3A – Mais Sobre R Markdown

Code

James Hunter, Ph.D.

06 June 2020

Eu gostaria de explicar mais sobre o uso de R Markdown. Ele pode ser uma ferramenta muito útil com qualquer tipo de trabalho analítico, não só o curso de Análise dos Dados. Como eu tinha dito antes, é minha ferramenta principal para estudos científicos. A razão disso é que ele fica muito fléxivel em termos de tipos de documentos que pode produzir e com a formatação desses documentos.

Este assunto, como a programação de R, também tem recursos mais profundos que são a fonte deste capítulo. O criador de R Markdown é Yihue Xie de RStudio. Os posts de blog dele e o livro **R Markdown: The Definitive Guide**[1] usei bastante na minha educação sobre R Markdown. Também, as materiais das aulas de Alison Hill, PhD[2], também de RStudio, guiaram a preparação deste capítulo.

Elementos de Um Documento R Markdown

O Que É “Markdown”

“Markdown” é um sistema de escrever documentos sem todo a complicação de um processador de palavras como Microsoft Word. Os arquivos Markdown são arquivos de texto simples. Formatação como **negrito** ou *itálico* são marcados simplesmente com caráteres de texto específicos que são traduzidos quando o documento é renderizado para formato final. Markdown foi criado como um sistema em 2004. A ideia foi de deixar pessoas “escrever [documentos] utilizando um formato de texto simples que seria fácil para escrever e ler”[3]. Vários softwares foram desenvolvidos para aproveitar do formato Markdown durante a década seguinte, mas ele nunca tornou muito popular. Markdown foi estendido logo depois o lançamento por um sistema chamado PANDOC, que permitiu que a formatação de Markdown seja aplicado a um grande número de tipos dos arquivos.

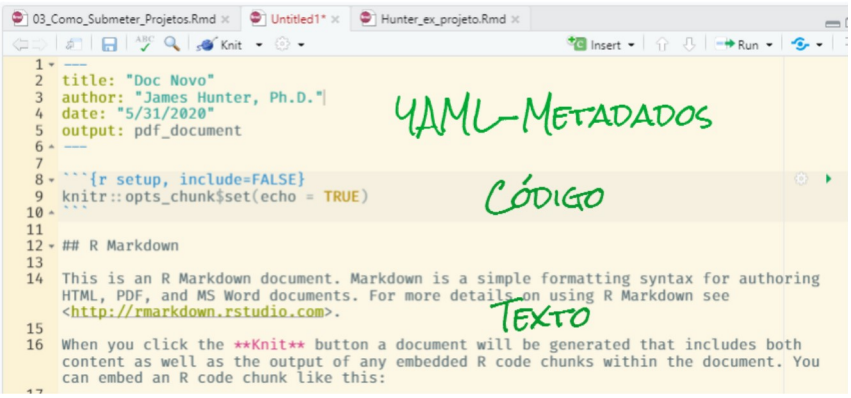
Foram as adaptações do formato Markdown para tipos de documentos específicos que fez Markdown crescer bastante. Entre eles a incorporação de Markdown em RStudio e a elaboração do formato do documento `.rmd`. Foi a combinação de Markdown e PANDOC que facilitou a criação de R Markdown que usamos aqui.

A implementação de Markdown em RStudio permite que você faz algumas coisas importantes:

1. Criar um **documento dinâmico**: quando os seus dados mudam, pode simplesmente *knit* o documento de novo com os novos documentos sem precisar re-escrever ele. Ao início do curso, falei de meu trabalho com os dados de COVID-19. Faço uma nova versão do relatório cada 2 dias. Não preciso mudar nada no documento, só clicar em *Knit* e voilà, nova versão. Reduz bastante o trabalho.
2. Incorporar todas as partes de análise no mesmo documento. Não precisa mais integrar por mão um documento de Word com uma planilha de Excel e um gráfico criado em um terceiro software. Todos os elementos ficam dentro de um documento só.

Elementos de R Markdown

Em Capítulo 3, descrevi rapidamente como formatar documentos em R Markdown. Agora, explicarei mais um pouco sobre como R Markdown funciona, o processo de impimir um documento RMD e formatar ele com YAML, um das três componentes de um arquivo `.rmd`. Um documento R Markdown tem três elementos diferentes: YAML, blocos de texto e blocos de código. O primeiro instrui RStudio e o pacote `R Markdown` o que eles devem fazer quando você *knit* o documento. Esse quer dizer quando você produz a versão impressa no formato *html*, *pdf* ou Word. O foco aqui seria na parte de YAML.



Partes de R Markdown

YAML - O Cabeçalho do Documento

No início do documento tem que pôr a parte escrito na linguagem *YAML*. YAML pode dizer "Yet Another Markup Language" ou como alguns pensam "YAML Ain´t Markup Language" porque não usa a mesma sintaxe que o resto do docukmento. O que tem no YAML é os metadados que servem como um endereço num envelope. Os campos que vai inserir aqui contam para RStudio onde mandar o documento (a HTML, PDF ou Word) e como.

Metadados são dados sobre o documento não dados vindo do documento. Este documento aqui fala de R Markdown. Mas, a parte de YAML

```
---
title: "3A – Mais Sobre R Markdown"
author: "James Hunter, Ph.D."
date: "`r format(Sys.Date(), "%d %B %Y")`"
output:
  html_document:
    toc: true
    theme: flatly
    df_print: paged
    code_download: true
---
```

YAML deste Capítulo

Esta parte do documento tem o seguinte formato. Ele começa com três hífens numa linha sozinhos seguidos por instruções no formato de `<key>: <value>`, assim:

```
---
key: value
---
```

Linha de YAML

Complexidade de YAML

A secção de YAML pode ser tão simples ou complicada como você quiser. A versão mais simples, mas útil é aquela do novo documento acima. Lá só tem campos para título, autor, data e qual tipo de output que você quer usar. Naquele caso, o output do processo do *knit* vai ao um documento *pdf* utilizando Latex.

O YAML para este capítulo é mais complicado. Lá, tem título e autor como antes. A data aqui usa algum código "inline" de R para formatar a data automaticamente com a data em que o documento seja *knit* (processado). Neste caso, vou criar um documento *html*, ou seja um documento que é uma página de web (`output: html_document`). Este output tem vários opções, que seguem a formula de *key* e *value* como mostrado na figura acima. Entre as opções, estou includindo um índice analítico (`toc`) e estou contando para `knitr` que eu quero impressões dos *data frames* (tibbles) com paginação. Finalmente, o produto final vai ter uma caixa ao início que permite que o leitor faz um download dos blocos de código no arquivo.

O resultado dessas opções é o seguinte.

Cabeçalho do Arquivo

Temas nos Documentos R Markdown

Existe mais uma linha no YAML: `theme: flatly`. Você pode ter bastante controle sobre as cores e as fontes que *knitr* vai usar para imprimir seu documento, especificando uma tema. Essas temas vêm de uma biblioteca no internet das temas chamada Bootswatch.[4] Bootswatch faz parte do sistema de desenvolvimento dos sites de web, Bootstrap. Os temas incluídos no R Markdown são:

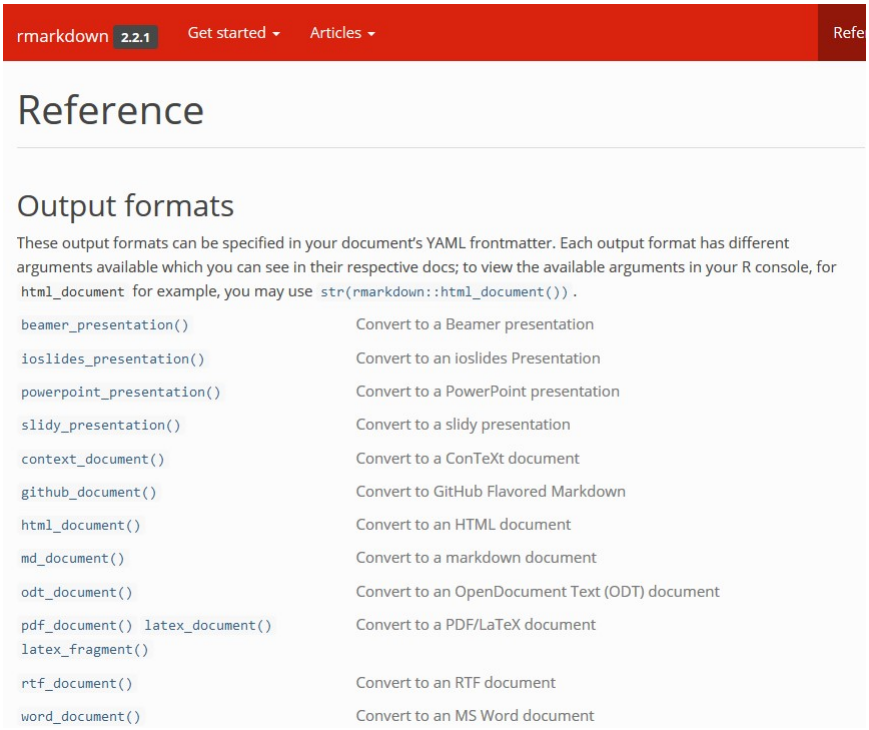
- default
- cerulean
- journal
- flatly
- darkly
- readable
- spacelab
- united
- cosmo
- lumen
- paper
- sandstone
- simplex
- yeti

Você pode achar exemplos das fontes e cores usadas nos temas no site de Bootswatch. Eles dão para o autor do arquivo mais controle sobre a construção dos documentos.

Tipos de Output

Este capítulo enfatiza documentos do texto, mas é importante lembrar que existem temas e tipos de output para slides, folhetos, panfletos, sites, blogs e até livros. Pode achar uma lista dos outputs no site de R Markdown:

https://rmarkdown.rstudio.com/docs/reference/index.html#section-output-formats (https://rmarkdown.rstudio.com/docs/reference/index.html#section-output-formats).



Tipos de Output

Formatação dos Resultados nos Blocos de Código

Em Capítulo 3, falei da formatação dos blocos de código em si. Aqui gostaria de expandir isso um pouco e falar sobre uma maneira de apresentar seus resultados em um formato flexível e fácil para o leitor ler.

Bloco de Código `setup`

No primeiro bloco de código, seguindo do cabeçalho YAML, é útil para especificar alguns padrões que você quer usar no documento inteiro. A figura seguinte mostra os padrões que estabeleci para este capítulo.

Bloco Setup

Este bloco faz duas coisas. Primeiro, ele usa a função `opts_chunk$set()` para estabelecer o que será normal neste documento, o que vai ocorrer em todos os blocos de código. Há 53 opções disponíveis[5]. Consulte o R Markdown Cheat Sheet para ver as opções mais importantes. Neste documento, usei:

- `echo = TRUE` : os blocos de código vão aparecer no documento final;
- `message = FALSE` : as mensagens de R seriam suprimidas;

- `warning = FALSE` : advertências de R sobre possíveis problemas com o resultado do código seriam suprimidas;
- `fig.width=5` : largura de todas as figuras seria 5 polegadas;
- `fig.height=4` : altura de todas as figuras seria 4 polegadas;
- `fig.align = "center"` : todas as figuras seriam centralizadas na página.

Esses comandos se localizam no pacote `knitr` porque este pacote faz o trablho da produção do documento final.

A segunda parte do bloco carrega os pacotes que uso nos blocos de documento.

O pacote `pacman` tem duas vantagens sobre o uso repetitivo de `library()` . Primeiro, pode pôr todos os pacotes que quer carregar em um vetor simples. Também a função `pacman::p_load()` pode lidar com pacotes que ainda você não instalou se eles vêm de CRAN ou de um repositório de GitHub. Se o pacote não existe na sua instalação de R, a função procura ele no CRAN, faz o download e instalação no R e carrega ele automaticamente. Se o pacote fica no GitHub, você tem que listar ele com o nome do repositório (e.g., `jameshunterbr/meu_pacote`).

Você não precisar ficar com esses especificações em todos os blocos de código. No cabeçalho de cada bloco, você pode revisar um padrão de bloco `setup` a vontado, simplesmente colocando o nome do argumento e um novo valor para ele. Por exemplo, se tem um bloco que você não quer mostrar o código, pode estruturar o bloco assim.

Echo de Código Suprimida

Apesar que a opçaõ global é de incluir o código no documento, neste bloco não vai ser incluído porque por este bloco `echo = FALSE` .

Produção do Documento

Quando o seu documento está pronto para ser publicado ou testado[6], pode clicar no palavra na barra acima do documento *Knit*. RStudio vai proceder a interpretar seu Markdown e aplicar o programa de PANDOC para criar um documento no formato que você especificou no campo de `output:` no YAML. Vai abrir uma aba ao lado de Console chamado “R Markdown” onde vai mostrar o progresso do processo.

Se o *knitting* do documento para por causa de um erro, este painel vai mostrar onde o erro ocorreu (qual linha do documento) e o que causou o problema. Se encontra um erro, concerte e tentar de novo o *knit*. Se o processo continua até ele mostra que o output foi criado, você terminou. RStudio vai produzir uma cópia do produto final para você pode ver o que o software pensou que você quis fazer. Se você quer modificar a aparência, pode fazer uma correção e fazer o *knit* de novo. Normalmente, você não consegue tudo que quer fazer em termos de formatação na primeira tentativa. Tenha um pouco de paciência e refazer o cíclo de *knit* - corrigir até o html, pdf ou Word é satisfatório.

Conversão de *html* a *pdf*

Os temas são disponíveis para documentos com um `output` de *html*. Se você quer usar *html* como formato para escrever seu documento, mas realmente quis um *output* de *pdf*, pode converter o output a pdf facilmente. Só precisa seguir os passos seguintes.

1. RStudio vai abrir uma cópia de seu *html* o na aba de *Viewer* ou numa janela separada. Nos dois casos, vai ter um botão que permite que você abrir o arquivo para o seu browser.

Botão de Browser

2. Com o arquivo carregado como página no browser, pode usar o comando de imprimir para imprimir o documento para um arquivo. Este arquivo vai ter o formato *pdf*. Não posso falar muito exatamente como esse vai acontecer porque cada browser tem seu próprio jeito de imprimir páginas.

---

1. Xie, Y, Allaire, JJ, Grolemund, G, **R Markdown: The Definitive Guide**, 2020, https://bookdown.org/yihui/rmarkdown/ (https://bookdown.org/yihui/rmarkdown/).↩
2. **An Introduction to R Markdown**, https://rstd.io/rmd4cdc (https://rstd.io/rmd4cdc).↩
3. Wikipedia contributors. Markdown. Wikipedia, The Free Encyclopedia. June 3, 2020, 02:02 UTC. Available at: https://en.wikipedia.org/w/index.php?title=Markdown&oldid=960453793 (https://en.wikipedia.org/w/index.php?title=Markdown&oldid=960453793). Accessed June 5, 2020.↩
4. https://bootswatch.com/3/ (https://bootswatch.com/3/)↩
5. Pode executar o código seguinte para ver essas opções e os valores *default* delas: `str(knitr::opts_chunk$get())` .↩
6. Recomendo que você testa a aparência de seu documente frequentemente para acertar que está com a formatação que você espera e que seus resultados de programação e análise são claros.↩