Communicating Your Analytics Through RMarkdown

Introduction to Outbreak Analytics using R (2021)

Yang Liu, PhD

23 November, 2021

1 What is RMarkdown?

2 Why?

3 How does it work?

4 Piecing Things Together

5 Additional Resources

What is RMarkdown?

What is RMarkdown?

image credit: mohamed_hassan (pixabay.com)

What is RMarkdown?

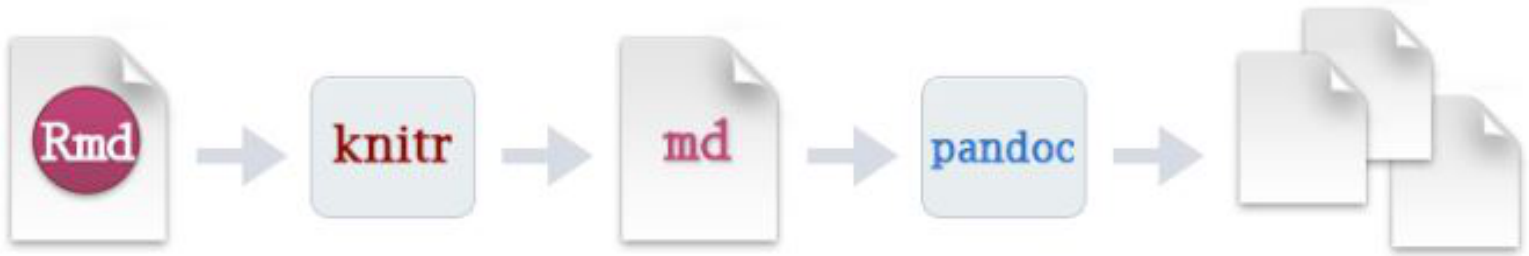


image credit: http://rstudio.com

The biggest difference between RMarkdown and Microsoft Word

The things you would normally do by **clicking around**, you now do through **codes**.

Why?

Why?

1. It is dynamic.

What you **see**:

If the radius of a circle is 2, then the area of the circle is 12.57.

What you **type**:

````
```{r circle}
radius = 2
```
````

If the radius of a circle is `` `r radius` ``, then the area of the circle is `` `r round(pi*radius*radius, 2)` ``.

Why?

1. It is dyanmic.
2. It is flexible:

Code Accepted:

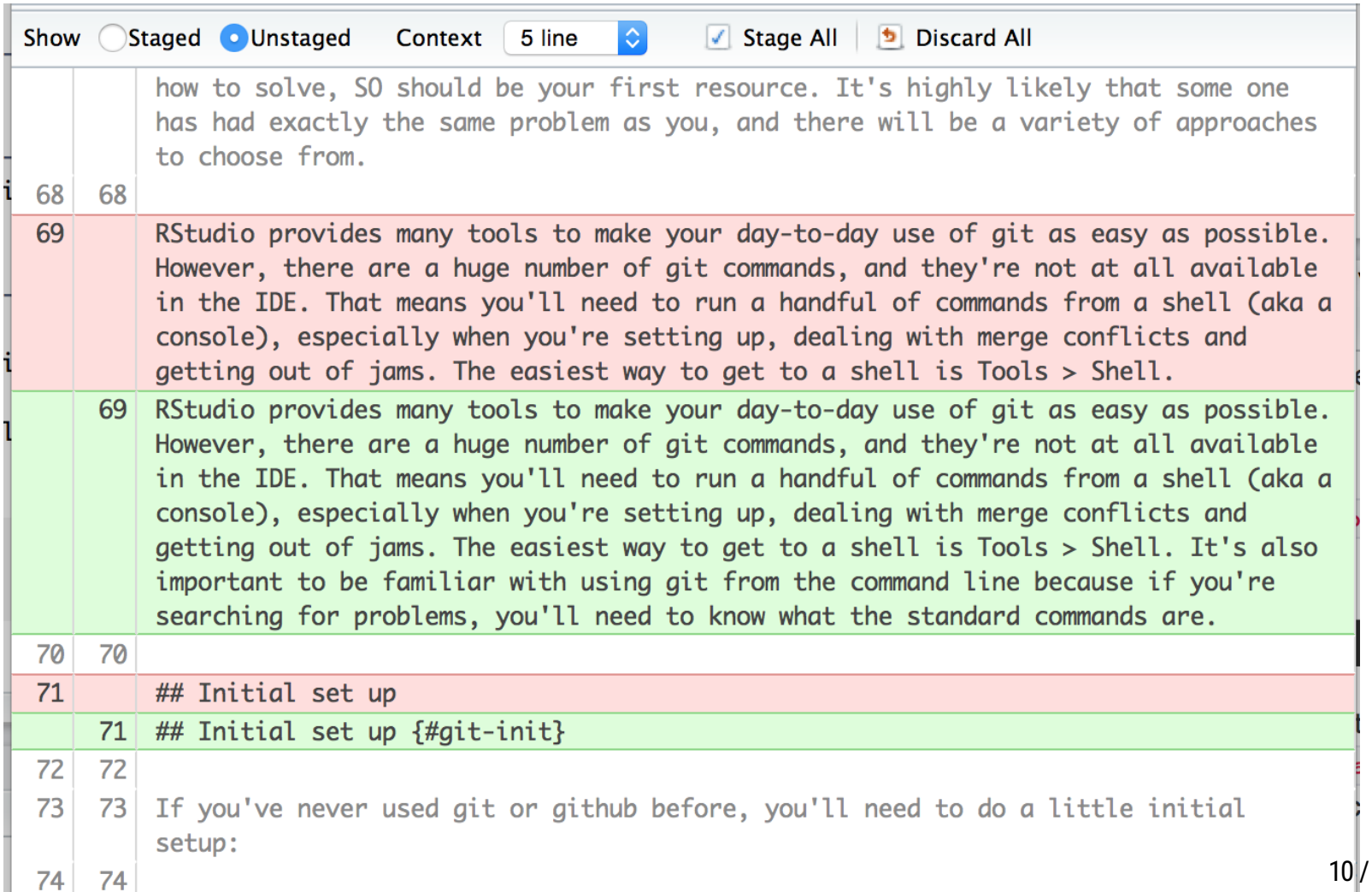
- R
- Python
- Bash
- Rcpp
- Stan
- JavaScript
- CSS

Output Types:

- PDF
- html
- Word
- Powerpoint
- Many more...

Why?

1. It is dynamic
2. It is flexible.
3. It plays well with GitHub:



Why?

1. It is dyanmic.
2. It is flexible.
3. It plays well GitHub.
4. It is all-in-one interface to manage your document:

- Images
- Tables
- References and bibliography

Why?

1. It is dyanmic.
2. It is flexible.
3. It plays well GitHub.
4. It is all-in-one interface to manage your document.
5. It is useful:

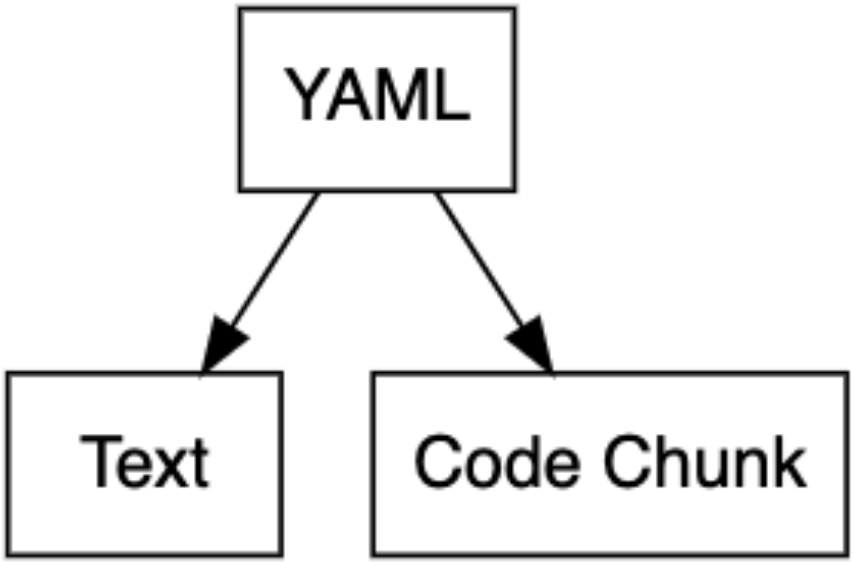
- Preparing Manuscripts/ Presentations;
- Generating Reports;
- Sharing Results;
- Developing Websites;
- Many more…

How does it work?

Getting Started

```
install.packages("tinytex")
install.packages("rmarkdown")
library(rmarkdown)
tinytex::install_tinytex()
```

Overall Structure



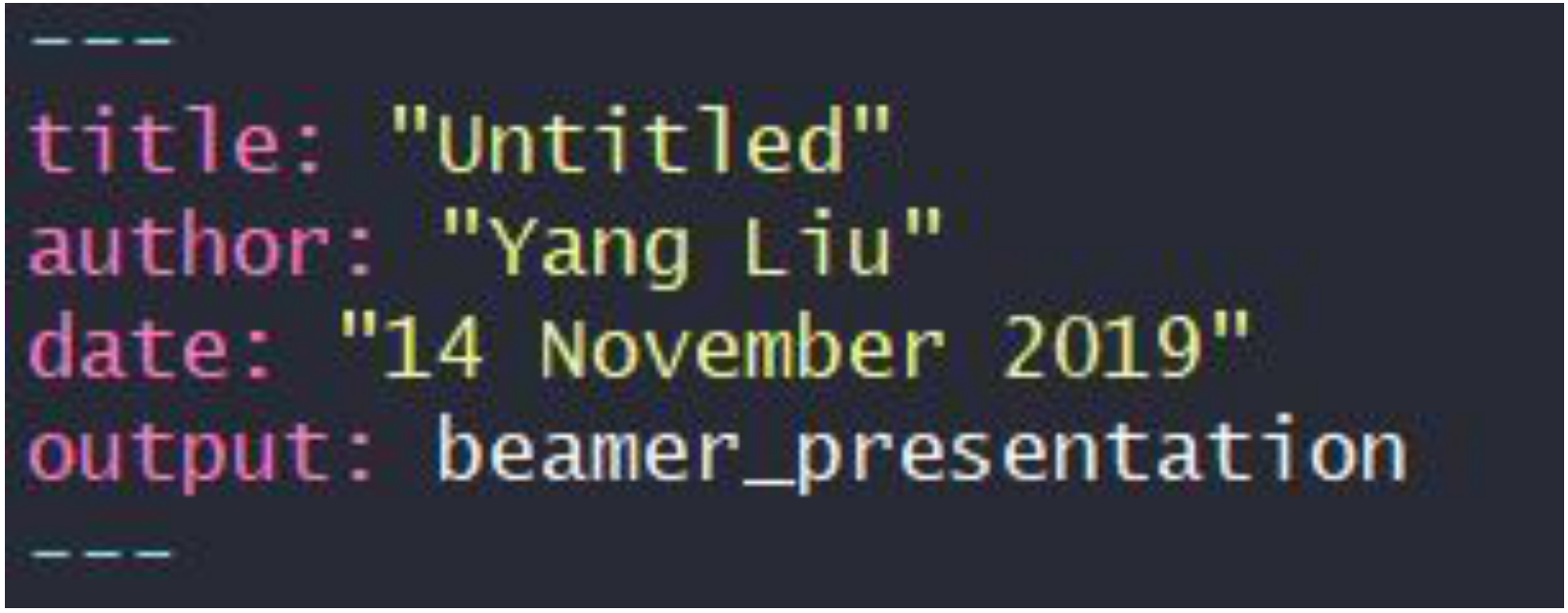
YAML

YAML

The **YAML** (YAML Ain't Markup Language) section is often on the very top of a RMarkdown document. It is essentially the **metadata**, describing the overall environment the document is working in.

Now in your RStudio, go to
`File -> New File -> R Markdown... -> Document -> PDF -> OK`.

You just made a YAML section!



`output:` is used to specify the output file type. - PDF = `pdf_document`
- html = `html_document` - Word = `word_document` - Powerpoint = `powerpoint_presentation` - PDF Presentation = `beamer_presentation`

What else can go here?

- **Page setup:** font type/ sizes, margin sizes, theme/ colour scheme…;
- **Other formmating setups:** image sizes/ locations…;
- **Document components and their settings:** table of contents, references/ bibliographies…

Now that you've got your first `.Rmd`

Compared to an R Script Window, a markdown document source window:

- **Does not** have the "Source on Save" tick box or the "Source" button;
- **Does not** have the "Code Tools" button;
- **Does not** have the "Compile Report" button;
- Now have the option to insert code chunks and run code chunks separately;
- Now have the option to move around sections;
- Now have the option to publish to the web.

Text

Example: Working with fonts

What you **see**:

April is the *cruellest month*,
breeding ~~lilacs out of the dead land~~, mixing `memory` and `desire`,
stirring dull $\sqrt{roots}$ with spring rain.

What you **type**:

```
**April** is the _cruellest month_,
~~breeding lilacs out of the dead land,~~
mixing `memory` and `desire`,
stirring dull `$\sqrt{roots}$` with
\textcolor{blue}{spring rain}.
```

Example: Working with lists

What you **see**:

April is the cruellest month,

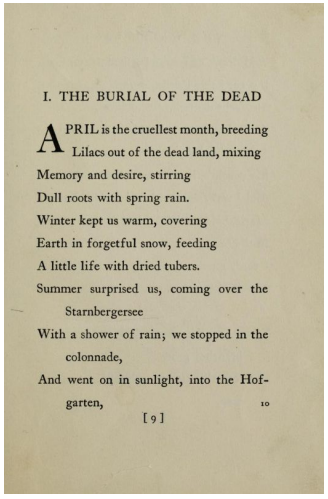
- breeding lilacs out of the dead land,
- mixing memory and desire,
    - stirring dull roots with bluespring rain.

What you **type**:

```
April is the cruellest month,
+ breeding lilacs out of the dead land,
+ mixing memory and desire,
    - stirring dull roots with blue spring rain.
```

Text: Working with images

What you **see**:



I. THE BURIAL OF THE DEAD

APRIL is the cruellest month, breeding
Lilacs out of the dead land, mixing
Memory and desire, stirring
Dull roots with spring rain.
Winter kept us warm, covering
Earth in forgetful snow, feeding
A little life with dried tubers.
Summer surprised us, coming over the
Starnbergersee
With a shower of rain; we stopped in the
colonnade,
And went on in sunlight, into the Hof-
garten, 10

[9]

image credit: https://i.pinimg.com/originals/f9/22/c6/f922c68817b1b1b4f01b929524d2afca.jpg

What you **type**:

```
![](ifoundthisonline.jpg)
```

Text: Working with equations

What you **see**:

In RMarkdown, it's easy to work with equations both inline (e.g., sample mean, defined as $\bar{x} = \frac{\sum x}{n}$).

What you **type**:

```
In RMarkdown, it's easy to work with equations
both inline (e.g., sample mean, defined as
$\bar{x} = \frac{\sum_{}^{} {x}}{n}$).
```

Text: Working with equations

What you **see**:
Or in display mode:

$$s = \sqrt{\frac{\sum(x - \bar{x})^2}{n - 1}}$$

What you **type**:

```
Or in display mode: $$s =\sqrt{ \frac{
\sum_{}^{} (x-\bar{x})^2}{n-1}}$$
```

Code Chunks

Code Chunk

Additionally, blocks of codes can actually be included as a part of your RMarkdown file (`.rmd`). In between your writing, a code chunk may look like the following:

````
```{r}
plot(things)
```
````

After which, you can resume writing.

Code Chunks

But when would we need this?

- Print
- Data
- Tables (e.g., model output)
- Plots
- Demonstrate Methods

What if we want to print stuff?

- We would like our codes to run in silence.
- We would like the results of the codes to be included in the document.

And how do we do that?

Options we'll need here are:

- `echo`: when set to T, display code in output document.
- `eval`: when set to T, run code in chunk.

Example: Print Mode Output

What you **see**:

term	estimate	std.error	statistic	p.value
(Intercept)	53.9555609	0.3149949	171.29025	0
gdpPercap	0.0007649	0.0000258	29.65766	0

What you **type**:

````
```{r echo = F, eval = T}
data <- gapminder::gapminder
kable(broom::tidy(lm(data = data,
lifeExp~gdpPercap)))
```
````

Example: Print scatter plot

What you **see**:

What you **type**:

```
```{r echo = F, eval = T, fig.height = 1}
ggplot(data, aes(x = gdpPercap,
y = lifeExp)) +
geom_point() +
labs(x = "GDPpc",
y = "lifeExp")
```
```

What if we want to demonstrate stuff?

- We would like our codes to appear in text without being ran.
- We don't really care about the results in this case.

Example: Demonstrate through codes

What you **see**:

```
ggplot(data, aes(x = gdpPercap,
                 y = lifeExp)) +
  geom_point() +
  labs(x = "GDPpc",
       y = "lifeExp")
```

What you **type**:

````
```{r echo = T, eval = F}
ggplot(data, aes(x = gdpPercap,
y = lifeExp)) +
geom_point() +
labs(x = "GDPpc",
y = "lifeExp")
```
````

RMarkdown Engine

We mentioned that RMarkdown can handle more than just R. We can specify the type of language of a specific code chunk is ran in:

Conditioned on You already have a version of Python installed locally, what you **see**:

```
## 2
```

What you **type**:

````
```{python echo = F}
1+1
```
````

Piecing Things Together

Generating output

1. Press the `Knit` button in the Source window.
2. `render` function, i.e., `rmarkdown::render("XXXX.Rmd")`

Additional Resources

Additional Resources

- R Markdown Quick Tour
- Introduction to RMarkdown
- R Markdown: The Definitive Guide
- R Markdown:: CHEAT SHEET

Let's now move on to the practical file! You should not need more resources than the above to complete the practical.

1. `RMarkdown_PracticalInstructions.pdf`
2. `RMarkdown_Practical.Rmd`