

oxforddown:
An Oxford University Thesis
Template for R Markdown

Author Name

Your College
University of Oxford

*A thesis submitted for the degree of
Doctor of Philosophy*

Michaelmas 2018

Abstract

This *R Markdown* template is for writing an Oxford University thesis. The template is built using Yihui Xie's `bookdown` package, with heavy inspiration from Chester Ismay's `thesisdown`, and the `OxThesis` L^AT_EX template (most recently adapted by John McManigle).

This template's sample content include illustrations of how to do the various things you need to write a thesis in R Markdown, and largely follow the structure from this R Markdown workshop.

Congratulations for taking a step further into the lands of open, reproducible science, by writing your thesis using a tool that allows you to transparently include tables and dynamically generated plots directly from the underlying data. Hip hooray!

oxforddown:
An Oxford University Thesis
Template for R Markdown



Author Name
Your College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy
Michaelmas 2018

For Yihui Xie

Acknowledgements

This is where you will normally thank your advisor, colleagues, family and friends, as well as funding and institutional support. In our case, we will give our praises to the people who developed the ideas and tools that allow us to push open science a little step forward by writing plain-text, transparent, and reproducible theses in R Markdown.

We must be grateful to John Gruber for inventing the original version of Markdown, to John MacFarlane for creating Pandoc (<http://pandoc.org>) which converts Markdown to a large number of output formats, and to Yihui Xie for creating **knitr** which introduced R Markdown as a way of embedding code in Markdown documents, and **bookdown** which added tools for technical and longer-form writing.

Special thanks to Chester Ismay, who created the **thesisdown** package that helped many a PhD student write their theses in R Markdown. And a very special tahnks to John McManigle, whose adaption of Sam Evans' adaptation of Keith Gillow's original maths template for writing an Oxford University DPhil thesis in L^AT_EX provided the template that I adapted for R Markdown.

Finally, profuse thanks to JJ Allaire, the founder and CEO of RStudio, and Hadley Wickham, the mastermind of the tidyverse without whom we'd all just given up and done data science in Python instead. Thanks for making data science easier, more accessible, and more fun for us all.

Ulrik Lyngs
Linacre College, Oxford
2 December 2018

Abstract

This *R Markdown* template is for writing an Oxford University thesis. The template is built using Yihui Xie's `bookdown` package, with heavy inspiration from Chester Ismay's `thesisdown`, and the `OxThesis` L^AT_EX template (most recently adapted by John McManigle).

This template's sample content include illustrations of how to do the various things you need to write a thesis in R Markdown, and largely follow the structure from this R Markdown workshop.

Congratulations for taking a step further into the lands of open, reproducible science, by writing your thesis using a tool that allows you to transparently include tables and dynamically generated plots directly from the underlying data. Hip hooray!

Contents

List of Figures	vii
List of Tables	viii
List of Abbreviations	ix
Introduction	1
Why use it?	1
Who should use it?	2
1 R Markdown Basics: The Markdown syntax	3
1.1 Markdown basic syntax	4
1.1.1 Italics and bold	4
1.1.2 Inline code	4
1.1.3 Sub and superscript	4
1.1.4 Strikethrough	4
1.1.5 ‘Escaping’ (aka “What if I need an actual asterisk?”)	4
1.1.6 Endash (–), emdash (—)	4
1.1.7 Blockquotes	5
1.1.8 Headings	5
1.1.9 Lists	5
1.1.10 Line breaks	6
1.1.11 Hyperlinks	6
1.1.12 Footnotes	6
1.1.13 Comments	7
1.1.14 Math	7
1.2 Additional resources	8

2	Adding code	9
2.1	Code chunks	9
2.1.1	Setup chunks	10
2.1.2	Including images	11
2.1.3	Including plots	12
2.1.4	Including tables	13
2.1.5	A note on content positioning	14
2.2	Inline code	14
2.2.1	Referring to results computed in other languages than R . . .	15
3	Citations and cross-references	16
3.1	Citations	16
3.1.1	PDF output	17
3.1.2	Gitbook output	18
3.1.3	Insert references easily with the <code>citr</code> add-in	19
3.2	Cross-referencing	19
3.2.1	Section references	19
3.2.2	Figure (image and plot) references	20
3.2.3	Table references	20
3.2.4	Including page numbers	21
4	Final Notes on The OxThesis template and on collaboration	23
4.1	Beginning chapters with quotes	23
4.2	Highlighting corrections	24
4.2.1	Short, inline corrections	24
4.2.2	Blocks of added or changed material	24
4.2.3	Stopping corrections from being highlighted in the output . .	24
4.3	Diving in to the OxThesis LaTeX template	25
4.4	Collaborative writing	25
	Conclusion	26
	Appendices	
A	The First Appendix	28
B	The Second Appendix, for Fun	29
	Works Cited	30

List of Figures

2.1	Code chunk syntax	10
2.2	Oxford logo	11
2.3	Oxford logo, rotated	12
2.4	A ggplot of car stuff	13
3.1	The ‘citr’ add-in	19
3.2	A marvel-lous meme	21

List of Tables

2.1	A knitr kable table	13
3.1	Stopping cars	21

List of Abbreviations

- 1-D, 2-D** . . . One- or two-dimensional, referring in this thesis to spatial dimensions in an image.
- Otter** One of the finest of water mammals.
- Hedgehog** . . . Quite a nice prickly friend.

Introduction

Welcome to the *R Markdown* Oxford University thesis template. This sample content is adapted from **thesisdown** and the formatting of PDF output is adapted from the OxThesis L^AT_EX template. Hopefully, writing your thesis in R Markdown will provide a nicer interface for to the OxThesis template for those that have never used TeX or LaTeX before. More importantly, using *R Markdown* allows you to embed chunks of code directly into your thesis, and generate plots and tables directly from their underlying data, avoiding copy-paste steps. Finally, this *R Markdown* template gets you in the habit of doing reproducible research, which benefits you long-term as a researcher, but also will greatly help anyone that is trying to reproduce or build onto your results down the road.

The use of LaTeX in combination with *Markdown* is more consistent than the output of a word processor, much less prone to corruption or crashing, and the resulting file is smaller than a Word file. While you may never have had problems using Word in the past, your thesis is likely going to be about twice as large and complex as anything you've written before, taxing Word's capabilities.

Why use it?

R Markdown creates a simple and straightforward way to interface with the beauty of LaTeX. Packages have been written in **R** to work directly with LaTeX to produce nicely formatting tables and paragraphs. In addition to creating a user friendly interface to LaTeX, *R Markdown* also allows you to read in your data, to analyze it and to visualize it using **R**, **Python** or other languages, and provide the documentation and commentary on the results of your project.

Further, it allows for results of code output to be passed inline to the commentary

of your results. You'll see more on this later, focusing on **R**. If you are more into **Python** or something else, you can still use *R Markdown* - see 'Other language engines' in Yihui Xie's *R Markdown: The Definitive Guide*.

Who should use it?

Anyone who needs to use data analysis, math, tables, a lot of figures, complex cross-references, or who just cares about reproducibility in research can benefit from using *R Markdown*. If you are working in 'softer' fields, the user-friendly nature of the *Markdown* syntax and its ability to keep track of and easily include figures, automatically generate a table of contents, index, references, table of figures, etc. should still make it of great benefit to your thesis project.

Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit...

There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain...

— Cicero's *de Finibus Bonorum et Malorum*.

1

R Markdown Basics: The Markdown syntax

Contents

1.1	Markdown basic syntax	4
1.1.1	Italics and bold	4
1.1.2	Inline code	4
1.1.3	Sub and superscript	4
1.1.4	Strikethrough	4
1.1.5	‘Escaping’ (aka “What if I need an actual asterisk?”)	4
1.1.6	Endash (–), emdash (—)	4
1.1.7	Blockquotes	5
1.1.8	Headings	5
1.1.9	Lists	5
1.1.10	Line breaks	6
1.1.11	Hyperlinks	6
1.1.12	Footnotes	6
1.1.13	Comments	7
1.1.14	Math	7
1.2	Additional resources	8

Here is a brief introduction to using *R Markdown*. *Markdown* is a simple formatting syntax for authoring HTML, PDF, and MS Word documents and much, much more. *R Markdown* provides the flexibility of *Markdown* with the implementation of **R** input and output. For more details on using *R Markdown*

see <http://rmarkdown.rstudio.com>.

Be careful with your spacing in *Markdown* documents. While whitespace largely is ignored, it does at times give *Markdown* signals as to how to proceed. As a habit, try to keep everything left aligned whenever possible, especially as you type a new paragraph. In other words, there is no need to indent basic text in the Rmd document (in fact, it might cause your text to do funny things if you do).

1.1 Markdown basic syntax

1.1.1 Italics and bold

- *Italics* are done like `*this*` or `_this_`
- **Bold** is done like `**this**` or `__this__`
- ***Bold and italics*** is done like `***this***`, `____this____`, or (the most transparent solution) `**_this_**`

1.1.2 Inline code

- Inline code is created with backticks like ``this``

1.1.3 Sub and superscript

Sub₂ and super² script is created like `this~2~` and `this^2^`

1.1.4 Strikethrough

- ~~Strikethrough~~ is done `~~like this~~`

1.1.5 ‘Escaping’ (aka “What if I need an actual asterisk?”)

- To include an actual `*`, `_` or `\`, add another `\` in front of them: `*`, `_`, `\\`

1.1.6 Endash (—), emdash (—)

- `—` and `—` with `-` `--` and `---`

1.1.7 Blockquotes

Do like this:

Put a > in front of the line.

1.1.8 Headings

- are done with #’s of increasing number, i.e.
 - # First-level heading
 - ## Second-level heading
 - ### Etc.

In PDF output, a level-five heading will turn into a paragraph heading, i.e. `\paragraph{My level-five heading}`, which will appear as bold text on the same line as the subsequent paragraph.

1.1.9 Lists

Unordered list by starting a line with an * or a -:

- Item 1
- Item 2

Ordered lists by starting a line with a number:

1. Item 1
2. Item 2

Notice that you can mislabel the numbers and *Markdown* will still make the order right in the output.

To create a sublist, indent the values a bit (at least four spaces or a tab):

1. Item 1
2. Item 2

3. Item 3

- Item 3a
- Item 3b

1.1.10 Line breaks

The official *Markdown* way to create line breaks is by ending a line with more than two spaces.

Roses are red. Violets are blue.

This appears on the same line in the output, because we didn't add spaces after red.

Roses are red.

Violets are blue.

This appears with a line break because I added spaces after red.

I find this is confusing, so I recommend the alternative way: Ending a line with a backslash will also create a linebreak:

Roses are red.

Violets are blue.

To create a new paragraph, you put a blank line.

Therefore, this line starts its own paragraph.

1.1.11 Hyperlinks

- This is a hyperlink created by writing the text you want turned into a clickable link in `[square brackets followed by a](https://hyperlink-in-parentheses)`

1.1.12 Footnotes

- Are created¹ by writing either `^[my footnote text]` for supplying the footnote content inline, or something like `[^a-random-footnote-label]` and

¹my footnote text

supplying the text elsewhere in the format shown below ²:

```
[^a-random-footnote-label]: This is a random test.
```

1.1.13 Comments

To write comments within your text that won't actually be included in the output, you use the same syntax as for writing comments in HTML. That is, `<!-- this will not be included in the output -->`.

1.1.14 Math

The syntax for writing math is stolen from LaTeX. To write a math expression that will be shown **inline**, enclose it in dollar signs. - This: `$A = \pi*r^{2}$`
Becomes: $A = \pi * r^2$

To write a math expression that will be shown in a block, enclose it in two dollar signs.

This: `$$A = \pi*r^{2}$$`

Becomes:

$$A = \pi * r^2$$

To create numbered equations, put them in an 'equation' environment and give them a label with the syntax `(\#eq:label)`, like this:

```
\begin{equation}
  f\left(k\right) = \binom{n}{k} p^k\left(1-p\right)^{n-k}
  (\#eq:binom)
\end{equation}
```

Becomes:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (1.1)$$

For more (e.g. how to theorems), see e.g. the documentation on bookdown.org

²This is a random test.

1.2 Additional resources

- *R Markdown: The Definitive Guide* - <https://bookdown.org/yihui/rmarkdown/>
- *R for Data Science* - <https://r4ds.had.co.nz>

2

Adding code

Contents

2.1	Code chunks	9
2.1.1	Setup chunks	10
2.1.2	Including images	11
2.1.3	Including plots	12
2.1.4	Including tables	13
2.1.5	A note on content positioning	14
2.2	Inline code	14
2.2.1	Referring to results computed in other languages than R	15

The magic of R Markdown is that we can add code within our document to make it dynamic.

We do this either as *code chunks* (generally used for loading libraries and data, performing calculations, and adding images, plots, and tables), or *inline code* (generally used for dynamically reporting results within our text).

2.1 Code chunks

The syntax of a code chunk is shown in Figure 2.1.

Common chunk options include (see e.g. bookdown.org):

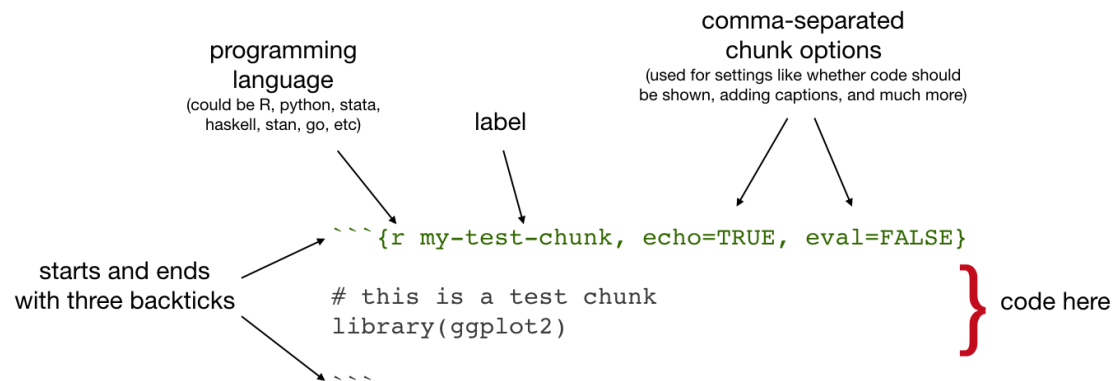


Figure 2.1: Code chunk syntax

- `echo`: whether or not to display code in knitted output
- `eval`: whether or not to run the code in the chunk when knitting
- `include`: whether to include anything from the code chunk in the output document
- `fig.cap`: figure caption

IMPORTANT: Do *not* use underscores in your chunk labels - if you do, you are likely to get an error in PDF output saying something like “! Package caption Error: \caption outside float”.

2.1.1 Setup chunks

An R Markdown document usually begins with a chunk that is used to **load libraries**, and to **set default chunk options** with `knitr::opts_chunk$set`.

In your thesis, this will probably happen in **index.Rmd** and/or as opening chunks in each of your chapters.

```

```{r setup, include=FALSE}
don't show code unless we explicitly set echo = TRUE
knitr::opts_chunk$set(echo = FALSE)

library(tidyverse)
```

```



Figure 2.2: Oxford logo

2.1.2 Including images

Code chunks are also used for including images, with `include_graphics` from the `knitr` package, as in Figure 2.2

```
knitr::include_graphics("figures/beltcrest.png")
```

Useful chunk options for figures include: - `out.width` (use with a percentage) for setting the image size - if you've got an image that gets waaay to big in your output, it will be constrained to the page width by setting `out.width = "100%"`

Figure rotation

You can use the chunk option `out.extra` to rotate images.

The syntax is different for LaTeX and HTML, so for ease we might start by assigning the right string to a variable that depends on the format you're outputting to:

```
if (knitr::opts_knit$get('rmarkdown.pandoc.to') == 'latex'){
  rotate180 <- "angle=180"
} else {
```

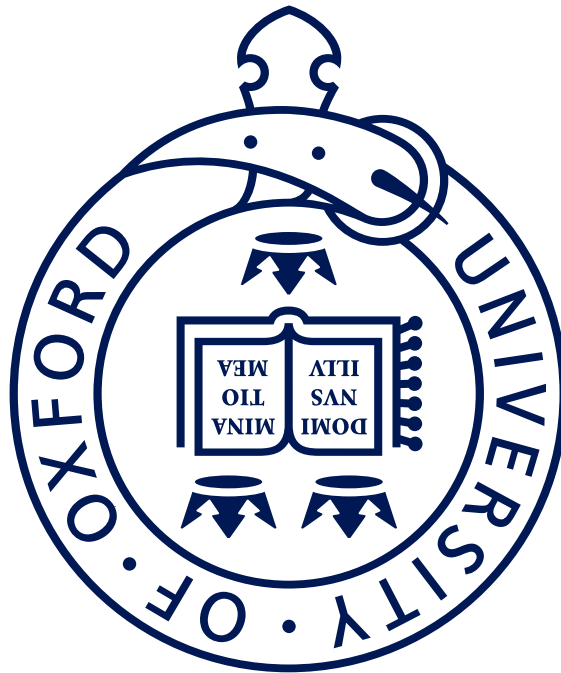


Figure 2.3: Oxford logo, rotated

```
rotate180 <- "style='transform:rotate(180deg);'"
}
```

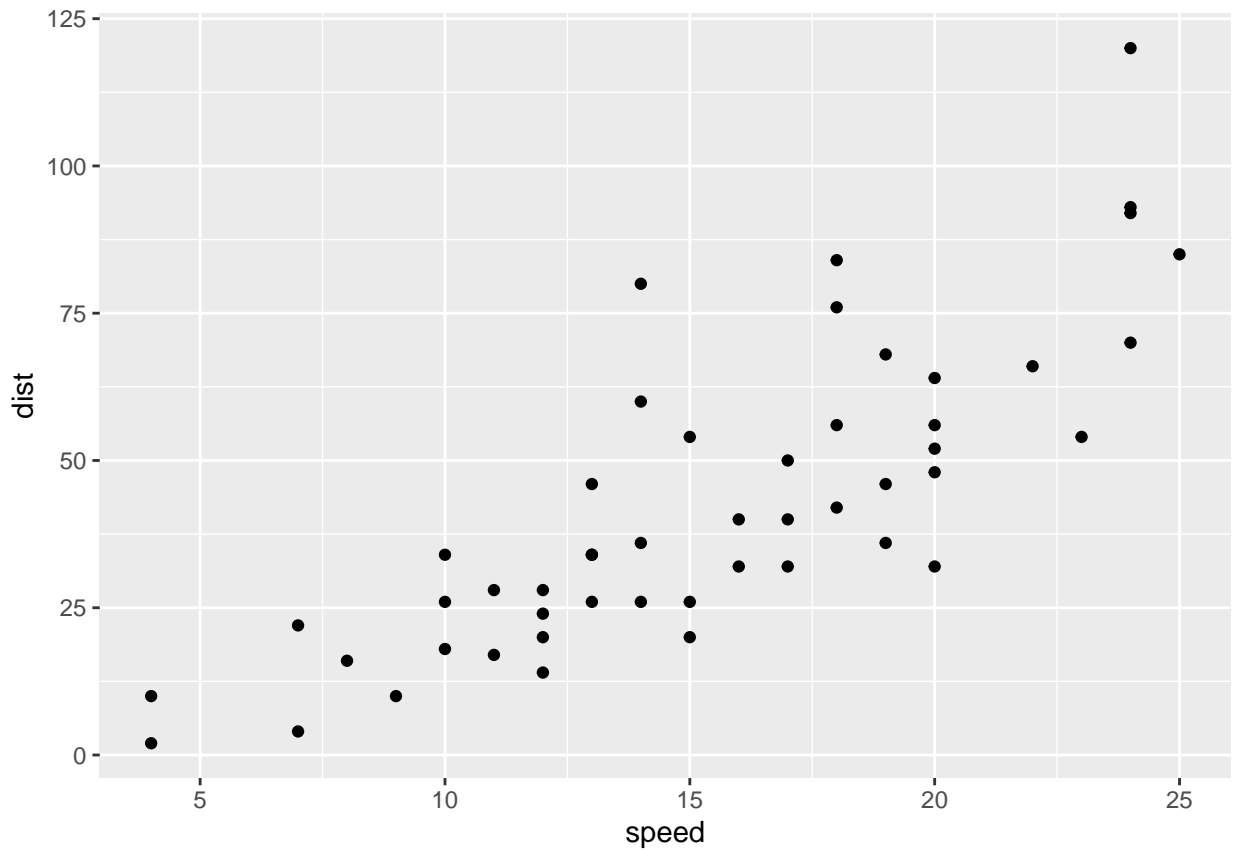
Then you can reference that variable as the value of `out.extra` to rotate images, as in Figure 2.3.

2.1.3 Including plots

Similarly, code chunks are used for including dynamically generated plots. You use ordinary code in R or other languages - Figure 2.4 shows a plot of the `cars` dataset of stopping distances for cars at various speeds (this dataset is built in to **R**).

```
cars %>%
  ggplot() +
  aes(x = speed, y = dist) +
  geom_point()
```

Under the hood, plots are included in your document in the same way as images - when you build the book or knit a chapter, the plot is automatically generated from your code, saved as an image, then included into the output document.

**Figure 2.4:** A ggplot of car stuff**Table 2.1:** A knitr kable table

| speed | dist |
|-------|------|
| 4 | 2 |
| 4 | 10 |
| 7 | 4 |
| 7 | 22 |
| 8 | 16 |
| 9 | 10 |

2.1.4 Including tables

Tables are usually included with the `kable` function from the `knitr` package.

Table 2.1 shows the first rows of that cars data - read in your own data, then use this approach to automatically generate tables.

```
cars %>%
  head() %>%
  knitr::kable(caption = "A knitr kable table")
```

- Gotcha: when using `kable`, captions are set inside the `kable` function
- The `kable` package is often used with the `kableExtra` package

2.1.5 A note on content positioning

One thing that may be annoying is the way *R Markdown* handles “floats” like tables and figures.

In your PDF output, LaTeX will try to find the best place to put your object based on the text around it and until you’re really, truly done writing you should just leave it where it lies.

When the time comes for you to make final tweaks to content positioning, read the relevant R Markdown documentation to see if there are easy ways to do what you want.

If you have very specific needs, you might have to read up on LaTeX (https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions) for your PDF output and/or on how to style HTML documents with CSS for your gitbook output.

2.2 Inline code

‘Inline code’ simply means inclusion of code inside text.

The syntax for doing this is ``r R_CODE``

For example, ``r 4 + 4`` would output 8 in your text.

You will usually use this in parts of your thesis where you report results - read in data or results in a code chunk, store things you want to report in a variable, then insert the value of that variable in your text.

For example, we might assign the number of rows in the `cars` dataset to a variable:

```
num_car_observations <- nrow(cars)
```

We might then write:

“In the `cars` dataset, we have ``r num_car_observations`` observations.”

Which would output:

“In the `cars` dataset, we have 50 observations.”

2.2.1 Referring to results computed in other languages than R

At the moment, inline code only works with R, so syntax such as ``python code here`` is not valid. However, you can use the `reticulate` package to access variables from python chunks. Here’s a Python code chunk:

```
my_number = 4 + 8
print(my_number)
```

```
## 12
```

The `reticulate` package allows **R** to access variables defined in python environments with the syntax `py$variable`:

```
library(reticulate)
py$my_number
```

```
## [1] 12
```

This means that inline, we can include results from python chunks with ``r py$variable``. For example, we can state that the value of `my_number` defined in the python chunk, is 12.

3

Citations and cross-references

Contents

| | |
|--|-----------|
| 3.1 Citations | 16 |
| 3.1.1 PDF output | 17 |
| 3.1.2 Gitbook output | 18 |
| 3.1.3 Insert references easily with the <code>citr</code> add-in | 19 |
| 3.2 Cross-referencing | 19 |
| 3.2.1 Section references | 19 |
| 3.2.2 Figure (image and plot) references | 20 |
| 3.2.3 Table references | 20 |
| 3.2.4 Including page numbers | 21 |

3.1 Citations

The usual way to include citations in an *R Markdown* document is to put references in a plain text file with the extension **.bib**, in **BibTeX** format.¹ Then reference the path to this file in **index.Rmd**'s YAML header with **bibliography: example.bib**.

Most reference managers can create a .bib file with you references automatically. However, the **by far** best reference manager to use with *R Markdown* is Zotero

¹The bibliography can be in other formats as well, including EndNote (**.enl**) and RIS (**.ris**), see rmarkdown.rstudio.com/authoring_bibliographies_and_citations.

with the Better BibTex plug-in, because the `citr` plugin for RStudio (see below) can read references directly from your Zotero library!

Here is an example of an entry in a `.bib` file:

```
@article{Shea2014,
  author =      {Shea, Nicholas and Boldt, Annika},
  journal =      {Trends in Cognitive Sciences},
  pages =        {186--193},
  title =        {{Supra-personal cognitive control}},
  volume =        {18},
  year =          {2014},
  doi =           {10.1016/j.tics.2014.01.006},
}
```

In this entry highlighted section, ‘Shea2014’ is the **citation identifier**. To default way to cite an entry in your text is with this syntax: `[@citation-identifier]`.

So I might cite some things (Shea et al. 2014; Lottridge et al. 2012).

3.1.1 PDF output

In PDF output, the bibliography is handled by the OxThesis LaTeX template. If you set `bib-humanities: true` in `index.Rmd`, then in-text references will be formatted as author-year; otherwise references will be shown as numbers.

If you choose author-year formatting, a number of variations on the citation syntax are useful to know:

- Put author names outside the parenthesis
 - This: `@Shea2014 says blah.`
 - Becomes: Shea et al. (2014) says blah.
- Include only the citation-year (in parenthesis)
 - This: `Shea et al. says blah [-@Shea2014]`
 - Becomes: Shea et al. says blah (2014)

- Add text and page or chapter references to the citation
 - This: [see @Shea2014, pp. 33-35; also @Wu2016, ch. 1]
 - Becomes: Blah blah (see Shea et al. 2014, pp. 33-35; also Wu 2016, ch. 1).

3.1.2 Gitbook output

In gitbook output, citations are by default inserted in the Chicago author-date format.

To change the format, add `csl: some-other-style.csl` in **index.Rmd**'s YAML header. You can browse through and download styles at zotero.org/styles.

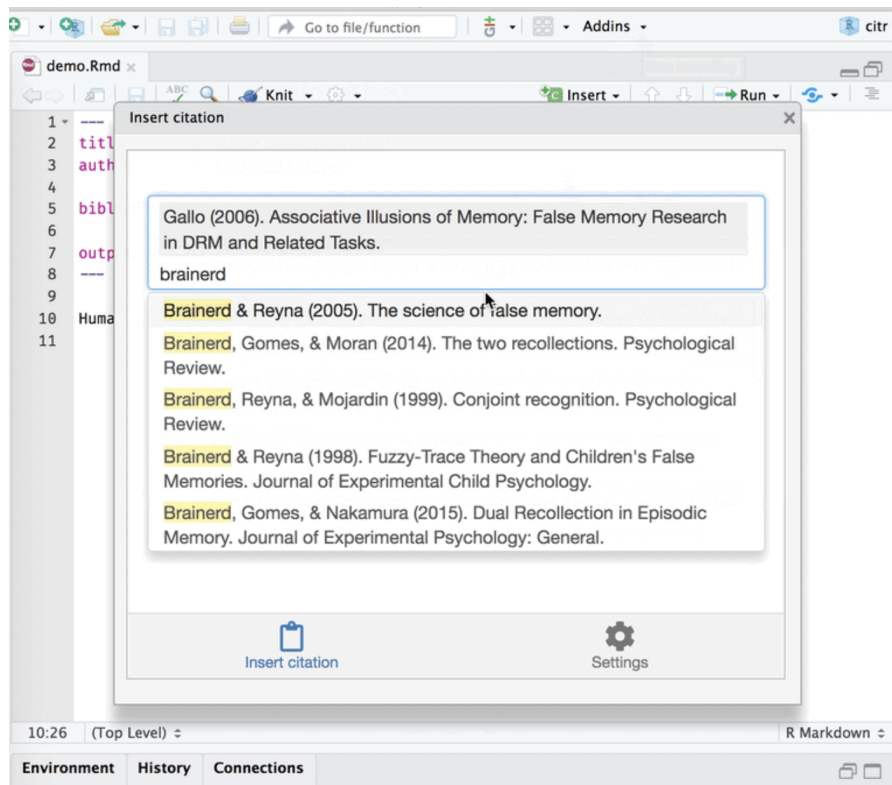


Figure 3.1: The ‘citr’ add-in

3.1.3 Insert references easily with the `citr` add-in

For an easy way to insert citations, try the `citr` RStudio add-in (Figure 3.1). You can install this add-in by typing `install.packages("citr")` in the R Console.

3.2 Cross-referencing

We can make cross-references to **sections** within our document, as well as to **figures** (images and plots) and **tables**.

The general cross-referencing syntax is `\@ref(label)`

3.2.1 Section references

Headers are automatically assigned a reference label, which is the text in lower caps separated by dashes. For example, `# My header` is automatically given the label `my-header`. So `# My header` can be referenced with `\@ref(my-section)`

Remember what we wrote in section 3.1?

We can also use **hyperlink syntax** and add `#` before the label, though this is only guaranteed to work properly in HTML output:

- So if we write `Remember what we wrote up in [the previous section](#citations)?`
- It becomes `Remember what we wrote up in the previous section?`

Creating custom labels

It is a very good idea to create **custom labels** for our sections. This is because the automatically assigned labels will change when we change the titles of the sections - to avoid this, we can create the labels ourselves and leave them untouched if we change the section titles.

We create custom labels by adding `{#label}` after a header, e.g. `# My section {#my-label}`. See our chapter title for an example. That was section 3.

3.2.2 Figure (image and plot) references

- To refer to figures (i.e. images and plots) use the syntax `\@ref(fig:label)`
- **GOTCHA:** Figures and tables must have captions if you wish to cross-reference them.

Let's add an image:

```
knitr::include_graphics("figures/captain.jpeg")
```

We refer to this image with `\@ref(fig:captain)`. So Figure 3.2 is this image.

And in Figure 2.4 we saw a cars plot.

3.2.3 Table references

- To refer to tables use the syntax `\@ref(tab:label)`

Let's include a table:

**Figure 3.2:** A marvel-lous meme**Table 3.1:** Stopping cars

| speed | dist |
|-------|------|
| 4 | 2 |
| 4 | 10 |
| 7 | 4 |
| 7 | 22 |
| 8 | 16 |

```
knitr::kable(cars[1:5,],
             caption="Stopping cars")
```

We refer to this table with `\@ref{tab:cars-table2}`. So Table 3.1 is this table.

And in Table 2.1 we saw more or less the same cars table.

3.2.4 Including page numbers

Finally, in the PDF output we might also want to include the page number of a reference, so that it's easy to find in physical printed output. LaTeX has a command for this, which looks like this: `\pageref{fig/tab:label}` (note: curly

braces, not parentheses)

When we output to PDF, we can use raw LaTeX directly in our .Rmd files. So if we wanted to include the page of the cars plot we could write:

- This: `Figure \@ref(fig:cars-plot) on page \pageref(fig:cars-plot)`
- Becomes: Figure 2.4 on page 13

Include page numbers only in PDF output

A problem here is that LaTeX commands don't display in HTML output, so in the gitbook output we'd see simply "Figure 2.4 on page".

One way to get around this is to use inline R code to insert the text, and use an `ifelse` statement to check the output format and then insert the appropriate text.

- So this: ``r ifelse(knitr::is_latex_output(), "Figure \@ref(fig:cars-plot) on page \pageref{fig:cars-plot}", "")``
- Inserts this (check this on both PDF and gitbook): Figure 2.4 on page 13

Note that we need to escape the backslash with another backslash here to get the correct output.

There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.

— Charles Darwin (Darwin 1859)

4

Final Notes on The OxThesis template and on collaboration

Contents

| | | |
|------------|---|-----------|
| 4.1 | Beginning chapters with quotes | 23 |
| 4.2 | Highlighting corrections | 24 |
| 4.2.1 | Short, inline corrections | 24 |
| 4.2.2 | Blocks of added or changed material | 24 |
| 4.2.3 | Stopping corrections from being highlighted in the output | 24 |
| 4.3 | Diving in to the OxThesis LaTeX template | 25 |
| 4.4 | Collaborative writing | 25 |

4.1 Beginning chapters with quotes

The OxThesis LaTeX template lets you inject some wittiness into your thesis by including a block of type `savequote` at the beginning of chapters. To do this, use the syntax ````{block type='savequote'}`.¹

Add the reference for the quote with the chunk option `quote_author="my author name"`. You will also want to add the chunk option `include=knitr::is_latex_output()` so that quotes are only included in PDF output.

¹For more on custom block types, see the relevant section in *Authoring Books with R Markdown*.

It's not possible to use markdown syntax inside chunk options, so if you want to e.g. italicise a book name in the reference use a 'text reference': Create a named piece of text with '(ref:label-name) My text', then point to this in the chunk option with `quote_author='(ref:label-name)'`.

4.2 Highlighting corrections

For when it comes time to do corrections, you may want to highlight changes made when you submit a post-viva, corrected copy to your examiners so they can quickly verify you've completed the task. You can do so like this:

4.2.1 Short, inline corrections

Highlight **short, inline corrections** by wrapping them in a `` tag with the class 'correction'. In other words, if you do `like this`, the text between the span tags will be highlighted in blue in the output.

4.2.2 Blocks of added or changed material

Highlight entire **blocks of added or changed material** by putting them in a block of type `correction`, using the syntax ````\begin{block type='correction'}````.² Like so:

For larger chunks, like this paragraph or indeed entire figures, you can use the `correction` block type. This environment **highlights paragraph-sized and larger blocks** with the same blue colour.

4.2.3 Stopping corrections from being highlighted in the output

For **PDF** output, go to `index.Rmd` and (i) set `corrections: false` under `params` in the YAML header (stops block of corrections from being highlighted), (ii) comment

²In the `.tex` file for PDF output, this will put the content between `\begin{correction}` and `\end{correction}`; in gitbook output it will be put between `<div class="correction">` and `</div>`.

out `pandoc_args`: `["--lua-filter=scripts_and_filters/correction_filter.lua"]`
 (stops inline corrections from being highlighted).

For **gitbook** output, go to **style.css** and comment out the styling for `.correction`.

4.3 Diving in to the OxThesis LaTeX template

For LaTeX minded people, you can read through **templates/template.tex** to see which additional customisation options are available as well as **templates/ociamthesis.cls** which supplies the base class. For example, **template.tex** provides an option for master's degree submissions, which changes identifying information to candidate number and includes a word count. At the time of writing, you must set this directly in **template.tex** rather than from the YAML header in **index.Rmd**.

4.4 Collaborative writing

Best practices for collaboration and change tracking when using R Markdown are still an open question. In the blog post **One year to dissertate** by Lucy D'Agostino, which I highly recommend, the author notes that she knits `.Rmd` files to a `word_document`, then uses the `googledrive` R package to send this to Google Drive for comments / revisions from co-authors, then incorporates Google Drive suggestions *by hand* into the `.Rmd` source files. This is a bit clunky, and there are ongoing discussions among the *R Markdown* developers about what the best way is to handle collaborative writing (see issue #1463 on GitHub, where CriticMarkup is among the suggestions).

For now, this is an open question in the community of R Markdown users. I often knit to a format that can easily be imported to Google Docs for comments, then go over suggested revisions and manually incorporate them back in to the `.Rmd` source files. For articles, I sometimes upload a near-final draft to Overleaf, then collaboratively make final edits to the \LaTeX file there. I suspect some great solution will be developed in the not-to-distant future, probably by the RStudio team.

*Alles Gescheite ist schon gedacht worden.
Man muss nur versuchen, es noch einmal zu denken.
All intelligent thoughts have already been thought;
what is necessary is only to try to think them again.*
— Johann Wolfgang von Goethe (von Goethe 1829)

Conclusion

If we don't want Conclusion to have a chapter number next to it, we can add the `{-}` attribute.

More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendices



The First Appendix

This first appendix includes an R chunk that was hidden in the document (using `echo = FALSE`) to help with readability:

In 02-rmd-basics-code.Rmd

```
library(tidyverse)
knitr::include_graphics("figures/chunk-parts.png")
```

And here's another one from the same chapter, i.e. Chapter 2:

```
knitr::include_graphics("figures/beltcrest.png")
```

B

The Second Appendix, for Fun

Works Cited

- Darwin, Charles (1859). *On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. London: John Murray.
- Lottridge, Danielle et al. (2012). “Browser design impacts multitasking”. In: *Proceedings of the Human Factors and Ergonomics Society 56th Annual Meeting*.
- Shea, Nicholas et al. (2014). “Supra-personal cognitive control and metacognition”. In: *Trends in Cognitive Sciences* 18.4, pp. 186–193. URL: <http://dx.doi.org/10.1016/j.tics.2014.01.006>.
- Von Goethe, Johann Wolfgang (1829). *Wilhelm Meisters Wanderjahre oder die Entsagenden*. de. Cotta.
- Wu, Tim (2016). *The Attention Merchants: The Epic Scramble to Get Inside Our Heads*. Knopf Publishing Group.