CASA dissertation using Bookdown

Yifan Wu

CASA0010, MSc Smart Cities Dissertation

Supervisor: Dr Duncan Smith

Repository: https://andrewmaclachlan.github.io/CASA0005repo/

This dissertation is submitted in part requirement for the
MSc (Or MRes) in the Centre for Advanced Spatial Analysis,
Bartlett Faculty of the Built Environment, UCL

Word count: unknown 17:08 update

2025-06-28

Abstract

Some abstract text

Declaration

I hereby declare that this dissertation is all my own original work and that all sources have been acknowledged. It is xxx words in length

Acknowledgements

I would like to thank blah blah

Contents

List of Figures

List of Tables

Abbreviations

Term	**Abbreviation**
Digital Elevation Model	DEM
Digital Surface Model	DSM
Digital Terrain Model	DTM

Chapter 1

Bookdown basics

Bookdown enables us to take raw text files (e.g. Rmarkdown files) and output them into a number of different formats with ease. This is more useful than LaTex as you can create a word version for comments from your supervisor, a pdf for final submission and an online book for your own portfolio.

1.1 Structure

A bookdown book simply combines multiple Rmarkdown files into .pdf, html or .epub (but i've disabled epub here).

All Rmakdown files must be located in the base (or root) of the project. For example, don't got putting the .Rmd in a folder called chapters and then wonder why it's not working. They all must be in the same folder as the project file. You can however put output figure images into a folder (e.g. figures) then call them in.

1.2 Building the book

You will need the packages: `bookdown, kabble, knitr`

You also need to install `tinytex::install_tinytex()` for generating .pdfs from your book.

Once installed you can build the book by clicking the Build Book icon under the build tab in the top right quadrant.

To export the book to word follow the code below from Edd Berry:

```r
bookdown::preview_chapter("01-intro.Rmd",
                output_format = "bookdown::word_document2",
                output_file = paste0("thesis-intro-",
                                     format(Sys.Date(),
                                            ("%d-%m-%y")),
                                     ".docx"),
                output_dir = "chapter-previews")
```

You can also setup a reference document if you wish that word will style it on. To do so:

1. export to word using the code above.
2. change the headings styles in word
3. save the document
4. add the `output_options()` function to the code above

See Custom Word Templates for more detailed instructions.

```r
bookdown::preview_chapter("01-intro.Rmd",
                output_format = "bookdown::word_document2",
                output_file = paste0("thesis-intro-",
                                     format(Sys.Date(),
                                            ("%d-%m-%y")),
                                     ".docx"),
                output_dir = "chapter-previews",
                output_options = list(reference_docx = "word-style-ref.docx"))
```

You can also export the whole document as word by clicking Build Book and selecting it. However, note that tables built with `Kabble` don't work with word, so you have the following options

- create the html (gitbook) and copy the tables into word
- either exclude them (`eval=FALSE`) in the code chunk header
- use .pdf.

1.3 YAMLs

You will see some files with a `.yml` extension. These stand for yet another markup language.

Open the `_output.yml` and `_bookdown.yml`.

The `_output.yml` controls the settings for each outputted format.

The `_bookdown.yml` controls the order in which the files are made into chapters - you can also change the `Chapter` title if you wish here.

I have set these up to create nice outputs, so there is really no need to change anything, unless you want to add a new chapter and include it in the list. To do so, simply create a new Rmarkdown document, delete all the default content, add a first level heading, and the then add it to where you want it to appear in the document in the `_bookdown.yml`.

1.4 Formatting

You'll notice in the `_output.yml` that the various formats (html (or gitbook) and pdf) are calling styling codes. The gitbook calls `style.css` and the `pdf_book` calls `preamble.tex`. These files just style the various outputs.

I have copied in a basic styling with a CASA logo in the table of contents, you can replace this with other variations of the file if you wish or look at the minimal bookdown example or my CASA0005 .css file.

The best place to learn more about styling is the RStudio for education bookdown guide

For the .pdf version in the `preamble.tex` we have to use LaTex code. I am by no means an expert in this. If you Google each package it will tell you what it does. There isn't much here really, but all the header stuff refers to the headings at the top of each page.

This tutorial from OurCoding Club explains some of the LaTex packages in a bit more detail: https://ourcodingclub.github.io/tutorials/rmarkdown-dissertation/

1.5 Index file

Open up the `index.Rmd` this `Rmd` must be called first in the `_bookdown.yml` list. There are a number of options here, but i've set them all up to be compliant with the CASA thesis requirements.

You will need to change your title and name etc. Make sure you leave the a space here: `| Andrew MacLachlan` this won't work `|Andrew MacLachlan`

If you were to be printing your work, you'd want to change the `classoption` to `twosides` and make sure the `geometry margins` were correct. `linestrech` refers to the line spacing and the bibliography stuff we come on to later.

Input your GitHub repo and add a description.

If you ever wanted to just create a report and not a thesis you can change the `documentclass` to other options such `report`, `article` or `letter`

1.6 Preamble

Open the `preable.Rmd` and you will see all the sections that are required before the main text (e.g. Declaration and so on). At the top of the page i've used a code chunk set to LaTex, saying to use Roman numbering as we don't want page 1 to be the Declaration, we want it to be the first page of the Introduction. There are two conditions for each of the sections that state if output to HTML (gitbook) then do this, if output to LaTex then do this. This is the only place we have this. In our bookdown HTML we want to be able to click these sections, but in our LaTex .pdf we don't want them to appear in the table of contents. This is what this code is doing.

The Abstract is on the `index.Rmd` the same code condition applies to it, with Roman numbering also specified.

If you look back in the `_output.yml` you'll also see the `toc` (table of contents) is set to false. By default this appears right after the title, but we want this to come after our `preable.Rmd`. You'll see that i've called `\tableofcontents`, `\listoffigures` and `\listoftables` in the correct place again using a LaTex code chunk. These aren't required for the bookdown output.

The last section here is the abbreviations. To make this really easy, i've created an excel document to add them into. The code here will load that and then use the `kable` package to make a table. More on this later.I've also done the same for the research log in the appendix — excel sheet called `research_log`.

1.7 Change this to a thesis

Easy. Just change all the titles to what you want (e.g. Introduction, Literature Review, Methodology, Discussion, Conclusion). Some of the latter `.Rmds` (Discussion etc) are ready to go!

1.8 Word count

To get a word count install and then use the word count addin package through Tools>Addins>Wordcount

1.9 Adding a pdf

If you wish to add another .pdf as an Appendix (in your .pdf) then again we need a bit of LaTex code

`\includepdf[pages={-}]{mypdf.pdf}`

If you look in the `08-Appendix.Rmd` then you will see another if LaTex section, simply add in the line of code above, replacing `mypdf.pdf` with your pdf title in the main project folder. It will then be appended to the thesis. Of course, this isn't required in the online book, but you just link to them on GitHub or embed .pdfs using:

You'd need a condition around this like in the `preamble.Rmd` but a link is fine. I embedded a .pdf in the assignment resources of CASA0005

1.10 Writing code

Use one project for your thesis and another for your analysis. Don't try and do it all in a thesis project. You can set your output folder from your main analysis project to the thesis project and then easily load the figures in.

1.11 Package reproducibility

Have you ever created an R script, come back to it 6 months later and wonder why it's not working correctly? It's probably because of package updates.

`renv` (pronounced R - env) can capture the packages used in your project and re-create your current library. You simply:

1. Create a new project - `renv::init()`
2. Create a snapshot - `renv::snapshot()`
3. Call the snapshot to load - `renv::restore()`

The package information and dependencies are stored in a `renv.lock` file.

When R loads a package it gets it from the library path, which is where the packages live. Sometimes there are two libraries a system and a user library - use `.libPaths()`. The system library = the packages with R, the user library = packages you have installed.

When you load a package it loads the first instance it comes across, user comes before system. To check - `find.package("tidyverse")`

All your projects use these paths! If you load different packages and versions of them + dependencies. E.g.

- Project 1 used `sf` version 0.9-8
- Project 2 used `sf` version 0.9-6

Switching between projects would mean you have the wrong version as they use the same libraries.

`renv` - each project gets it's own library! Project local libraries.

When you use `renv::init()` the library path will be changed to a project local one.

It will create a lock file that holds all the package information.

To re-create my environment once you have forked and pulled this repository you would use `renv::restore()`.

Of coruse some projects use the same package version — such as `tidyverse`, `renv` has a global cache of all the libraries. So there is a massive database of your libraries then each project library links it from there, meaning you don't have 10 versions of the same `tidyverse`.

You can also interact with `git`

- `renv::history()` — finds the commits where the lock file changed
- `renv::revert(commit = "id")` — changes the lock file back to what it was at a commit

For more information watch renv: Project Environments for R introduction video: https://www.rstudio.com/resources/rstudioconf-2020/renv-project-environments-for-r/

Chapter 2

Cross referencing

2.1 Literature Review

Understanding the relationship between urban cycling accessibility and infrastructure quality has been a prominent focus in spatial and transport research over the past decade. Numerous studies have emphasized the role of dedicated cycle lanes, traffic-calmed streets, and connected networks in promoting bicycle usage in cities (Pucher & Buehler, 2008; Hull & O'Holleran, 2014).

The concept of Low-Stress Network (LTS) classification proposed by Mekuria et al. (2012) has been widely adopted as a framework to assess the comfort and safety of cycling environments. It categorizes street segments based on traffic speed, volume, and intersection complexity, providing a scalable metric for infrastructure evaluation. Recent works (Conveyal, 2020; Broach, 2021) further operationalize LTS methods using OpenStreetMap and GTFS data, enabling large-scale, reproducible analyses.

Environmental perceptions also significantly influence cycling behaviour. Studies have shown that air quality (de Nazelle et al., 2011), green view index (Zhang et al., 2022), and even soundscape (Aletta et al., 2019) can affect the attractiveness of cycling routes. Integrating such perceptual dimensions into infrastructure analysis has become a growing trend, leading to the development of composite indicators like the Cycling Environment Comfort Index (CECI), which attempt to synthesize structural and perceptual factors into a unified framework.

However, there remain several gaps in the literature. First, most accessibility-based studies focus on either spatial network topology or perceived comfort but rarely combine both systematically. Second, few empirical studies have validated LTS-based classifications using real-world cycling behaviour data, such as GPS trajectories or app-based route choices. Third, while multi-criteria indicators like CECI are gaining popularity, their construction methods vary greatly across studies, lacking consensus on factor weighting or standardization.

This dissertation aims to address these gaps by proposing a hybrid evaluation framework that integrates network-level accessibility measures with environmental perception indices. The review above forms the conceptual basis for this integration, drawing from methodological insights and empirical findings in existing research.

Test! Harvey et al. (2024) systematically compared seven LTS classification methods across different data sources and found considerable variation, questioning the assumption of commensurability (**harvey_comparing_2024**).

Chapter 3

Equations and direct quotes

This section will focus on equations and direct quotes

3.1 Equations

You need to include equations with some LaTeX. The easiest way to do this is to use an online tool such as: https://latex.codecogs.com/eqneditor/editor.php. It can be a real pain to get these right, but once you've worked it out it will be much easier than dealing with word equation editor.

$$p = h\frac{c}{\varrho} \tag{3.1}$$

To reference this in the text we use: (**??**)

You can also test your code in your RMarkdown document using `$` e.g.

$$p = h\frac{c}{\varrho}$$

However, this won't generate an equation number and you can't cross reference it. But we can use this logic to define the parameters within the equation e.g. where

h is Plank's constant, 6.626×10^-34Js

`$$` puts it on a new line, single `$` keeps it on the same line (in the text)

3.2 Block quotes (or direct quotes)

You may wish to quote a large section from a source, to do this use a block quote.

Simply input a `>` before the text. For example `> This is a quote`.

> "This is a direct quote"

You can also provide an attribution at the footer of the quote using `tufte::quote_footer()`, either a name or a reference. You'll need to install the `tufte` package to use this. For example, `> [include only r here] tufte::quote_footer('--- Joe Martin')` or `> [include only r here] tufte::quote_footer('--- @xie2015')`

Giving...

— Joe Martin

— **xie2015<empty citation>**

Instead you could just include the reference at the end of the quote, using the same method to reference as we've seen before..e.g. `> "This is a direct quote" @xie2015, Equation \@ref(eq:test)`...giving

> "This is a direct quote" **xie2015<empty citation>** (**??**)

Chapter 4

Figures, tables, hosting GitBook

This section is going to focus including figures and creating tables

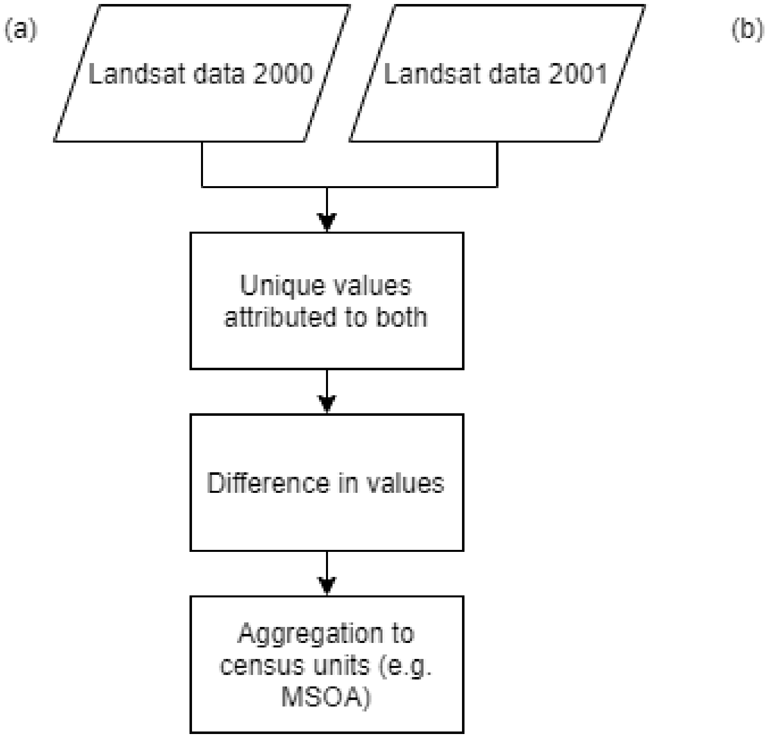
(a)
Landsat data 2000
Landsat data 2001
(b)
Unique values attributed to both
Difference in values
Aggregation to census units (e.g. MSOA)

Figure 4.1: Summary of methdological procedure for (a).... and (b)....

4.1 Including figures and tables

4.1.1 Figures

To include the figure above use the code:

```
knitr::include_graphics(here::here('general_images','example_flow.png'))
```

within a code chunk. In the chunk options you can specify the width and figure captions e.g.

`out.width="100pt", fig.cap="Summary of methdological procedure for (a).... and (b)...."`.

However, if you do show the code with `echo=TRUE` then you can't specify the `out.width`.

For making flow diagrams have a look at:

1. Lucidchart
2. Draw.io

4.1.2 Tables

For creating tables i'd suggest creating either an excel file or `.csv` and then reading the data into R and using the `kabble` package to format it how you wish. The example below is from the abbreviations section

```r
library(tidyverse)
library(knitr)
library(kableExtra)
library(readxl)
library(fs)
library(here)


#read in data
```

Term	Abbreviation
Digital Elevation Model	DEM
Digital Surface Model	DSM
Digital Terrain Model	DTM

```
read_excel(here("tables", "abbreviations.xlsx"))%>%
  arrange(Term) %>% # i.e. alphabetical order by Term
  # booktab = T gives us a pretty APA-ish table
  knitr::kable(booktabs = TRUE)%>%
  kable_styling(position = "center")%>%
  # any specifc row changes you want
    row_spec(.,
  row=0,
  bold = TRUE)
```

You do loads of things with kabble including adding small visulisations within the table - consult the documentation for more info.

If in doubt, keep it simple

other useful arguments for tables:

- `column_spec(2, width = "9cm")` = set column width
- `kable(timeline,longtable = T....`= allow the table to go over multiple pages

For example…