pandoc.to") == 'latex'
```

In chunks with `gt` code, we set `eval=!is_latex`. In code chunks with `kableExtra` code, we set `eval=!is_rtf`.

Customizing tables with `gt`

We can use functions from the `gt` package to post-process and customize a `modelsummary` table:[2]

```
library(gt)

msummary(models,
         title = 'Table customized using `gt` functions.',
         stars = TRUE,
         notes = c('First custom note to contain text.',
                   'Second custom note with different content.')) %>%
   # spanning labels
   tab_spanner(label = 'Miles / Gallon', columns = 2:3) %>%
   tab_spanner(label = 'Horsepower', columns = 4) %>%
   # color
   tab_style(style = cell_text(color = "blue", weight = "bold"),
             locations = cells_body(columns = 1)) %>%
```

[2]The following code chunk is set to `eval=!is_latex`. It will not be executed when the Rmarkdown document is compiled to a PDF file.

Table 2: Table customized using ‘kableExtra‘ functions.

	Miles / Gallon		Horsepower
	Model 1	Model 2	Model 3
(Intercept)	37.885***	28.725***	-215.768**
	(2.074)	(7.592)	(88.362)
cyl	-2.876***	-2.484***	39.012***
	(0.322)	(0.447)	(5.205)
drat		1.872	33.662*
		(1.494)	(17.385)
Num.Obs.	32	32	32
R2	0.726	0.740	0.728
Adj.R2	0.717	0.722	0.709
AIC	169.3	169.6	326.7
BIC	173.7	175.5	332.6
Log.Lik.	-81.653	-80.809	-159.348

* p < 0.1, ** p < 0.05, *** p < 0.01
First custom note to contain text.
Second custom note with different content.

```
   tab_style(style = cell_fill(color = "pink"),
             locations = cells_body(rows = 3))
```

Customizing tables with `kableExtra`

In `modelsummary`, the default output format for HTML tables is `gt`. If we want to use `kableExtra` functions to customize our HTML tables instead, we must declare our preference by setting the `modelsummary_html` option:

```
library(kableExtra)

options(modelsummary_html = 'kableExtra')
```

We can use functions from the `kableExtra` package to post-process and customize a `modelsummary` table:[3]:

```
msummary(models,
         title = 'Table customized using `kableExtra` functions.',
         stars = TRUE,
         notes = c('First custom note to contain text.',
                   'Second custom note with different content.')) %>%
    kable_styling %>%
    # spanning labels
    add_header_above(c(" " = 1, "Miles / Gallon" = 2, "Horsepower" = 1)) %>%
    # color
    row_spec(3, background = 'pink') %>%
    column_spec(1, color = 'blue')
```

[3]The following code chunk is set to `eval=!is_rtf`. It will not be executed when the Rmarkdown document is compiled to an RTF file.