RMARKDOWN - MANUSCRIPT WORKFLOW REVISITED

Using RStudio to Write Theses, Paper, and Presentations

Erik Kusch

erik.kusch@au.dk

Section for Ecoinformatics & Biodiversity
Center for Biodiversity and Dynamics in a Changing World (BIOCHANGE)
Aarhus University

10/09/2020

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

What is `Rmarkdown`?

What it is (*`Rmarkdown`/`.Rmd`*)

In its **essence**, `Rmarkdown` is a **coding document containing text** that can be used to generate:

- Manuscripts
- Presentations
- Posters
- Books
- ...

whilst **maintaining reproducibility** of research.

It is *heavily reliant* on the LaTeX machinery.

What you need (*Dependencies*)

To be able to use the bare minimum of `Rmarkdown`, you need an installation of `R` and `RStudio`.

Advanced use of `Rmarkdown` is usually fuelled through LaTeX and so you'll need installations of:

- LaTeX (for generation of pdf documents)
- JabRef (for referencing)
- Perl (for glossaries)

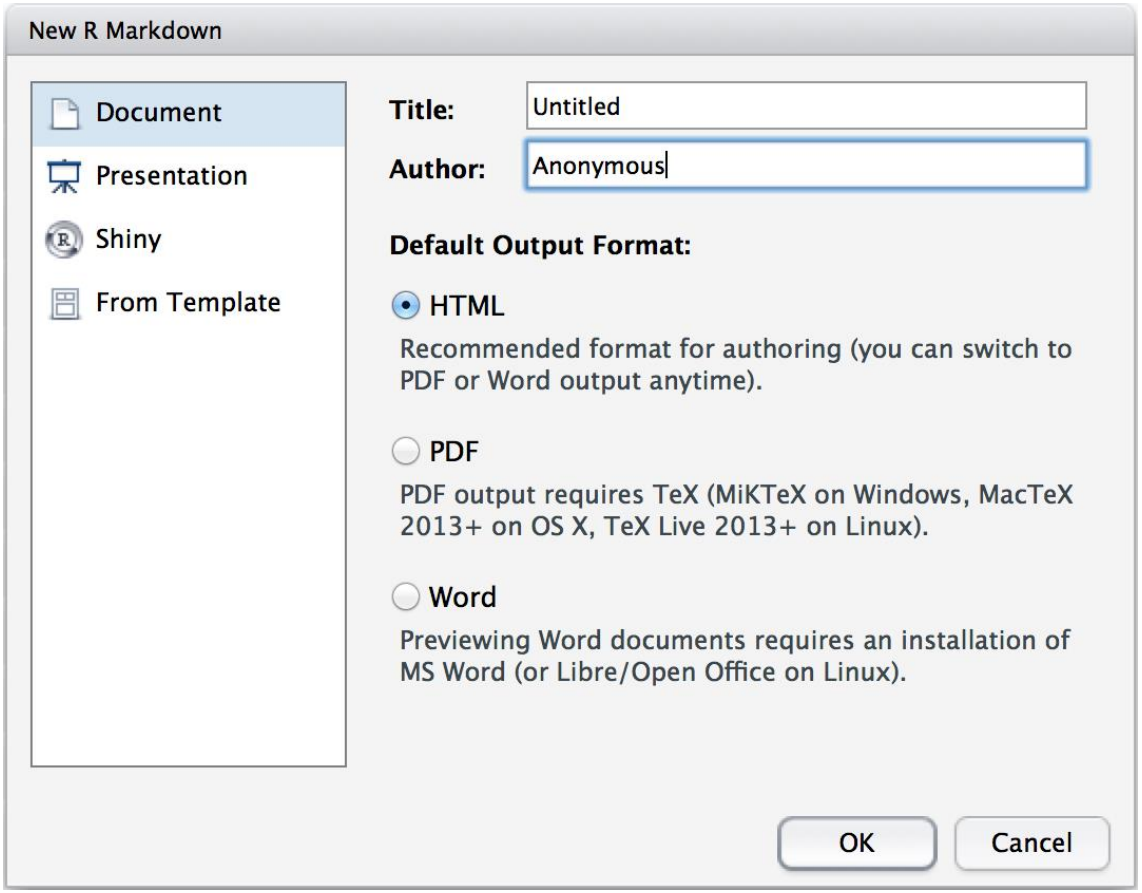
Why use `Rmarkdown`?

Using `Rmarkdown` for your research comes with a multitude of advantages:

1. Entire **workflow in one program** (`RStudio`)
2. **Research** and reports **reproducible** at the click of **one button**
3. **Combines** `R` functionality and LaTeX formatting (if desired)
4. **Consistent formatting**
5. **Clear presentation of code**
6. **Dynamic documents** (you can generate various output document types)
7. Applicable for **almost all document types** you may desire as an output (e.g. manuscripts, presentations, posters, etc.)

`Rmarkdown` Worflow

In the file menu of `RStudio`, click: **File** → **New File** → **R Markdown**



Write and edit the document and *compile* by **knitting**.

`Rmarkdown` Resources

I cannot cover every aspect of `Rmarkdown` coding in one go. In case you bite, here are resources to help you along:

- **Guides**
  - An Introduction: https://rmarkdown.rstudio.com/lesson-1.html
  - The *'Definitive Guide'*: https://bookdown.org/yihui/rmarkdown/
- **CheatSheets**
  - Reference Guide: https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
  - Proper CheatSheet: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

→ I recommend learning `Rmarkdown` by inspecting working scripts (including the one used to produce this presentation). You can find some on my github: https://github.com/ErikKusch.

`Rmarkdown` Resources

I cannot cover every aspect of `Rmarkdown` coding in one go. In case you bite, here are resources to help you along:

- **Guides**
  - An Introduction: https://rmarkdown.rstudio.com/lesson-1.html
  - The *'Definitive Guide'*: https://bookdown.org/yihui/rmarkdown/
- **CheatSheets**
  - Reference Guide: https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
  - Proper CheatSheet: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

→ I recommend learning `Rmarkdown` by inspecting working scripts (including the one used to produce this presentation). You can find some on my github: https://github.com/ErikKusch.

Rmarkdown Resources

I cannot cover every aspect of Rmarkdown coding in one go. In case you bite, here are resources to help you along:

- **Guides**
  - An Introduction: https://rmarkdown.rstudio.com/lesson-1.html
  - The *'Definitive Guide'*: https://bookdown.org/yihui/rmarkdown/
- **CheatSheets**
  - Reference Guide: https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
  - Proper CheatSheet: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

→ I recommend learning Rmarkdown by inspecting working scripts (including the one used to produce this presentation). You can find some on my github: https://github.com/ErikKusch.

`Rmarkdown` Resources

I cannot cover every aspect of `Rmarkdown` coding in one go. In case you bite, here are resources to help you along:

- **Guides**
  - An Introduction: https://rmarkdown.rstudio.com/lesson-1.html
  - The *'Definitive Guide'*: https://bookdown.org/yihui/rmarkdown/
- **CheatSheets**
  - Reference Guide: https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
  - Proper CheatSheet: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

→ I recommend learning `Rmarkdown` by inspecting working scripts (including the one used to produce this presentation). You can find some on my github: https://github.com/ErikKusch.

Overview

`Rmarkdown` files usually come down to **three main components**:

1. The *YAML Header* - This sets up important characteristics of your document such as:
   - Title, Subtitle
   - Author and Date
   - Output File Type
   - Document Class and Packages (if output type is LaTeX dependant)
2. *Plain Text* - This is where you write the text of your document. LaTeX formatting is supported when the YAML Header indicates a LaTeX dependant output file type.
3. *Code Chunks* - Here, you will implement R codes (Matlab and Python coding is supported as well) and run your analyses.

Overview

`Rmarkdown` files usually come down to **three main components**:

1. The *YAML Header* - This sets up important characteristics of your document such as:
   - Title, Subtitle
   - Author and Date
   - Output File Type
   - Document Class and Packages (if output type is LaTeX dependant)
2. *Plain Text* - This is where you write the text of your document. LaTeX formatting is supported when the YAML Header indicates a LaTeX dependant output file type.
3. *Code Chunks* - Here, you will implement R codes (Matlab and Python coding is supported as well) and run your analyses.

Overview

`Rmarkdown` files usually come down to **three main components**:

1. The *YAML Header* - This sets up important characteristics of your document such as:
   - Title, Subtitle
   - Author and Date
   - Output File Type
   - Document Class and Packages (if output type is LaTeX dependant)
2. *Plain Text* - This is where you write the text of your document. LaTeX formatting is supported when the YAML Header indicates a LaTeX dependant output file type.
3. *Code Chunks* - Here, you will implement R codes (Matlab and Python coding is supported as well) and run your analyses.

Overview

`Rmarkdown` files usually come down to **three main components**:

1. The *YAML Header* - This sets up important characteristics of your document such as:
   - Title, Subtitle
   - Author and Date
   - Output File Type
   - Document Class and Packages (if output type is LaTeX dependant)
2. *Plain Text* - This is where you write the text of your document. LaTeX formatting is supported when the YAML Header indicates a LaTeX dependant output file type.
3. *Code Chunks* - Here, you will implement R codes (Matlab and Python coding is supported as well) and run your analyses.

YAML Header

A non-exhaustive list of YAML specifiers:

YAML	Function
title	The heading of your work
subtitle	A subheading for your work
author	You and co-workers (if applicable)
date	Date of delivery or presentation
output	Whether to create html, pdf, or presentation
includes	Packages and auxiliary files
documentclass	LaTeX document class (if applicable)
classoption	LaTeX document option (if applicable)
fontsize	Base font size throughout document

Text

Formatting can offset the inherently boring nature of plain text and can be done:

- Through base syntax
- Through LaTeX syntax

Text is *hard coded* but results and code outputs can be included through *soft-coding* via **in-line code**.

Text can also be supplied via **auxillary files** using the `child` syntax supplied by `Rmarkdown`. This is useful for big documents but makes cross-referencing very complicated.

Text

Formatting can offset the inherently boring nature of plain text and can be done:

- Through base syntax
- Through LaTeX syntax

Text is *hard coded* but results and code outputs can be included through *soft-coding* via **in-line code**.

Text can also be supplied via **auxillary files** using the `child` syntax supplied by `Rmarkdown`. This is useful for big documents but makes cross-referencing very complicated.

Text

Formatting can offset the inherently boring nature of plain text and can be done:

- Through base syntax
- Through LaTeX syntax

Text is *hard coded* but results and code outputs can be included through *soft-coding* via **in-line code**.

Text can also be supplied via **auxillary files** using the `child` syntax supplied by `Rmarkdown`. This is useful for big documents but makes cross-referencing very complicated.

Code Chunks

R Code chunks are started with ```` ```{r} ```` and ended with ```` ``` ````. Chunks are supplied with optional arguments within the *{}* brackets. A non-exhaustive list of **chunk options**:

Option	Function
name	A name for the chunk, these have to be individual
eval	`TRUE` if code in chunk should be executed
include	`TRUE` if chunk should be run and displayed in final document
echo	`TRUE` if code will be displayed
tidy	`TRUE` if code should be displayed according to `tidy` formatting rules
cache	`TRUE` if cache for chunk is to be stored for faster compilation on repeated knitting

In-Line Code

In-line code can be used to display code output alongside plain text. This is done as follows:

```
# establish and R object to reference in-line
Answer <- 42
```

Then, you write `` `r Answer` `` and receive the output 42.

This is especially useful when wanting to generate reports that may be re-evaluated using different analyses or novel data sets as results may change.

Figures and `Rmarkdown`

Using `Rmarkdown` *for data visualisation* is **extremely useful**:

- Figures are generated on each compilation (unless cache is stored) thus always being up-to-date with the underlying data
- Sub-figures are established automatically as long as sub-captions are supplied
- Figures are registered automatically for the table of figures
- Figure placement on the page is fixed (no dragging around as in MS Word)
- No scaling artefacts
- **Entire workflow can happen in just one document!**

`Rmarkdown` **can handle all plotting outputs** of `R` coding including `ggplot2` plots.

Figures and `Rmarkdown`

Using `Rmarkdown` *for data visualisation* is **extremely useful**:

- Figures are generated on each compilation (unless cache is stored) thus always being up-to-date with the underlying data
- Sub-figures are established automatically as long as sub-captions are supplied
- Figures are registered automatically for the table of figures
- Figure placement on the page is fixed (no dragging around as in MS Word)
- No scaling artefacts
- **Entire workflow can happen in just one document!**

`Rmarkdown` **can handle all plotting outputs** of `R` coding including `ggplot2` plots.

Figures and `Rmarkdown`

Using `Rmarkdown` *for data visualisation* is **extremely useful**:

- Figures are generated on each compilation (unless cache is stored) thus always being up-to-date with the underlying data
- Sub-figures are established automatically as long as sub-captions are supplied
- Figures are registered automatically for the table of figures
- Figure placement on the page is fixed (no dragging around as in MS Word)
- No scaling artefacts
- **Entire workflow can happen in just one document!**

`Rmarkdown` **can handle all plotting outputs** of `R` coding including `ggplot2` plots.

Setting Code Up

Basic options for coding chunks and kitting options should be defined at the beginning of the `Rmarkdown` document:

```
# R options
options(scipen=1, digits=4) # output options such as rounding digits

# Knitting options
knitr::opts_chunk$set(echo = TRUE, # show code
                      message = FALSE, # don't display code messages
                      warning = FALSE, # don't display code warnings
                      cache = TRUE, # establish chaches
                      cache.lazy=TRUE, # don't check caches on re-run
                      tidy=TRUE, # format tidy by default
                      tidy.opts=list(width.cutoff=50), # width cutoff
                      fig.height=8, digits=4 # default figures sizes
                      )
```

This chunk should usually be set to `include = FALSE` to run the code but not show any of it.

Text Formatting

Basic `Rmarkdown` formatting can be used to generate more lively text:

Formatting	Result
Plain text	Plain text
italics and _italics_	*italics* and *italics*
bold and __bold__	**bold** and **bold**
superscript^2^	superscript^2
~~strikethrough~~	~~strikethrough~~

A line break in `Rmakrdown` can be achieved by ending a line with two spaces.

More advanced text formatting is possible via the LaTeX engine. An exhaustive guides for LaTeX formatting can be found here (this covers almost all matters LaTeX): http://www.rpi.edu/dept/arc/docs/latex/latex-intro.pdf.

Text Formatting

Basic `Rmarkdown` formatting can be used to generate more lively text:

Formatting	Result
Plain text	Plain text
italics and _italics_	*italics* and *italics*
bold and __bold__	**bold** and **bold**
superscript^2^	superscript^2
~~strikethrough~~	~~strikethrough~~

A line break in `Rmakrdown` can be achieved by ending a line with two spaces.

More advanced text formatting is possible via the LaTeX engine. An exhaustive guides for LaTeX formatting can be found here (this covers almost all matters LaTeX): http://www.rpi.edu/dept/arc/docs/latex/latex-intro.pdf.

Text Tables

Rmarkdown is capable of generating tables through plain text inputs:

```
Table Header | Second Header
------------ | -------------
Table Cell | Cell 2
Cell 3 | Cell 4
```

turns into:

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

Figures

Figures which aren't generated via R but present as files on the hard drive have to be included into the Rmarkdown document the LaTeX way.

This is done using the **includegraphics** command in the **graphicx** package for LaTeX. The output of this command is sensitive to other environment specifiers such as floats (*centering, raggedright, etc.*).

A LaTeX **figure environment** is useful for *including figures* which have been included this way *into the table of figures and make them available for cross-referencing*.

Figures

Figures which aren't generated via R but present as files on the hard drive have to be included into the Rmarkdown document the LaTeX way.

This is done using the **includegraphics** command in the **graphicx** package for LaTeX. The output of this command is sensitive to other environment specifiers such as floats (*centering, raggedright, etc.*).

A LaTeX **figure environment** is useful for *including figures* which have been included this way *into the table of figures and make them available for cross-referencing*.

Figures

Figures which aren't generated via `R` but present as files on the hard drive have to be included into the `Rmarkdown` document the LaTeX way.

This is done using the **includegraphics** command in the **graphicx** package for LaTeX. The output of this command is sensitive to other environment specifiers such as floats (*centering, raggedright, etc.*).

A LaTeX **figure environment** is useful for *including figures* which have been included this way *into the table of figures and make them available for cross-referencing*.

kable Tables I

The kable functionality in Rmarkdown enables us to generate appealing LaTeX -styled tables directly via R coding:

```
library(kableExtra)
tabout <- as.data.frame(matrix(c(1:20), nrow = 5))
rownames(tabout) <- c("R1", "R2", "R3", "R4", "R5")
colnames(tabout) <- c("C1", "C2", "C3", "C4")
kab <- kable(tabout, booktabs = TRUE, caption = paste("\\textbf{",
    "Short Description", " -} ", "Long Description",
    sep = ""), caption.short = "Short Description",
    escape = FALSE, format = "latex")
kab <- add_header_above(kable_input = kab, header = c(` ` = 1,
    `First two columns` = 2, `Last two columns` = 2),
    bold = TRUE, align = "c")
kab <- kable_styling(kab, latex_options = c("hold_position"))
print(kab)
```

Code chunks which generate kable tables need an additional argument: `results = 'asis'`.

kable Tables II

Table 5: **Short Description -** Long Description

	First two columns		Last two columns	
	C1	C2	C3	C4
R1	1	6	11	16
R2	2	7	12	17
R3	3	8	13	18
R4	4	9	14	19
R5	5	10	15	20

It is obvious how this way of table generation **enables reproducible research** as this tables can be directly generated from data and analyses outputs.

R Functions

Complex data analysis and data visualisation often invokes **user-generated R functions**. These should be *presented alongside all other results and data*.

Rmarkdown enables this in a three-step process:

1. **Write Function** in a chunk with `echo` set to `FALSE` (preferably in preamble)
2. **Call Function** into action in another chunk where it is needed
3. **Present Function** using another chunk which is set to `eval = FALSE` by calling the chunk option `ref.label='name'`, where `name` represent the name of the chunk established in step 1

R Functions

Complex data analysis and data visualisation often invokes **user-generated R functions**. These should be *presented alongside all other results and data*.

Rmarkdown enables this in a three-step process:

1. **Write Function** in a chunk with `echo` set to `FALSE` (preferably in preamble)
2. **Call Function** into action in another chunk where it is needed
3. **Present Function** using another chunk which is set to `eval = FALSE` by calling the chunk option `ref.label='name'`, where `name` represent the name of the chunk established in step 1

Cross-Referencing

Cross-referencing is vital for manuscripts writing and easy to implement in `Rmarkdown` when run through LaTeX:

Sections

- Index section by generating a label via the *label* command.
- Call section index via the *ref* command.

Figures

- Index figure via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Tables

- Index table via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Chunks

- Generate labelling automation using the *floatrow* package for LaTeX.
- Reference using the *ref* command and name of the chunk in question.

Cross-Referencing

Cross-referencing is vital for manuscripts writing and easy to implement in `Rmarkdown` when run through LaTeX:

Sections

- Index section by generating a label via the *label* command.
- Call section index via the *ref* command.

Figures

- Index figure via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Tables

- Index table via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Chunks

- Generate labelling automation using the *floatrow* package for LaTeX.
- Reference using the *ref* command and name of the chunk in question.

Cross-Referencing

Cross-referencing is vital for manuscripts writing and easy to implement in `Rmarkdown` when run through LaTeX:

Sections

- Index section by generating a label via the *label* command.
- Call section index via the *ref* command.

Figures

- Index figure via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Tables

- Index table via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Chunks

- Generate labelling automation using the *floatrow* package for LaTeX.
- Reference using the *ref* command and name of the chunk in question.

Cross-Referencing

Cross-referencing is vital for manuscripts writing and easy to implement in `Rmarkdown` when run through LaTeX:

Sections

- Index section by generating a label via the *label* command.
- Call section index via the *ref* command.

Figures

- Index figure via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Tables

- Index table via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Chunks

- Generate labelling automation using the *floatrow* package for LaTeX.
- Reference using the *ref* command and name of the chunk in question.

Cross-Referencing

Cross-referencing is vital for manuscripts writing and easy to implement in `Rmarkdown` when run through LaTeX:

Sections

- Index section by generating a label via the *label* command.
- Call section index via the *ref* command.

Figures

- Index figure via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Tables

- Index table via the *label* command in LaTeX environment or chunk option.
- Reference using the *ref* command.

Chunks

- Generate labelling automation using the *floatrow* package for LaTeX.
- Reference using the *ref* command and name of the chunk in question.

Bibliographies and Glossaries

Including a bibliography or a glossary is a necessity in manuscript writing:

Bibliography

- Install JabRef
- Generate .bib bibliography file
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *natbib*
- Reference using the *cite* command
- Establish bibliography suing the *bibliography command*

Glossary

- Install Perl
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *glossaries*
- Establish glossary entries in .tex file or preamble to document file
- Establish glossary at desired location in document using the *printglossary* command
- Use glossary commands in text
- Compiling requires an extra step through your LaTeX compiler for glossary to show up

Bibliographies and Glossaries

Including a bibliography or a glossary is a necessity in manuscript writing:

Bibliography

- Install JabRef
- Generate .bib bibliography file
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *natbib*
- Reference using the *cite* command
- Establish bibliography suing the *bibliography command*

Glossary

- Install Perl
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *glossaries*
- Establish glossary entries in .tex file or preamble to document file
- Establish glossary at desired location in document using the *printglossary* command
- Use glossary commands in text
- Compiling requires an extra step through your LaTeX compiler for glossary to show up

Bibliographies and Glossaries

Including a bibliography or a glossary is a necessity in manuscript writing:

Bibliography

- Install JabRef
- Generate .bib bibliography file
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *natbib*
- Reference using the *cite* command
- Establish bibliography suing the *bibliography command*

Glossary

- Install Perl
- In YAML header or auxiliary .tex file included in YAML header, load LaTeX package *glossaries*
- Establish glossary entries in .tex file or preamble to document file
- Establish glossary at desired location in document using the *printglossary* command
- Use glossary commands in text
- Compiling requires an extra step through your LaTeX compiler for glossary to show up

Beamer

`Beamer` is the LaTeX way of generating presentation slides and comes with a few quirks:

- Strong limitations on layout (it is best to chose from pre-established styles)
- No animations possible
- Stiff and unwieldy positioning of items

These limitations often lead to *headaches when generating the presentations* but **much cleaner looking presentations** with **consistent formatting**.

LaTeX column environment (used in `Beamer` presentations) do not allow for the evaluation of `Rmarkdown` syntax. This can be circumvented by defining novel commands for the column environment instead of calling the default LaTeX commands.

Beamer

`Beamer` is the LaTeX way of generating presentation slides and comes with a few quirks:

- Strong limitations on layout (it is best to chose from pre-established styles)
- No animations possible
- Stiff and unwieldy positioning of items

These limitations often lead to *headaches when generating the presentations* but **much cleaner looking presentations** with **consistent formatting**.

LaTeX column environment (used in `Beamer` presentations) do not allow for the evaluation of `Rmarkdown` syntax. This can be circumvented by defining novel commands for the column environment instead of calling the default LaTeX commands.

Beamer

`Beamer` is the LaTeX way of generating presentation slides and comes with a few quirks:

- Strong limitations on layout (it is best to chose from pre-established styles)
- No animations possible
- Stiff and unwieldy positioning of items

These limitations often lead to *headaches when generating the presentations* but **much cleaner looking presentations** with **consistent formatting**.

LaTeX column environment (used in `Beamer` presentations) do not allow for the evaluation of `Rmarkdown` syntax. This can be circumvented by defining novel commands for the column environment instead of calling the default LaTeX commands.

Beamer

`Beamer` is the LaTeX way of generating presentation slides and comes with a few quirks:

- Strong limitations on layout (it is best to chose from pre-established styles)
- No animations possible
- Stiff and unwieldy positioning of items

These limitations often lead to *headaches when generating the presentations* but **much cleaner looking presentations** with **consistent formatting**.

LaTeX column environment (used in `Beamer` presentations) do not allow for the evaluation of `Rmarkdown` syntax. This can be circumvented by defining novel commands for the column environment instead of calling the default LaTeX commands.