# Scientific Communication: TEX, RMarkdown
# Winter Institute in Data Science

Ryan T. Moore

2023-01-04

Scientific Communication

LaTeX

RMarkdown

# Scientific Communication

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly
- is replicable, portable

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly
- is replicable, portable
- automates frequently-used or tedious operations

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly
- is replicable, portable
- automates frequently-used or tedious operations
- (usually) separates content and formatting

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly
- is replicable, portable
- automates frequently-used or tedious operations
- (usually) separates content and formatting
- allows the author control . . .

# Scientific Communication

Good scientific communication

- conveys ideas, procedures, and results clearly
- is replicable, portable
- automates frequently-used or tedious operations
- (usually) separates content and formatting
- allows the author control . . .
- but supplies legible defaults and structure that allow the reader (and author) to focus on content

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)
- LATEX (Lamport): 1983 (easier TEX: `\section{}`, . . . )

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)
- LATEX (Lamport): 1983 (easier TEX: `\section{}`, ...)
- `Sweave`: $\approx$ 2000 (integrate R and TEX)

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)
- LATEX (Lamport): 1983 (easier TEX: `\section{}`, . . . )
- `Sweave`: ≈ 2000 (integrate R and TEX)
- Markdown (Gruber): 2004 (lightweight, legible)

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)
- LATEX (Lamport): 1983 (easier TEX: `\section{}`, . . . )
- `Sweave`: ≈ 2000 (integrate R and TEX)
- Markdown (Gruber): 2004 (lightweight, legible)
- `knitr` (Xie): 2012 (integrate R into anything)

# Scientific Communication

A brief history of languages

- `Plain` TEX (Knuth): 1978 (docs for any machine)
- LATEX (Lamport): 1983 (easier TEX: `\section{}`, . . . )
- `Sweave`: ≈ 2000 (integrate R and TEX)
- Markdown (Gruber): 2004 (lightweight, legible)
- `knitr` (Xie): 2012 (integrate R into anything)
- RMarkdown: (a `knitr` extension)

LaTeX

From the LaTeX Project (https://www.latex-project.org):

> *LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents. LaTeX is available as free software.*

# Convey ideas, procedures, results clearly

Consider

```
(E_i ^100 x_i) / (x^2 + theta sqrt(z))
```

# Convey ideas, procedures, results clearly

Consider

```
(E_i ^100 x_i) / (x^2 + theta sqrt(z))
```

This is difficult to read.

# Convey ideas, procedures, results clearly

Consider

```
(E_i ^100 x_i) / (x^2 + theta sqrt(z))
```

This is difficult to read.

On the other hand,

$$\frac{\sum_{i=1}^{100} x_i}{x^2 + \theta\sqrt{z}}$$

# Convey ideas, procedures, results clearly

Consider

```
(E_i ^100 x_i) / (x^2 + theta sqrt(z))
```

This is difficult to read.

On the other hand,

$$\frac{\sum_{i=1}^{100} x_i}{x^2 + \theta\sqrt{z}}$$

- Historically: difficult, bad spacing, etc. on WYSIWYGs

# Convey ideas, procedures, results clearly

Consider

```
(E_i ^100 x_i) / (x^2 + theta sqrt(z))
```

This is difficult to read.

On the other hand,

$$\frac{\sum_{i=1}^{100} x_i}{x^2 + \theta\sqrt{z}}$$

- Historically: difficult, bad spacing, etc. on WYSIWYGs
- Now, macOS's Pages *requests* your LaTeX

# Replicable, portable

LaTeX is plain text. You can open it in any text editor.

# Replicable, portable

LaTeX is plain text. You can open it in any text editor.

You could open/compile the first LaTeX document.

# Replicable, portable

LaTeX is plain text. You can open it in any text editor.

You could open/compile the first LaTeX document.

`.doc` is actually 4 different file formats!

# Automates frequently-used, tedious operations

Don't make a Table of Contents.

## Automates frequently-used, tedious operations

Don't make a Table of Contents.

Just `\tableofcontents`.

## Automates frequently-used, tedious operations

Don't make a Table of Contents.

Just `\tableofcontents`.

In LaTeX documents, the \ prefaces a command.

## Automates frequently-used, tedious operations

Don't make a Table of Contents.

Just `\tableofcontents`.

In LaTeX documents, the `\` prefaces a command.

In LaTeX, the `{ }` enclose arguments:

```
\command{arg1}
```

## Automates frequently-used, tedious operations

Don't make a Table of Contents.

Just `\tableofcontents`.

In LaTeX documents, the `\` prefaces a command.

In LaTeX, the `{ }` enclose arguments:

```
\command{arg1}
```

In LaTeX, the `[ ]` enclose (opt) further params:

```
\command[param1,param2]{arg1}
```

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

1. Create a `.bib` file of sources

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

1. Create a `.bib` file of sources
2. Each source has a citation key (E.g., `moore15`)

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

1. Create a `.bib` file of sources
2. Each source has a citation key (E.g., `moore15`)
3. Refer to the key to cite

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

1. Create a `.bib` file of sources
2. Each source has a citation key (E.g., `moore15`)
3. Refer to the key to cite
4. Include `.bib` file name, so LaTeX/`knitr` finds refs

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

1. Create a `.bib` file of sources
2. Each source has a citation key (E.g., `moore15`)
3. Refer to the key to cite
4. Include `.bib` file name, so LaTeX/`knitr` finds refs
5. (Include `.bst` to adjust reference style)

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

Write "One should read `\cite{moore2015}`.", then add

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

Write "One should read `\cite{moore2015}`.", then add

```
\bibliographystyle{apsa-leeper}
\bibliography{my_bib}
```

# Automates frequently-used, tedious operations

Don't compile a bibliography, or reformat it.

Write "One should read `\cite{moore2015}`.", then add

```
\bibliographystyle{apsa-leeper}
\bibliography{my_bib}
```

Which will render as "One should read Moore (2015)".

and

Moore, Ryan T. "Overcoming Barriers to Heterogeneous-Group Learning in the Political Science Classroom". *PS: Political Science & Politics*, 48(1):149–156, 2015.

To build documents into `.pdf`, need an installation of TeX.

You need a TeX engine for processing.

To build documents into `.pdf`, need an installation of TeX.

You need a TeX engine for processing.

Or `tinytex`.

Or Overleaf.

To build documents into `.pdf`, need an installation of TeX.

You need a TeX engine for processing.

Or `tinytex`.

Or Overleaf.

`R` → `RStudio` analogous to `TeX build` → `TeX editor`

## tinytex

If you don't have a TEX build, try **tinytex**.

(See http://www.ryantmoore.org/files/ht/httinytex.pdf)

# tinytex

If you don't have a TEX build, try `tinytex`.

(See http://www.ryantmoore.org/files/ht/httinytex.pdf)

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

# tinytex

If you don't have a TeX build, try `tinytex`.

(See http://www.ryantmoore.org/files/ht/httinytex.pdf)

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

Or, just create `.html` files instead, and print to `.pdf`.

# Resources

- "The Not So Short Introduction to LaTeX $2_\varepsilon$" (Oetiker, et al.)
- "The Comprehensive LaTeX Symbol List" (Pakin $\approx$ 15,000!)
- Detexify http://detexify.kirelabs.org/classify.html
- Overleaf (https://www.overleaf.com)

# RMarkdown

## First, a quick example!

- Create `.Rmd` file
  - RStudio $\rightsquigarrow$ File $\rightsquigarrow$ New file $\rightsquigarrow$ R Markdown
- Add name, title to preamble
- Compile
- Edit, compile

# RMarkdown

RMarkdown is

- light
- legible
- literate
- LaTeX

# RMarkdown: light

1. no huge build over R

# RMarkdown: light

1. no huge build over R
2. less “ink”

# RMarkdown: light

1. no huge build over R
2. less “ink”

```
1. Item 1
2. Item 2
```

# RMarkdown: light

1. no huge build over R
2. less “ink”

```
1. Item 1
2. Item 2
```

vs.

```
\begin{enumerate}
\item Item 1
\item Item 2
\end{enumerate}
```

# RMarkdown: legible

```
I am **serious**
```

vs.

```
I am \bf{serious}
```

or

```
I am {\bf serious}
```

# RMarkdown: legible

```
I am **serious**
```

vs.

```
I am \bf{serious}
```

or

```
I am {\bf serious}
```

(Better for tables, figures, etc. Harder to see/adjust aspects of presentation)

# RMarkdown: literate

- Code in these notes is run, output is recreated

# RMarkdown: literate

- Code in these notes is run, output is recreated
- Programming (data analysis) is **in** the text document

# RMarkdown: literate

- Code in these notes is run, output is recreated
- Programming (data analysis) is **in** the text document

# RMarkdown: literate

- Code in these notes is run, output is recreated
- Programming (data analysis) is **in** the text document

Look, three plus four is 7.

# RMarkdown: literate

- Code in these notes is run, output is recreated
- Programming (data analysis) is **in** the text document

Look, three plus four is 7.

What I typed above:

```
Look, three plus four is `r 3 + 4`.
```

# RMarkdown: literate

This literacy is great for papers: analysis is *in* your paper. No copy-paste, update figure, etc.

# RMarkdown: literate

This literacy is great for papers: analysis is *in* your paper. No copy-paste, update figure, etc.

Often want to *see* the code and the (somewhat more) raw results.

# RMarkdown: literate

This literacy is great for papers: analysis is *in* your paper. No copy-paste, update figure, etc.

Often want to *see* the code and the (somewhat more) raw results.

That makes this literacy perfect for reports ("notebooks").

# RMarkdown: LaTeX

In a paper, format the math *and* do the calculation:

Look: $3 + 4 = 7$.

# RMarkdown: LaTeX

In a paper, format the math *and* do the calculation:

Look: $3 + 4 = 7$.

Above I typed:

```
Look: $3 + 4 = `r 3 + 4`$.
```

# RMarkdown: LaTeX

In a paper, format the math *and* do the calculation:

Look: $3 + 4 = 7$.

Above I typed:

```
Look: $3 + 4 = `r 3 + 4`$.
```

(Warning: formatting the line above was a challenge.)

Valid LaTeX will almost always knit well.

# RMarkdown

For more formatting and chunk options, download the full

http://www.ryantmoore.org/files/class/introPolResearch/PS_Rmd_template_full.Rmd

or the simpler

http://www.ryantmoore.org/files/class/introPolResearch/PS_Rmd_template_simple.Rmd

# Rmd vs. the Console

Knit `.Rmd` file:

- R runs the code from top to bottom of `.Rmd` file

# Rmd vs. the Console

Knit `.Rmd` file:

- R runs the code from top to bottom of `.Rmd` file
- Does not run other code; does **not** look in Console's workspace

# Rmd vs. the Console

Knit `.Rmd` file:

- R runs the code from top to bottom of `.Rmd` file
- Does not run other code; does **not** look in Console's workspace

# Rmd vs. the Console

Knit `.Rmd` file:

- R runs the code from top to bottom of `.Rmd` file
- Does not run other code; does **not** look in Console's workspace

- `.Rmd` file needs to be entirely self-contained

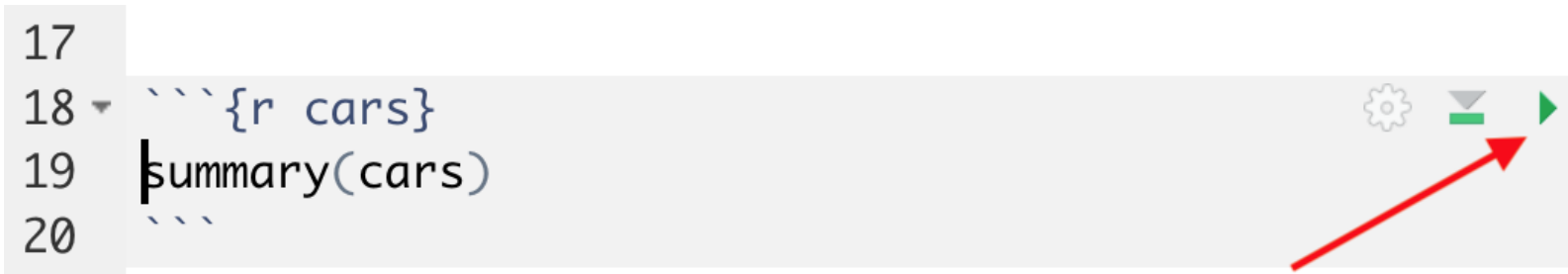
# Rmd vs. the Console

Knit `.Rmd` file:

- R runs the code from top to bottom of `.Rmd` file
- Does not run other code; does **not** look in Console's workspace

- `.Rmd` file needs to be entirely self-contained
- Set working directory, read data, create intermediate objects, etc. *within* the `.Rmd` file

# Rmd vs. the Console: Beware Green Arrow



# Rmd vs. the Console: Beware Green Arrow

- If you click, R looks in current workspace for objects, prints results to console or plotter, and reprints them in the confines of your `.Rmd`

# Rmd vs. the Console: Beware Green Arrow

- If you click, R looks in current workspace for objects, prints results to console or plotter, and reprints them in the confines of your `.Rmd`
- Behavior is "notebook-like". However, objects are not really *in* your `.Rmd` file, can be created out of order, are not part of the compiled/knit output

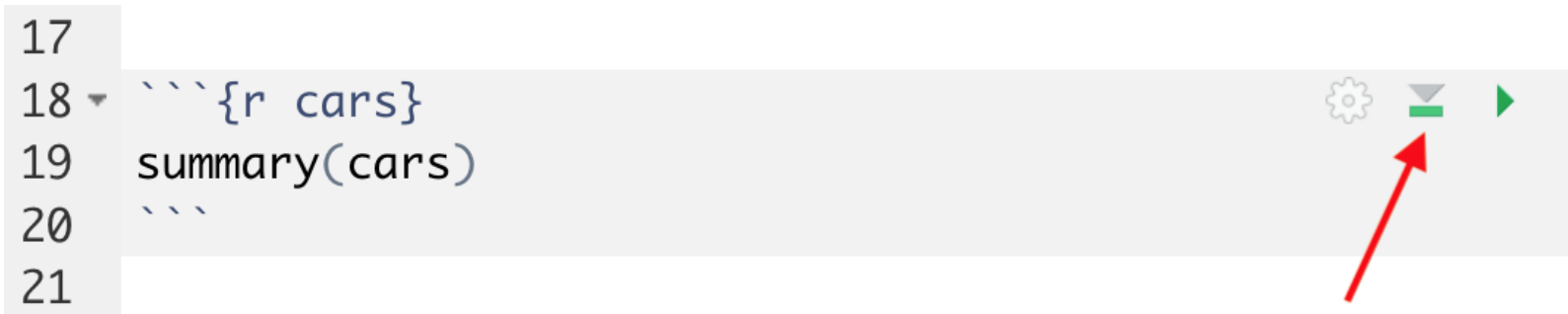
# Rmd vs. the Console: Beware Green Arrow

- If you click, R looks in current workspace for objects, prints results to console or plotter, and reprints them in the confines of your `.Rmd`
- Behavior is "notebook-like". However, objects are not really *in* your `.Rmd` file, can be created out of order, are not part of the compiled/knit output
- When you compile/knit to create an output file, only code that is run is code in your `.Rmd` file, from top to bottom

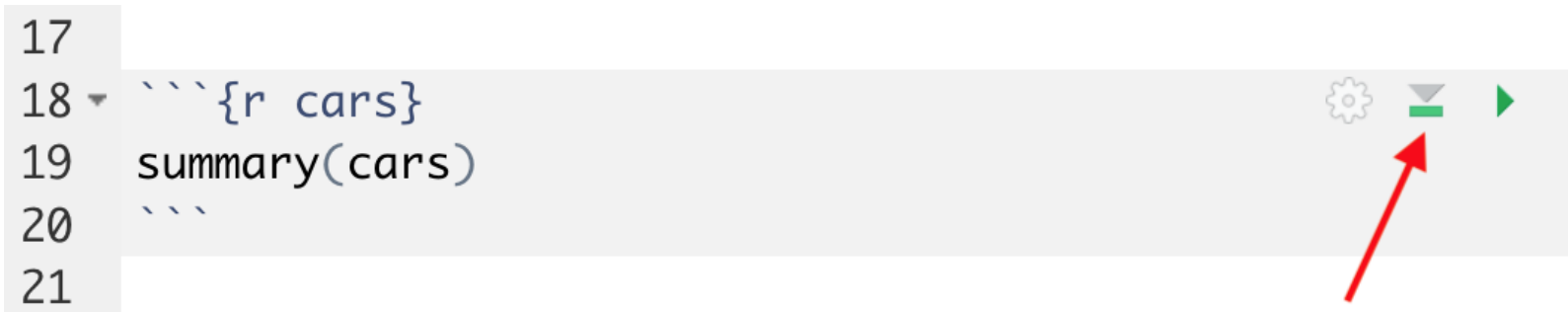
# Rmd vs. the Console: Beware Green Arrow

- If you click, R looks in current workspace for objects, prints results to console or plotter, and reprints them in the confines of your `.Rmd`
- Behavior is "notebook-like". However, objects are not really *in* your `.Rmd` file, can be created out of order, are not part of the compiled/knit output
- When you compile/knit to create an output file, only code that is run is code in your `.Rmd` file, from top to bottom
- This notebook-like feature is an aspect of RStudio that is not *inherent* in `.Rmd` (which could be compiled from a command line outside of RStudio, e.g.). To avoid confusion about the state, knit `.Rmd` file and look at output, rather than using green arrow.

# Rmd + the Console: Love the Green Bar



# Rmd + the Console: Love the Green Bar




"Run all chunks above" at Console.

(I know, you're wondering . . . )

(I know, you're wondering . . . )

How do I format LaTeX in LaTeX?

(I know, you're wondering . . . )

How do I format LaTeX in LaTeX?

`\LaTeX`

(I know, you're wondering . . . )

How do I format LaTeX in LaTeX?

`\LaTeX`

How to format "LaTeX is great."?

(I know, you're wondering . . . )

How do I format LaTeX in LaTeX?

`\LaTeX`

How to format "LaTeX is great."?

With a space in LaTeX: `\LaTeX~ is great.`

With a space in RMarkdown: `\LaTeX\ is great.`

# The Core Transformation Functions Quiz

Load the `gss_cat` data in the `forcats` package:

```
library(forcats)
data("gss_cat")
```

Create a `.Rmd` file, write code[1] in chunks to

1. Sort `gss_cat` by the values of `tvhours` (largest first). Store this as `gss_cat`, and show the first 10 rows.
2. Create a new var `birthyear` – each resp's `year` minus `age` – and attach it as column of `gss_cat`. Show `summary(gss_cat$birthyear)` and dimensions of `gss_cat`.
3. Create df `gss_cat_tv`, which has only the rows of `gss_cat` where `tvhours > 3`?
4. Calculate the mean value of `tvhours` in `gss_cat` within categories of `relig`. Sort this summary.

[1] `filter()`, `arrange()`, `group_by()`, `ungroup()`, `select()`, `rename()`, `mutate()`, `transmute()`, `summarise()`