ClavertonDown Example

Emma Cliffe, Skills Centre: MASH, University of Bath

August 2020

Contents

Introduction

ClavertonDown is aimed at staff at the University of Bath which is located on Claverton Down. It uses knitr, RMarkdown and Bookdown and aims to produce a system for creating sets of lecture notes in a variety of formats, including those which are technically accessible and those which are often preferred by disabled students. We aim for the system to be usable in both pure and applied mathematical settings. Requests to adapt the system or the outputs of the system to meet these goals will be considered by MASH. The main contact email address is mash-access@bath.ac.uk but you should use the Team set up by the Centre for Learning and Teaching for support in the first instance.

This document is **also** a work in progress. Like everything right now!

1 What formats do I need to produce and why?

ClavertonDown can produce:

- HTML book
- HTML page
- Standard, clear and large print PDF
- Word document
- EPub book

From a technical point of view only three of these formats of mathematical text are accessible:

- HTML using MathJax to render all mathematical text (HTML book and page)
  - HTML formats must meet WCAG 2.1 Level AA requirements and use MathJax to render all mathematical text.
  - You can check your document meets the legal requirement of WCAG 2.1 Level AA with e.g. Accessibility Insights for Web plugin for Chrome
  - You can check it is MathJax by right clicking on the expression (try it in the

HTML version):

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

- Word document using in-built Equation Editor for all mathematical text
  - Word formats must pass the in-built Accessibility Checker and use the in-built Equation Editor for all mathematical text.
  - You can use the in-built Word Accessibility Checker and information on Making your Word documents accessible to check your document.
  - You can use Review -> Read Aloud (Alt + Ctrl + Space) to check the maths
- EPub3 using MathML for all mathematical text:
  - If the HTML format is accessible then the EPub3 output from ClavertonDown will be also

The output from ClavertonDown will meet these requirements as far as possible. However, we cannot stop you from doing things which will fail to meet requirements. In particular, if you use images, sound, videos, colour or interactive/dynamic elements within the HTML you will need to ensure that you have done so within the accessibility requirements.

1.1 Why do we still supply inaccessible PDFs?

It isn't possible to create accessible PDFs using LaTeX. Even if a PDF is accessible the mathematical text within it will **still be technically inaccessible** regardless of how the PDF has been produced but...

- not all accessibility is about technical access - for some a clear or large print PDF is best
- currently, clear print is selected most often by disabled students in the Department of Mathematical Sciences
- PDF remains the best fallback when a student is concerned that a document is not being rendered as the author intended
- PDF remains the best starting point for creating hard copy resources which is important for some disabled students

Finally, our aim here is to produce choice for **all** students, disabled or not, and many will still select PDF.

1.2 Do I really have to put all this on Moodle?

If you use the same set up as this example document then all the outputs will be collected together into a single folder. We recommend that you:

- Zip up the folder and upload it to Moodle **as a file**
- Select Add a resource -> File. Give the resource a name (e.g. lecture notes 1).
- Add the zip file. Right click on the zip file, a window appears, click on unzip and wait until complete.
- Right click on the zip again and click on delete. You will have a folder remaining.
- Left click on the folder to open it.
- Find the main file index.html and right click on this, a window appears, click on Set main file.
- Change any other settings you wish in the resource and then save.

When you click on this resource you will see the HTML book version of the notes. This is technically accessible, searchable and usable on a mobile device. In the menu at the top there is a download button, from here a student can access any of the other formats. This is as close as we can get to replicating the Blackboard Ally student experience of being able to transform accessible resources to other formats.

2 Theorems and stuff

You will notice that there are a variety of ways that theorem type environments are presented in the different formats. All of these pass technical accessibility tests or are following RNIB Clear Print guidelines and have been used for at least a decade in our provision for disabled students. We have also take on feedback that in HTML formats colour might be helpful to show where environments start and finish. We have chosen to impose a structured colour scheme on you, for now. The colours have been chosen so that they pass accessibility tests for contrast with the text.

However, this doesn't mean we have got it right. Give us feedback. Ideally, get students to give feedback too.

Theorem 2.1 (Foo)**.**
This is a theorem environment already provided by Bookdown. It still works as before.

Proposition 2.2 (Thingy we need for 2.1)**.**
However, in this system you can now

- Change the numbering system of the inbuilt environments - this is done in the file _bookdown.yml. Please see the file in the same folder as this file.
- Reference other environments within names. Notice that you need to use a double backslash.

Proof (Of theorem 2.1)**.**
You can make new unnumbered theorem environments. You can call them whatever you like. They work kind of like the inbuilt ones but the environment is always newtheorem and the env defines the type.

You need to have predefined the type and made any changes to the standard print style in the file _bookdown.yml. Please see the file in the same folder as this file.

You can't control what things look like in the other formats as they have been designed with a variety of accessibility features. If you don't like how they look then maybe contact us and ask about it. It might be something we can think about changing or allowing author control or, it might not. □

Proof of theorem 2.1. Defining your own proof environment doesn't stop the inbuilt one working. □

Definition 2.1.
You can leave some things numbered on their own.

Definitions 2.2.
You can create new numbered theorem types. Unlike inbuilt environments, for them to be numbered in all formats they **must** have a label. If you forget then they won't be numbered in any format **except** for PDF and then your numbering won't match. I might try and fix this at some point. You can number them alone or with other inbuilt or newtheorem environments.

You need to have predefined the type, the numbering and made any changes to the standard print style in the file _bookdown.yml. Please see the file in the same folder as this file.

Take care not to reuse the inbuilt numbering labels with your new theorems! This will lead to odd things happening. The inbuilt labels to avoid are: thm, lem, cor, prp, cnj, def, exm, exr, fig, tab and eq.

References still work in the same way as in Bookdown. Now go to theorem 2.1 or proposition 2.2.

Here is some text which is not part of the below example.

Examples.
You can turn off the colour and padding in html, ePub and Word for any newtheorem or inbuilt theorem type. You do this in the _bookdown.yml file by adding the theorem name to the colouroff style_with list.

Here is some text which is not part of the above example.

2.1 But I want to number some of them...

Definitions.
This is unnumbered in all formats

Definitions 2.3.

This should be definitions 2.3 in all formats

2.1.1 Out of interest. . .

What happens if I try to number an environment I have specifically declared to be unnumbered? e.g. using {newtheorem, env=‘Examples’, label=“argh”}

This will fail to compile for PDF and it will produce “Examples (#Examples:argh)” as the title in other formats.

The only real benefit to be gained from specifically declaring an environment to be unnumbered is that you are less likely to accidentally number an instance of it!

2.2 Colour has meaning

The coloured styles for theorem-type environments in HTML, EPub and Word are an attempt to colour code based very broadly on the mathematical type of enviroment. This is to respond to feedback that clear visual markers of the start and end of theorems was important. For accessibility purposes the use of colour can also be used to facilitate cognitive processing and this is stronger if the choice of colour has meaning.

LaTeX styling, while originally intended to broadly correspond to semantic type is, in our decade or so of experience, not used this way by authors of lecture notes due to the fonts and styling which are imposed in standard LaTeX. So, the three styles in LaTeX:

- Plain
- Definition
- Remark

are used to impose the authors visual preferences mostly with respect to font, font attributes and layout. That is, they are usually syntactic choices, not semantic ones.

Hence, our assignment is NOT based, at all, on the PDF styling specified by the author, or any other information like e.g. numberwith.

Since the user can define any number of new theorem type environments we have to. . .

guess… to try and align colours to broad types. We have four HTML styles:

- ProofStyle: (P|p)roof(s), (S|s)olution(s), (S|s)oln(s), (R|r)emark(s) and anything including these words; if not then one of the below
- ExampleStyle: (E|e)xample(s), (E|e)xercise(s) and anything including these words; if not then one of the below
- DefinitionStyle: (D|d)efinition(s), (D|d)efn(s) and anything including these words; if not then defaults to the below
- TheoremStyle: Everything else - but should definitely include (T|t)heorem(s), (L|l)emma(s), (C|c)orollar(y|ies), (P|p)roposition(s), (C|c)onjecture(s).

You might find this automatic sorting to be wrong or annoying. Also, we can't possibly guess what you might call things so lots of types which aren't theorem-like will end up with the default theorem colouring which isn't helpful.

This is an accessibility issue - the use of colour to highlight meaning helps with processing so we need to give you control to an extent.

2.2.1 So…?

We have given the author direct control over which class of HTML to place each theorem type in - both in built and new theorems. You can override our guesses. We have also given you control to turn off italics in the book HTML (the other accessible formats have this turned off already).

- To turn off italics change style_with -> italicsoff to TRUE in the _bookdown.yml file (or add this to a file you already have). You don't need to specify this, if you don't, for backwards compatibility, it will default to italicsoff is FALSE
- To classify theorem types you can add, if you wish, a classify_as section to your _bookdown.yml. Again, you don't need to add this if you don't want to change anything. You can see an example of this in the _bookdown.yml file for this document in which the new theorem type Thought is classified as an example. If you remove this classification it will default to the theorem type and the below will change colour.

Thought.

Here is a thought.

3 Figures inside other environments

3.1 Here is a figure

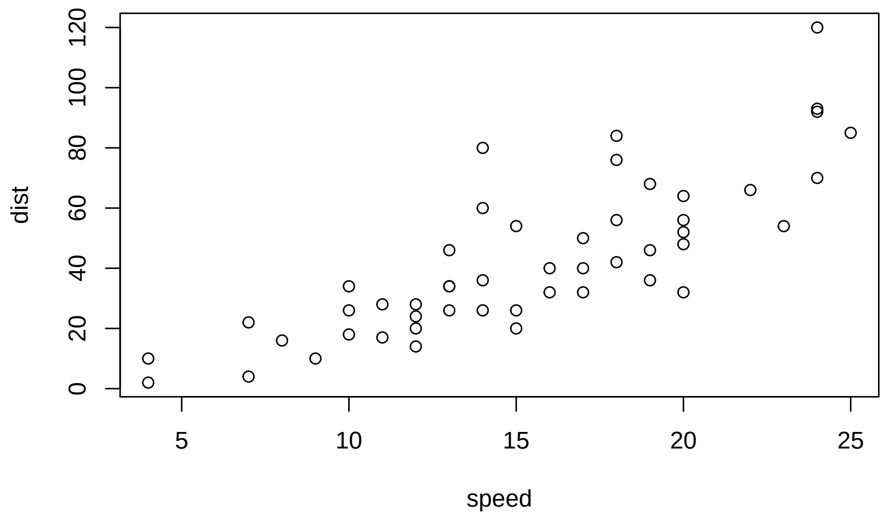

Figure 1: This is title and a caption with a reference 2.1 inside it

3.2 Here is the putting of a figure inside another built in environment

Example 3.1.

Here is an example.

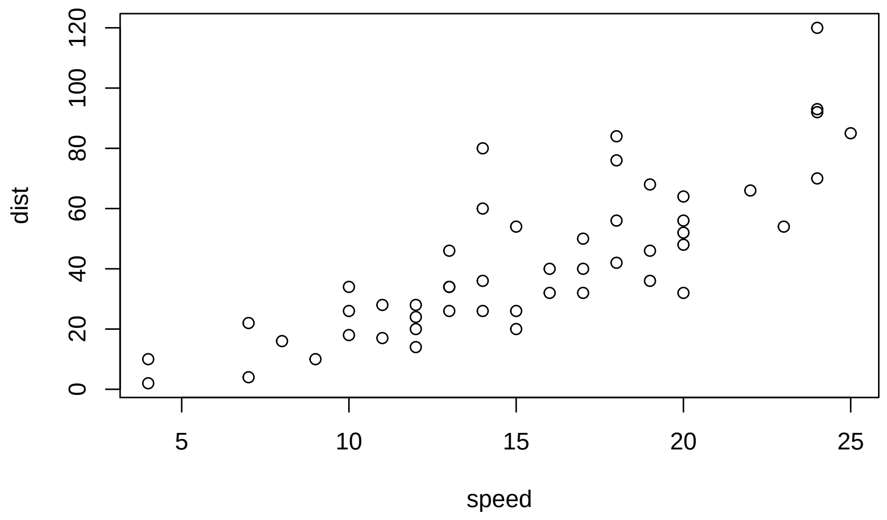

Figure 2: Something to do with cars

This is a test. So, you need an empty line before and after the above for it to be a float. At the end of an environment this means that you need TWO empty lines. This is Pandoc.

3.3 Here is the putting of a figure inside a newtheorem

Here we are going to do something a bit more impressive with the automatic creation of alternative image formats. This is only useful to you if you are using R to generate graphs etc.

Example.
An example in which we have autogenerated a useful description of the scatterplot for a person who cannot see it, using BrailleR. To see the content of the long description if you view the HTML in Firefox and right click you can request it.

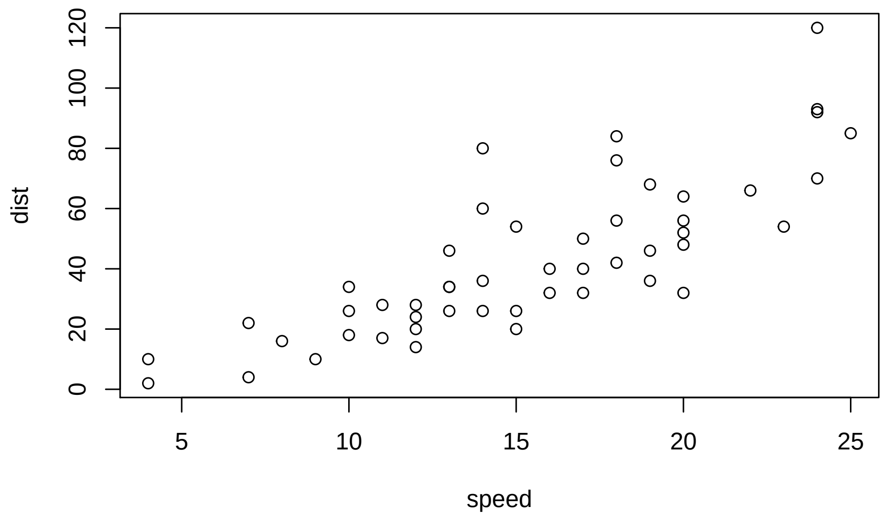

Figure 3: More cars things

4 Repeating environments

Sometimes I would like to repeat a definition from earlier and have it numbered the same. This now works as expected in all formats including the numbering of the environment. However, the numbering of **things inside the environment** will probably be wrong so take care. Tell us if this is important to you as it is low priority at the moment.

Inbuilt using own numbering

Theorem 2.1 (Foo)**.**
This is a theorem environment already provided by Bookdown. It still works as before.

Inbuilt using other inbuilt numbering

Proposition 2.2 (Thingy we need for 2.1)**.**
However, in this system you can now

- Change the numbering system of the inbuilt environments - this is done in the file _bookdown.yml. Please see the file in the same folder as this file.
- Reference other environments within names. Notice that you need to use a dou-

ble backslash.

Newtheorem using other inbuilt numbering

Definitions 2.2.
You can create new numbered theorem types. Unlike inbuilt environments, for them to be numbered in all formats they **must** have a label. If you forget then they won't be numbered in any format **except** for PDF and then your numbering won't match. I might try and fix this at some point. You can number them alone or with other inbuilt or newtheorem environments.

You need to have predefined the type, the numbering and made any changes to the standard print style in the file _bookdown.yml. Please see the file in the same folder as this file.

Take care not to reuse the inbuilt numbering labels with your new theorems! This will lead to odd things happening. The inbuilt labels to avoid are: thm, lem, cor, prp, cnj, def, exm, exr, fig, tab and eq.

I still need to test the other combinations. Please tell me if it is broken.

5 But... I want it to do...

We have chosen to create a new package instead of adding to Bookdown because we are diverging from what Bookdown was designed to do. This also means that we can mould how this system works to best meet the needs of students and lecturers.

Tell us what to do next. We'll tell you if it is possible and the likely priority!

6 LaTeX newcommands, usepackages etc.

You can add these in the header-includes part of the code at the start of this file. You cannot pass a filename to this at the moment. If this is a dealbreaker then let me know.

This will work in all formats. **All** other ways of adding commands and packages in will not result in the right thing happening in formats other than PDF, as far as I know.

$$BOO(x)$$

7 Great. How do I use this thing?

This package is not on CRAN and it is not going to be in the short or medium term. Instead you need to install it directly from github (if you want to know why then get in touch and ask) which means you will need devtools installed and loaded.

7.1 Installing ClavertonDown

Once you have devtools you need to load it:

```
library(devtools)
```

Then you can install ClavertonDown:

```
install_github(“BathMASH/ClavertonDown”)
```

Obviously, you need an internet connection.

7.2 Using ClavertonDown for the first time

I suggest that you download the files which were used to create this document:

- https://raw.githubusercontent.com/BathMASH/clavertondown/master/example/index.Rmd
- https://raw.githubusercontent.com/BathMASH/clavertondown/master/example/_bookdown.yml

Put them in a folder together, open index.Rmd in RStudio and use the Knit menu to compile the various formats. If this doesn’t work then something is wrong (get in touch Bath people!). The PDF command creates standard print, clear print and large print PDF.

Then… read the files.

7.2.1 Using the command line

If you prefer to compile from the system command line or via a script then commands along the lines of the below will work. You need to run all five to complete the full set of transformations in the example preamble.

```
Rscript -e “bookdown::render_book(‘index.Rmd’,‘clavertondown::gitbook_clav’)”

Rscript -e “bookdown::render_book(‘index.Rmd’,‘clavertondown::html_clav’)”

Rscript -e “bookdown::render_book(‘index.Rmd’,‘clavertondown::pdf_clav’)”

Rscript -e “bookdown::render_book(‘index.Rmd’,‘clavertondown::epub_clav’)”

Rscript -e “bookdown::render_book(‘index.Rmd’,‘clavertondown::word_clav’)”
```

The pdf_clav command renders standard print, clear print and large print PDF.

7.3 Moving from Bookdown to ClavertonDown

If you have been using Bookdown you may find that simply making changes to your preamble, as per the files above, will result in everything working as expected. You can then start using additional ClavertonDown functionality. If you do not use any of it then your content will, at the time of writing, still work in Bookdown if you change your preamble back. However, if you do not wish to use the additional functionality you should consider whether moving to ClavertonDown is the right choice for you - we are diverging from Bookdown and we cannot guarantee ongoing backwards compatibility. You should also consider that Bookdown is changing in ways that ClavertonDown will not and these are explained in the below section.

7.3.1 Divergence: why simple preamble changes may not have worked

If you have written your Bookdown more recently and are using the fenced div environments rather than the knitr engine environments then simple changes to your preamble will not move your theorem type statements to the new format. They will remain in Bookdown format. We won't be 'fixing' this because:

- Bookdown documents written using this method can only be converted to PDF and HTML formats. ClavertonDown exists to enable to conversion to a full range of accessible formats to enable any student to access lecture materials. PDF documents containing mathematical expressions can never be technically accessible and HTML format is only technically accessible for some assistive technology users. Whether you generally supply Word and EPub or not you need to be able to create it on request.
- The work we have done to enable extra functionality cannot be delivered via the fenced div approach at this time. It would require all of the functionality to be reimplemented at a completely different point in the conversion and, based on our current understanding, some of the functionality, which was requested by lecturers and students, could not be delivered at this time if at all.
- While we accept the reasoning for the change in Bookdown - that it enables richer content to be included within theorem type environments and has a clearer implementation - we have not had any request to enable the types of richer functionality meant here. Other 'missing' things were considered more of a problem for lecture materials. If future requests fall into these categories we will look at whether we can implement them while retaining the current functionality. It is also noted that our implementation of the knitr custom blocks has already diverged from that in Bookdown. While it is not correct to call it cleaner it should be more robust than the original custom block engines.

If you have a Bookdown document which uses fenced divs you should consider how important it is to you to move to ClavertonDown. We have supplied a function which you can run to create a copy of your document and attempt to change format - as Bookdown has created a function which goes back the other way. Our function is going in the more challenging direction and so it aims to only make the change when it is 'safe' to do so and maintains a complete original copy untouched without the user needing to ensure this. If when you run the function there are warnings you will need to do some work by hand to complete the change. If you are at Claverton Down then we do ask you to get in touch with us and tell us if you are then missing functionality and we will do our best to assist - or, at least, to be clear about what your options are. It is possible to transform back the other way (this is the easier and safer direction) but

- We will have retained a copy of your original files for you so you don't need to worry about this if you immediately change your mind!
- We assume you are transforming for a reason - to use the functionality of ClavertonDown. In this case any transform back will require you to undo any use of our extra functionality.

7.3.2 I want to transform anyway

If, having read the above, you wish to try it out then you need to change directory to that containing your document run the following on the R command line:

```
library("clavertondown")

intercept_theorems("./")
```

Please read the output carefully.

If you receive **any warning** then you have

- nested theorem type fenced div environments or
- knitr engine blocks (including but not limited to theorem type) nested in fenced div theorem type environments or
- You have used more than three colons when defining a fenced div despite it not being nested - just set it back to three in this case.

Nesting of this kind cannot be converted - theorem type environments cannot be nested in Clavertondown and knitr engine blocks cannot be nested within theorems. You will need to undo the nesting in whatever way makes sense for your content. This type of nesting is a primary example what Bookdown means by 'richer content'.

8 Can I tell people not at Claverton Down about this?

Yes, sure, that is why it is public. But, this is released with no promises and no warranty. Help will only be provided to staff and students of the University of Bath but we might fix bugs or consider suggestions by others.