Getting Started with Thesis Writing in RMarkdown and $\LaTeX$

Christelinda Laureijs

A short guide for curious and motivated researchers

June 20, 2024

COLOPHON

This document was typeset in R Markdown and the text was set in EB Garamond. The data and code used to produce this document are available at *https://github.com/christelinda-laureijs/reproducible-thesis*.

Contents

List of Figures

Introduction

Writing a paper in R can be challenging at first, but it is a rewarding process that will save you lots of time, effort and stress over the long term. This document provides examples and code for you to get started. It is also a sample RMarkdown document to see if you can knit PDFs from R and RStudio. Ideally, you should be able to click on Knit -> Knit to PDF and generate this file as a PDF without issues.

A note for Mac Users: When you knit the document the first time, you might see error messages with something about Cairo and your graphics device. Every device is different, but you might be able to fix it by installing XQuartz and then Cairo.

A note for all users: If you're getting error messages about images not being found, don't forget to check your knit settings (dropdown arrow by the *Knit* icon). Click on the Project Directory setting. You may need to switch the Knit Directory from *Document Directory* to *Project Directory* or the other way around. If one directory isn't working, try the other!

What are the advantages of writing in R?

Reproducibility

Your analyses and text are all in one place, so you will always know what you did to create a specific plot or how you got a certain p-value. Others will be able to follow your process by reading your code, and if you make your code and data availble, they can replicate your analyses (and ideally end up with the same numbers!).

Efficiency

Rather than typing the same things over and over again, you can set up functions to complete tasks like completing a statistical test or creating a plot. By using in-text R code, you won't need to manually type p-values and statistics and change them each time you re-do your analysis (hurray!!). Once your code is set up, it is easy to run your paper again with new data, and generate new plots, p-values, and statistical output tables.

Since you can define your colours, fonts, and ggplot theme at the beginning of your document, you don't need to fiddle with formatting each plot. All plots will have the same formatting, and it is easy to change colours/styling for all plots at once.

No more manual formatting!

You don't need to worry about the layout of your document at all. You do not need to manually number your figures or subheadings, format your Table of Contents, or readjust paragraphs each time you insert or remove a picture. You can easily cross-reference figures, equations and chapters. To insert a list of figures, just type \listoffigures - see, it's very doable!

Since RMarkdown is a plain-text document, your computer will be able to handle large documents much more easily than Word. It will not get buggy or slow when you have lots of high-resolution pictures.

Beautiful typography

When you knit a document using a PDF, R uses $\LaTeX$. $\LaTeX$ is a typesetting engine that uses mathematical algorithms to arrange text into the most ideal way according to typesetting rules. The resulting documents look very elegant. $\LaTeX$-produced documents look very distinctive and after a while, you'll be able to easily tell the differences between a document produced using $\LaTeX$ vs. Word.

Even if you stick to just the defaults, your documents will look great. If you need to use equations, $\LaTeX$ is one of the best ways to create clean, well-aligned equations. Look in the $\LaTeX$ documentation to learn how to align by equal signs or other characters!

$$\int_a^b x^2\,dx \tag{1}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

Free

R, Rmarkdown, LaTeX and the packages you'll use are all free and open-source.

Longevity

Your documents are plain text files, which means that they don't take up much space on your computer, and you can open them up years later in any plain text editor (even Notepad!). It also means that your files will always be available, and you won't get locked into a specific version of proprietary software.

Version Control

It is easy to set up version control with R, Git and GitHub (which is also free). I would highly recommend learning how to set up a project on GitHub. Each time you make a change to the document on your computer, Git will also save the changes to the online version of your file, hosted on GitHub. It means that you'll always have a backup copy, and if something wrong happens, you can revert back to an older version and see what changes you made.

What are the disadvantages of writing in R?

Learning curve

If you're not familiar with R, RMarkdown, and LaTeX, it will take much longer to set up your paper than in Word. It will also take a while to format your preamble to make your document look exactly the way you want it. You'll likely spend a lot of time googling things and reading through answers on StackOverflow and GitHub.

Troubleshooting

Things will break down, and sometimes you'll spend way more time than you had anticipated dealing with error messages. During these times, it does seem much easier to just open a Word document and type there.

To minimize errors, I would suggest these things:

- Run each code chunk from top to bottom to catch any errors. Ensure that everything runs before knitting.
- Knit your document frequently as you go to help spot errors.
- R will knit documents using a blank R session, so everything that needs to be in the document must be defined in your script.
- Turn off the "Save workspace image" option in your R Global Options to prevent old loaded packages and hidden variables from 'hovering' in the background and creating strange rrors.
- Frequently use `Run` -> `Restart R and clear output` to prevent objects from cluttering your workspace and causing dependencies. For example, if you define a variable in the console, but not your document you won't realize the problem until you have a fresh R session.
- Use knitr::knit_exit() to stop knitting early. It can be a useful way to identify the specific line of code that is causing knitting issues.
- Make sure that your PDF is closed when you knit. If you're knitting the file and the PDF is still open somewhere on your computer, it will break the code.

Separate content and layout

You don't get to see what your document looks like until you knit it. This can be disconcerting for some people but great for others because you aren't distracted by formatting. Although you should knit your document often, don't fuss around with formatting until your paper is almost done. Things like paragraph spacing will change, and it is better to wait until the end.

At the beginning, you may worry about floats (things like pictures, plots, and tables). LaTeX will 'float' these over the text and then plop them down in a way that minimizes the number of paragraph breaks and blank spots. This means that your floats will often be further from where you want them. It is very, very difficult to 'force' floats to go into a specific spot, and the place that LaTeX chooses is often the best layout-wise. Always use references like Figure 1, rather than "the figure below".

Collaboration issues

Unlike Word, you can't use track changes, comments, or shared files. The best way to simulate this is to set up a GitHub repository and add your supervisor as a collaborator. They could then use pull requests to suggest changes. This may cause issues if you have a supervisor who doesn't know how to use GitHub. You also may not be able to make this work if your supervisor doesn't want to or know how to comment a PDF.

How-Tos

Inserting citations

Citation managers like Zotero (free; highly recommended) make it easy to link your citations in R. [Insert info here about the BetterBibTeX package]

[Insert info about setting up citation keys]

Now, you can type something like [@demeyts2000], and the citation will be rendered as an in-text citation.

> Insulin is a hormone that regulates blood glucose levels, digestive processes, and body weight [@demeyts2000].

This document uses APA formatting because of the 'apa.csl' file in the `Templates/` folder. If you want a different format, you can download the appropriate CSL file online and replace `apa.csl` with your new csl file name in the YAML header.

Inserting plots

You can plot figures in R. This is one of the best parts - R will generate the plots each time the document knits so you don't have to repeatedly copy and paste pictures into your paper!

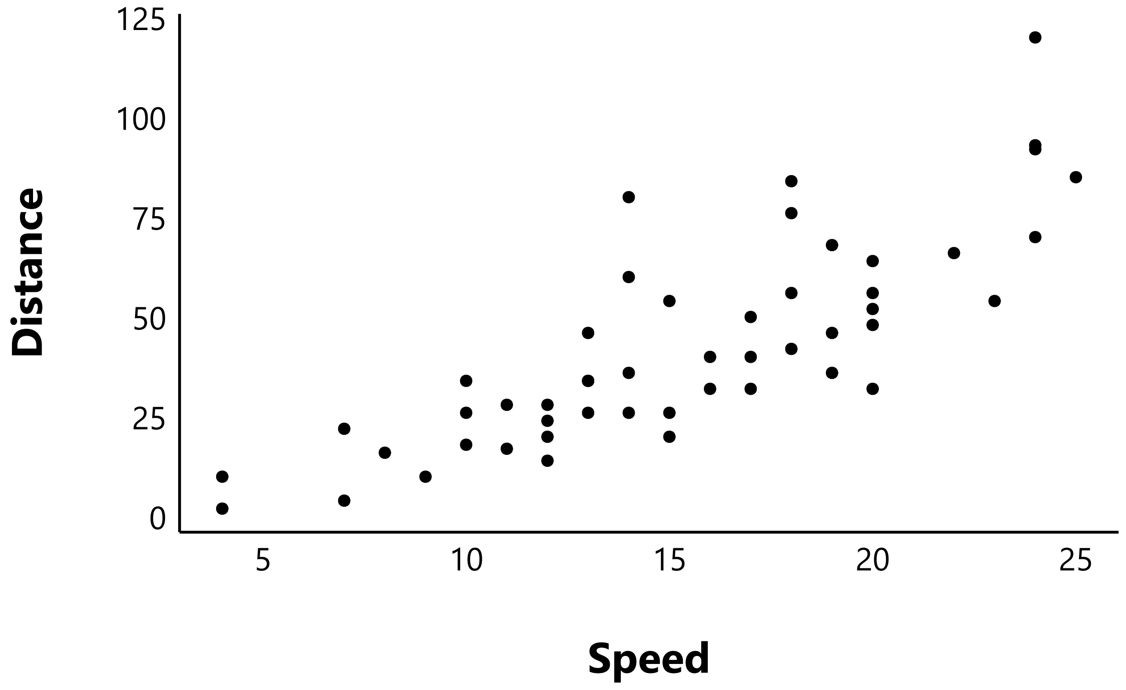


Figure 1: Braking distance is positively correlated with driving speed.

Inserting images

You can insert images that were generated elsewhere (.PNG, .JPG, etc.) through knitr. fig.cap can be quite long, and you can also add a label in the form of '\\label{text}' When you write a reference to Figure '\\label{insulin-pathway}', LaTeX will print out the text "Figure 2". If you reorder your figures, LaTeX will automatically change the figure numbers.

Important: To use `fig.cap` and `fig.scap`, you must define an `out.width` in the chunk options. If not, R will not recognize these as LaTeX commands.

Analysis

I'm using one of R's built-in datasets (*PlantGrowth*) to show examples of inserting model summary tables, results, plots, and p-values in a paper. This sample dataset explores plant mass after applying one of three treatment conditions (control, treatment 1 and treatment 2). I've hidden the code in the output document (chunk options `echo=F`) because you will probably not be displaying your code in your paper.

In-text R code

You can embed R code within normal text to create dynamic reports. For example, if you write ' r nrow(plant_data) ', this will be printed out as 30, as in:

> We weighed 30 plants.

Extracting P-values

You can take this even further to embed p-values, t-test statistics, and other values within the body text - no more copying and pasting! This will save you lots of time in the results section. It also reduces the likelihood of mistakes.

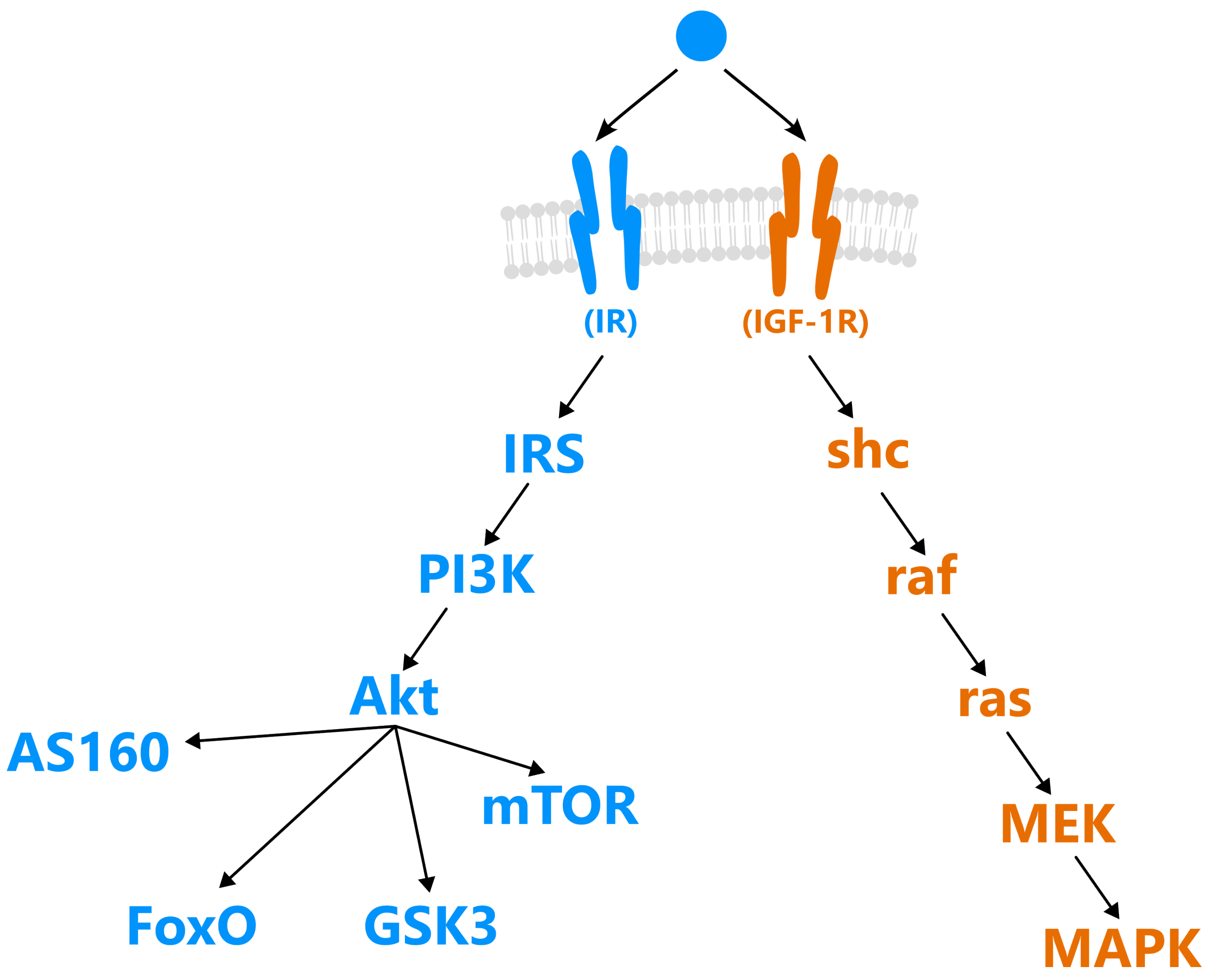
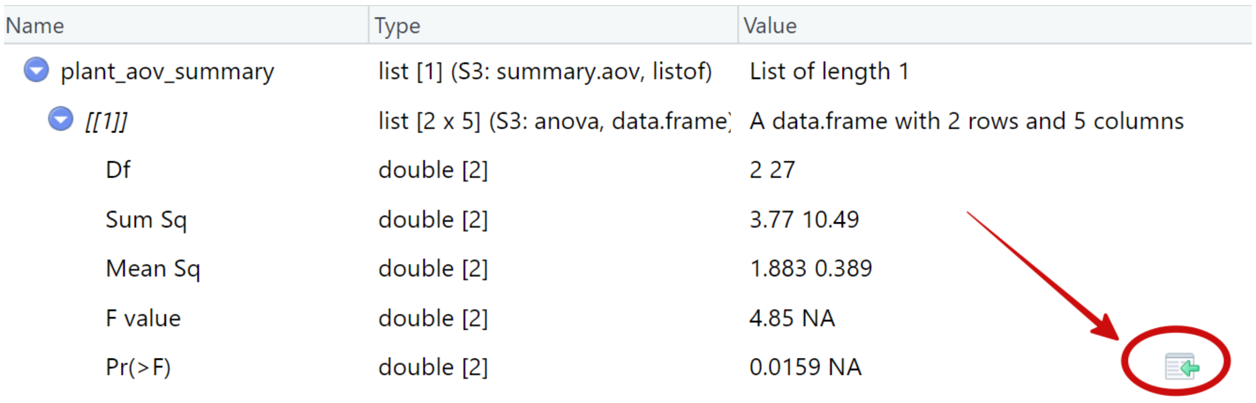


Figure 2: When insulin binds to a cell, it activates a series of molecular pathways involved in energy metabolism and gene expression…[if your figure caption is too long, you should define a short figure caption (fig.scap) which will go into the list of figures]…although it looks like there are two distinct pathways, the downstream components of the PI3K and MAPK pathways frequently interact with one another (De Meyts, 2000).

It will take some time to find out how to extract p-values from different data types. In RStudio, always check what class your model is. Then, you can google things like "Extract p-value from an object of class anova in r".

If you click on the model summary object in RStudio, it will open the model in a new window. You can click on any one of the green arrow icons on the right-hand side. The code required to select it will then appear in the console, where you can copy and paste it.



Name	Type	Value
plant_aov_summary	list [1] (S3: summary.aov, listof)	List of length 1
[[1]]	list [2 x 5] (S3: anova, data.frame)	A data.frame with 2 rows and 5 columns
Df	double [2]	2 27
Sum Sq	double [2]	3.77 10.49
Mean Sq	double [2]	1.883 0.389
F value	double [2]	4.85 NA
Pr(>F)	double [2]	0.0159 NA

Figure 3: RStudio provides a handy selection tool to help you find the code needed to extract a specific value from an object. Click on the green arrow icon to see the appropriate selection code appear in your console.

If you're stuck, you could also use the `str()` function to see the structure of the model. This can provide a list of the parts of the model and give you an idea of which variables you need to select the specific p-value that you want.

```r
str(plant_aov_summary)
```

```
List of 1
 $ :Classes 'anova' and 'data.frame':   2 obs. of  5 variables:
  ..$ Df     : num [1:2] 2 27
  ..$ Sum Sq : num [1:2] 3.77 10.49
  ..$ Mean Sq: num [1:2] 1.883 0.389
  ..$ F value: num [1:2] 4.85 NA
  ..$ Pr(>F) : num [1:2] 0.0159 NA
 - attr(*, "class")= chr [1:2] "summary.aov" "listof"
```

Look at the results. If I want to extract the p-value row, it looks like I will need to write plant_aov_summary[[1]][["Pr(>F)"]]. The [[1]] is because this is a list of one, and the values we need are in the first element of the list. Notice how this returns another list:

```
plant_aov_summary[[1]][["Pr(>F)"]]
```

```
[1] 0.01590996         NA
```

The first p-value is the p-value associated with the treatment groups. The second is NA because it is for the model residuals. To extract the first p-value, add another [[1]] to select the first element of this list. The final code is:

```
plant_aov_summary[[1]][["Pr(>F)"]][[1]]
```

```
[1] 0.01590996
```

Rounding

To make "prettier" p-values you could round them to 3 digits using the `round()` function. However, I would suggest using the `pvalString()` function from the `LazyWeave` package. This rounds p-values to three digits, and automatically formats them to publication style p-values if they are smaller or larger than typical endpoints. As an example, pvalString(0.00000005) will become $p < 0.001$.

```
pvalString(plant_aov_summary[[1]][["Pr(>F)"]][[1]])
```

```
[1] "0.016"
```

Now, putting it into an in-text R code block, we can write something like this with automatically inserted p-values.

> Plant mass varied significantly according to treatment type ($p = 0.016$).

Tables

There are many packages that will allow you to create publication-quality tables out of summary tables in R. This is good, because the default summary tables are not visually appealing:

```
summary(plant_data_aov)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
group        2  3.766  1.8832   4.846 0.0159 *
Residuals   27 10.492  0.3886
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The first step is to use the `tidy()` function from the `broom` library. This reformats the summary object to a tibble (a special type of dataframe) and does some automatic cleaning. For example, it removes the significance codes and converts the column names to code-friendly names with no spaces.

```
plant_data_aov %>%
  tidy()
```

```
# A tibble: 2 x 6
  term         df sumsq meansq statistic p.value
  <chr>     <dbl> <dbl>  <dbl>     <dbl>   <dbl>
1 group         2  3.77  1.88       4.85  0.0159
2 Residuals    27 10.5   0.389     NA     NA
```

The next step is to round the values and re-format the p-values. The `across()` function from `dplyr` allows us to apply the same rounding function to multiple columns simultaneously. We can also change the term names to title case using `str_to_title(term)`. Lastly, we can use the `rename()` function to give the columns more useful names for a publication.

```r
plant_data_aov %>%
  tidy() %>%
  mutate(
    across(sumsq:statistic,round, 2),
    p.value = pvalString(p.value),
    term = str_to_title(term)
  ) %>%
  rename(
    Term = term,
    'Sum of Squares' = sumsq,
    'Mean Squares' = meansq,
    'F-statistic' = statistic,
    'p-value' = p.value
  )
```

```
# A tibble: 2 x 6
  Term         df `Sum of Squares` `Mean Squares` `F-statistic` `p-value`
  <chr>     <dbl>            <dbl>          <dbl>         <dbl> <chr>
1 Group         2             3.77           1.88          4.85 0.016
2 Residuals    27            10.5            0.39         NA    <NA>
```

The last step is to use the `kable()` package to modify the table style and presentation. The documentation for the `kable` and `kableExtra` packages has lots of useful information on the available arguments. A particularly useful argument is `align`, which allows you to specify how the text is aligned in the columns. For example, `align = c('l', 'c', 'l')` will generate columns that are left-, centre- and left-aligned.

The new code section looks like this:

```
kable(
  booktabs = T,
  linesep = '',
  escape = F,
  caption = "Plant growth varies significantly with treatments. ANOVA summary table for
) %>%
  column_spec(1, width = "1.5in")
```

The final table looks publication-ready!

Table 1: Plant growth varies significantly with treatments. ANOVA summary table for a model examining the effect of treatment on plant mass.

Term	df	Sum of Squares	Mean Square	F	p-value
Group	2	3.76634	1.8831700	4.85	0.016
Residuals	27	10.49209	0.3885959	NA	NA

LaTeX Basics

You can learn a lot about LaTeX as you customize your paper. This is a very valuable skill, and it will be particularly useful if you're planning to stay in academia.

Most LaTeX commands start with a forward slash and include arguments in curly brackets. Many of them are intuitive:

Use `\\setcounter{tocdepth}{2}` to set the table of content depth to header 2

Use `\\tableofcontents` to insert a table of contents

If you want to make any cross references, you can define them using labels. You may have seen this in the *Inserting Images* section. For example, you can write Figure '\\label{insulin-pathway}' to automatically include the correct figure number.

It's not recommended to do too much formatting within your document. Ideally, all of your formatting styles should be defined in the preamble.

Changing the formatting

If you want to change how any part of this document looks, go to Templates/MtA-Thesis-Preamble.tex.

Other Tools

To create high-quality schematics, I would highly recommend Inkscape, which is a free and open source vector editor. You're probably familiar with using the shapes tools in PowerPoint. Inkscape is like this, but you have much more control over the alignment, and many helpful tools that allow you to do more advanced techniques.

References