Bookdown: Flexible Document Creation in RStudio

Joseph Thiers

Contents

Chapter 1

Introduction to Bookdown

In today's fast-paced academic and professional environments, the ability to create dynamic, reproducible documents is crucial. Bookdown empowers users to combine text, code, and visualizations in a single, streamlined workflow. It is ideal for creating the type of documentation that best fits your needs, whether that be single-page assignments, reports, academic papers, or even full-length books.

1.1 Why RStudio and Bookdown?

Bookdown offers a range of benefits:

- Seamlessly integrates text, code, and figures.
- Supports multiple output formats (HTML, PDF, EPUB).
- Simplifies the creation of reproducible and professional documents.
- Ideal for mathematics, statistics, and data science professionals.

1.2 Robust Documentation and Support

- Bookdown Documentation https://bookdown.org/yihui/bookdown/
- R Markdown Cookbook https://bookdown.org/yihui/rmarkdown-cookbook/
- Pandoc https://pandoc.org/
- RStudio Bookdown Community https://community.rstudio.com/tags/bookdown

- Gallery of Books you can Download https://bookdown.org/home/archive/

1.3 What You'll Learn in This Tutorial

This tutorial will guide you through the essential aspects of using Bookdown:

- **Chapter 1**: Introduction to Bookdown – Learn about its purpose and benefits for structured documentation.
- **Chapter 2**: Getting Started – Install R, RStudio, and Bookdown, create a project, and render your first book.
- **Chapter 3**: Writing Content – Organize chapters, use Markdown, format text, add code chunks, and images.
- **Chapter 4**: Cross-Referencing – Reference sections, figures, tables, and equations effectively.
- **Chapter 5**: LaTeX – Add equations, theorems, lemmas, and proofs.
- **Chapter 6**: Advanced Features – Manage citations, use LaTeX packages, and more!
- **Chapter 7**: Customizing Output – Configure formats like HTML, PDF, and EPUB, and style your book with CSS or LaTeX.
- **Chapter 8**: LaTeX Distributions - Different distributions available.
- **Chapter 9**: Advanced Text Formatting Options - Advanced Markdown and Pandoc code to stylize the book to your needs.
- **Chapter 10**: Example Document: Union Earnings Analysis

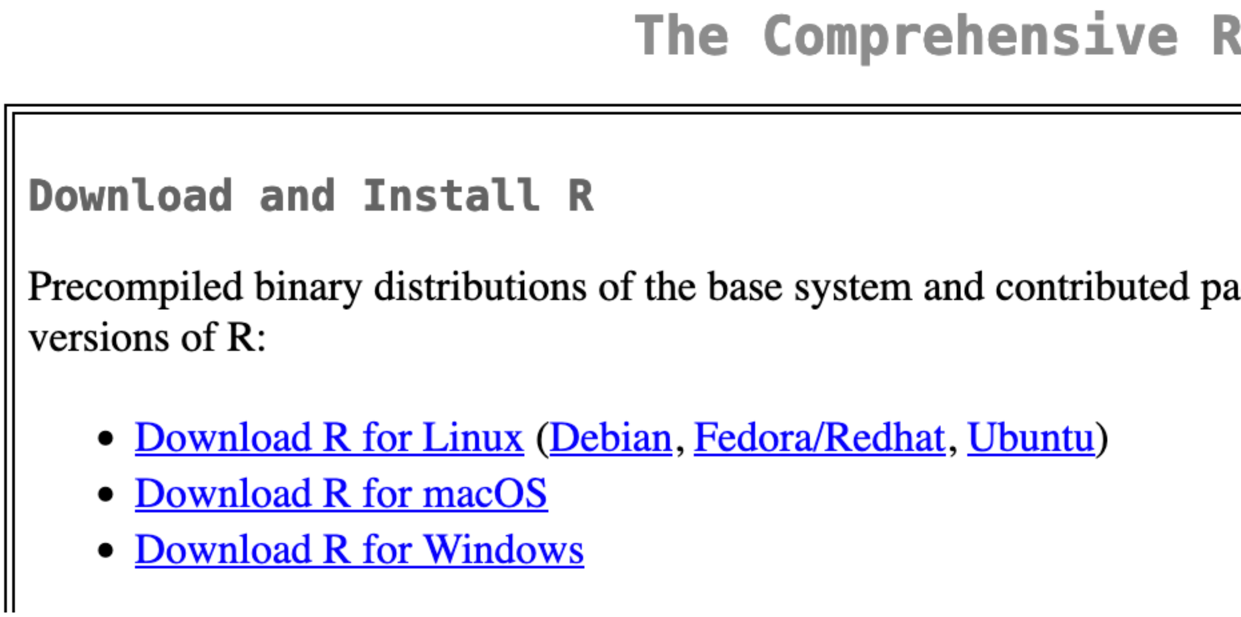
By the end of this tutorial, you'll have the knowledge to create, customize, and publish professional-grade documents.

Chapter 2

Getting Started

To get started with Bookdown you need to install R, RStudio, Bookdown, and the LaTeX distribution of your choice if you wish to output as a PDF. This can be accomplished by following these steps::

1. **Install R** Go to the R Project download page and download the latest version of R for your operating system (Windows, macOS, or Linux). Follow the installation instructions provided.



The Comprehensive R

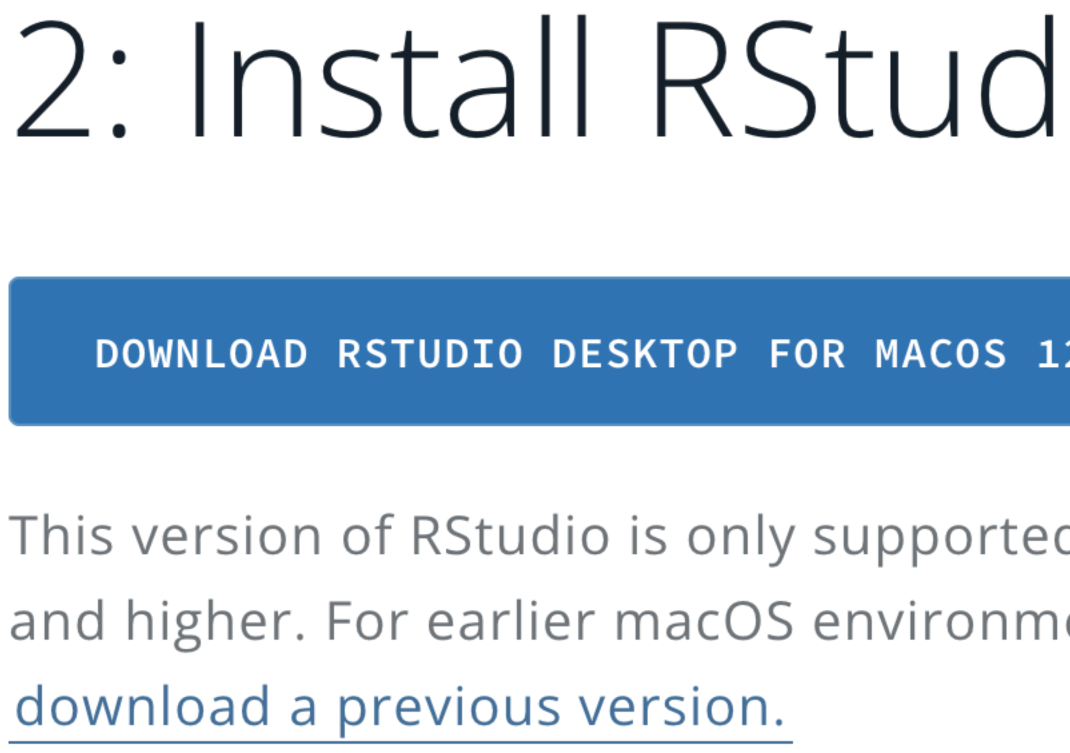
Download and Install R

Precompiled binary distributions of the base system and contributed pa
versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

2. **Install RStudio**:

Go to the RStudio download page and select the appropriate version for your operating system. Download and follow the installation instructions.



2: Install RStud

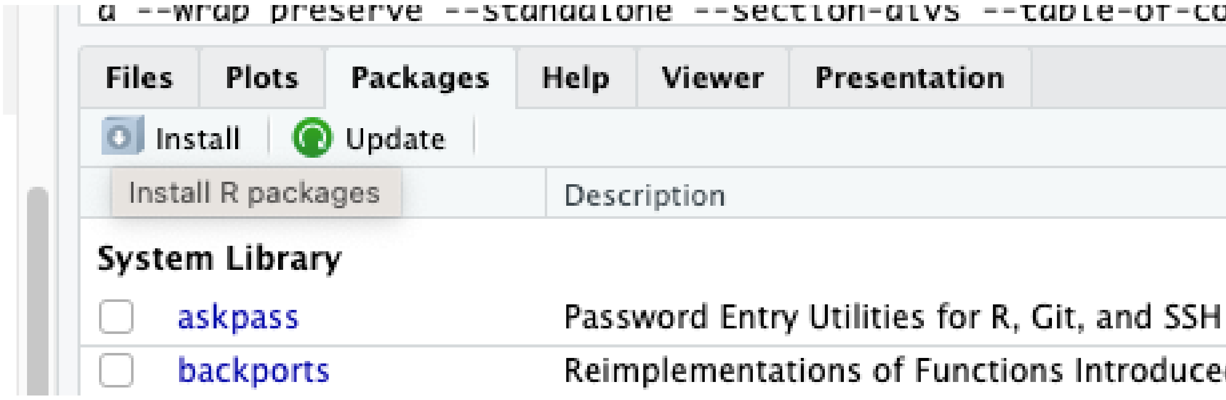
This version of RStudio is only supportec
and higher. For earlier macOS environm
download a previous version.

3. **Install Bookdown**:
   Once RStudio is installed, install the Bookdown package by typing the following command in the RStudio console:

```
install.packages("bookdown")
```

   Alternatively, you can install the Bookdown package via the RStudio **Packages** pane:

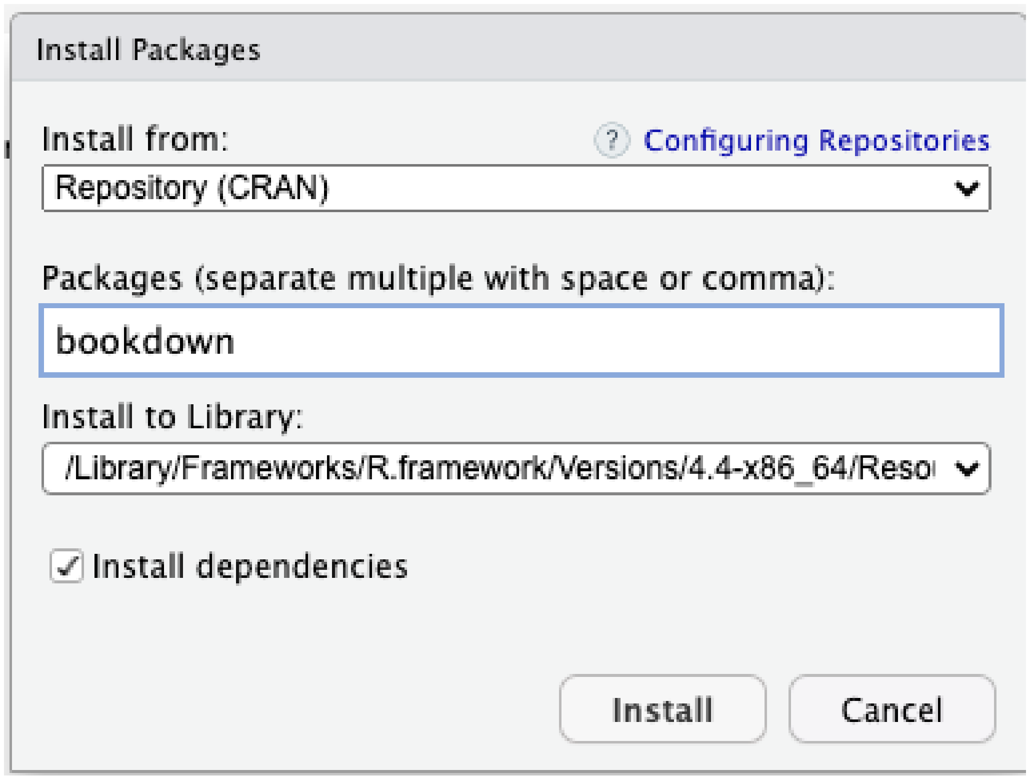
   - Select **Packages** in the bottom right-hand corner of RStudio.



System Library

☐	askpass	Password Entry Utilities for R, Git, and SSH
☐	backports	Reimplementations of Functions Introduce

- Click **Install**, type `bookdown` in the **Packages** box, and click **Install**.



Install from: Repository (CRAN)

Packages (separate multiple with space or comma): bookdown

Install to Library: /Library/Frameworks/R.framework/Versions/4.4-x86_64/Reso

✓ Install dependencies

4. **Install LaTeX distribution of your choice**: The distribution you choose is entirely up to you and your needs. For a list of recommended LaTeX distributions please see LaTeX Distributions Chapter 8

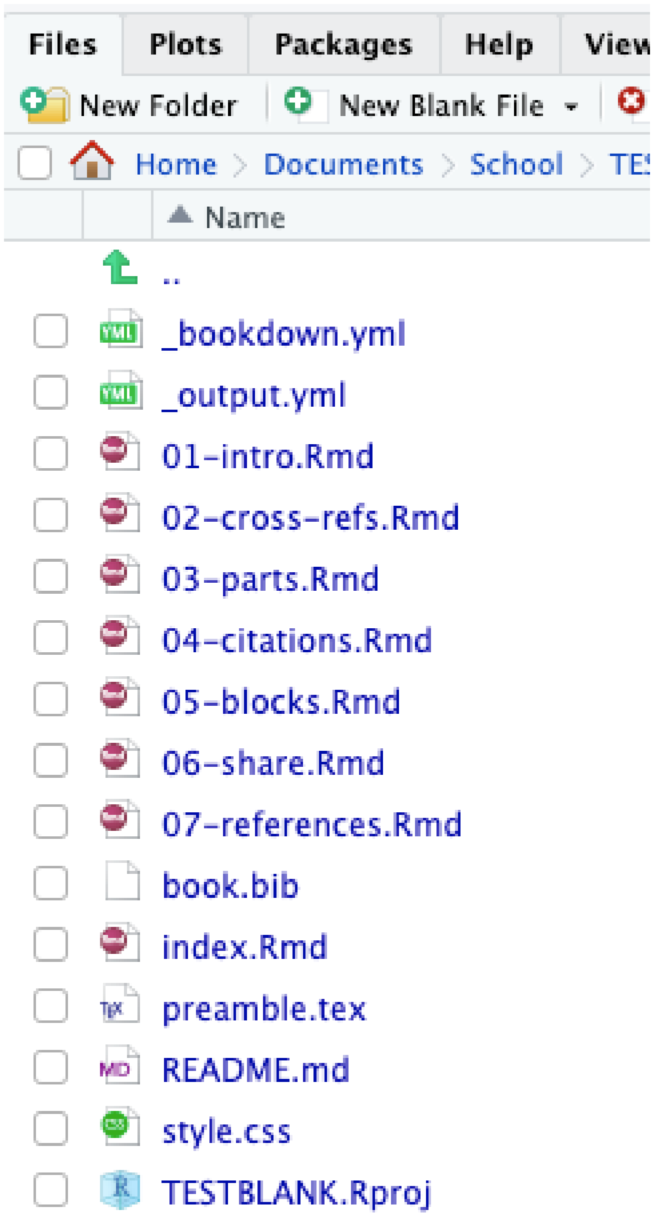
To get started quickly TinyTeX is recommended as it can be installed from within RStudio by running the following code:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

Once this is complete Bookdown is now installed and you are ready to create your first Bookdown project.

5. **Create a New Bookdown Project in RStudio**:
   - In RStudio, go to **File > New Project**.
   - Select **New Directory** and then **Book Project using Bookdown**.
   - Name your project and choose a location to save it to.

Now you have a newly created Bookdown project ready to be edited to fit your needs. Bookdown comes with a base set of files so that users can get started quickly with minimal fuss. Spend a few moments to explore the files in the bottom right corner of RStudio, these are the default files Bookdown creates to get you started. It even includes sample chapters for you!



```
_bookdown.yml
_output.yml
01-intro.Rmd
02-cross-refs.Rmd
03-parts.Rmd
04-citations.Rmd
05-blocks.Rmd
06-share.Rmd
07-references.Rmd
book.bib
index.Rmd
preamble.tex
README.md
style.css
TESTBLANK.Rproj
```

Let's go over quickly what you're seeing.

`_bookdown.yml` This file controls the settings for your Bookdown project, such as the order of the chapters and output options.

`_output.yml` This file specifics the formats your book will be rendered into, such as PDF, HTML, or Word.

`index.Rmd` Is the main file where your book starts. This page contains the title, author, and any other introductory information you may want your readers to know.

`01-intro.Rmd`, `02-cross-regs.Rmd`, `03-parts.Rmd`, `04-citations.Rmd`, `05-blocks.Rmd`, `06-share.Rmd` and `07-references.Rmd` are all sample chapters that the initial bookdown project creates for you. Starting from 01-07 these are the chapters after your Introduction and you can have as many as you want, or even just a single chapter. This way you have handy easy to edit chapters ready for you to edit, simplifying the work you need to get started.

`book.bib` is the reference file you place all your references you may need to cite in your book.

`preamble.tex` is where you enter any packages you may wish to use for your LaTeX entries.

`style.css` is where you will enter any custom CSS you may want to use to customize the look of your book in html.

When you're ready to complete your project just follow the last step below to render your book.

6. **Render Your Newly Created Book**:
   In the **Build** pane:
   - Select **Build Book** and choose your output format, or select *All formats* to render your files as HTML, PDF, and EPUB.
   - You can also render the book directly from the R console with the following command:

   ```
   bookdown::render_book("index.Rmd")
   ```

Chapter 3

Writing Content

In this chapter, we will explore how to write and structure content in Bookdown using R Markdown syntax. Bookdown allows you to create well-organized documents by combining text, code, and references. Here, we'll cover how to organize chapters, use Markdown and LaTeX for formatting, and format text, code chunks, and images.

3.1 Creating Chapters and Sections

Each chapter in Bookdown is represented by a separate `.Rmd` file, and each `.Rmd` file should begin with a first-level heading, marked by a single `#` symbol. For example, this chapter file is `02-writing-content.Rmd` and the file starts with:

```
# Writing Content
```

3.1.1 Organizing Chapters

Chapters are automatically numbered based on their order in the project directory. Ensure that each file name reflects its chapter number (e.g., `02-writing-content.Rmd` for Chapter 2). So if you need to add new chapters to your book just make sure you adjust the filename of any other chapters you already have to ensure it fits. For example, if you need to add a new chapter between 3 and 4, you would need to make sure that you rename your chpater 4 file from 03 to 04, and your new file would start with 03.

3.1.2 Adding Sections and Subsections

You can add sections and subsections within a chapter using second-level and higher-level headings:

```
## Section Title
### Subsection Title
```

This hierarchy organizes the document according to your needs, and these sections will automatically appear in the table of contents.

3.2 Formatting Text in Bookdown

Bookdown supports a wide range of Markdown formatting. Here are a few basics:

- **Bold**: `**bold text**` → **bold text**
- *Italics*: `*italicized text*` → *italicized text*
- **Bullet Points**:
  - First item
  - Second item
- **Numbered Lists**:
  1. First item
     - Even sublists
     - Like this
  2. Second item

3.2.1 Creating Tables in Bookdown

Tables are a powerful way to organize and present data in Bookdown. You can create tables using basic Markdown syntax.

3.3 Creating Basic Markdown Tables

It's quite easy to create your own table as well. All you need to do is format it as such:

```
| Column 1     | Column 2     | Column 3     |
|--------------|--------------|--------------|
| Row 1, Col 1 | Row 1, Col 2 | Row 1, Col 3 |
| Row 2, Col 1 | Row 2, Col 2 | Row 2, Col 3 |
```

This will then be outputted as:

Column 1	Column 2	Column 3
Row 1, Col 1	Row 1, Col 2	Row 1, Col 3
Row 2, Col 1	Row 2, Col 2	Row 2, Col 3

For additional text formatting options refer to Chapter 9 Advanced Text Formatting.

Use these formatting options to style text and create lists within your chapters.

3.4 Adding Code Chunks

One of the strengths of Bookdown is the ability to incorporate code into your document, whether it's R code, Markdown, LaTeX, Python, or other languages. Below are examples of how to include different types of code in your Bookdown project.

3.4.1 Example: Loading Data

To include R code chunks in your Bookdown project, use three backticks ``` with `{r}` specifying the language. For other languages you would just change r to the language of your choice such as Python or Java. For example say you want to show users how you set a working directory for your data analysis on Union Wages, you could do it as follows:

```
{r eval=FALSE, warning=FALSE, include=TRUE, results='hide'}
# Set the working directory and load data files
setwd("~/Documents/School/RaceIncomeCalifornia")
demographics <- readr::read_csv('demographics.csv')
wages <- readr::read_csv('wages.csv')
states <- readr::read_csv('states.csv')
```

In this example, the `eval=FALSE` option prevents the code from being executed during rendering, while the `warning=FALSE` and `include=TRUE` options suppress warnings and ensure the chunk is included in the output.

3.4.2 Customizing Code Chunk Options

Use chunk options to control how the code and output appear in the document. Here is a short list of different code chunk options that you have as a user. All code chunk options are formatted with an equal sign after them, before TRUE, FALSE, or other option, when adding them to your code chunk as such: `eval=TRUE` or `fig.align='center'`. Here is a list of some basic and handy code chunk options to get you started.

Basic Options

Option	**Description**
`eval`	Executes the code if `TRUE` (default); skips execution if `FALSE`.
`echo`	Displays the code in the output if `TRUE` (default); hides code if `FALSE`.
`results`	Determines how results are displayed (`"markup"`, `"asis"`, `"hide"`).
`message`	Displays messages from the code if `TRUE` (default); hides them if `FALSE`.
`warning`	Displays warnings if `TRUE` (default); suppresses them if `FALSE`.
`error`	Displays error messages if `TRUE`; suppresses them if `FALSE`.

Output Control

Option	**Description**
`fig.width`	Width of the generated figure (in inches).
`fig.height`	Height of the generated figure (in inches).
`fig.align`	Aligns the figure (`"left"`, `"right"`, `"center"`).
`fig.cap`	Adds a caption to the figure (e.g., `fig.cap="Figure caption"`).
`out.width`	Width of the figure output (e.g., `out.width="50%"`).
`out.height`	Height of the figure output (similar to `out.width`).
`fig.show`	Controls figure display (`"asis"`, `"hold"`, `"hide"`).

As an example say you need to run some behind the scenes data manipulation of what you've downloaded so that you can do an analysis and plot later on in your document. You could set your code chunk options so that it runs but is hidden by setting `eval=TRUE` so that it executes the code, `echo=FALSE` to not display the code, then `results=hide`, `warning=FALSE`, and `message=FALSE` to prevent any results or possible warning messages from appear in your code.

```
{r, eval=TRUE, echo=FALSE, results='hide', warning=FALSE, message=FALSE}

setwd("~/Documents/School/RaceIncomeCalifornia")
demographics <- readr::read_csv('demographics.csv')
wages <- readr::read_csv('wages.csv')
states <- readr::read_csv('states.csv')

filtered_wages <- wages[wages$year >= 2000, ]
filtered_states <- states[states$year >= 2000, ]
filtered_demographics <- demographics[demographics$year >= 2000, ]
```

```
merged_data <- merge(states, wages, by = "year", all.x = TRUE)


if (!requireNamespace("data.table", quietly = TRUE)) {
  install.packages("data.table")
}
library(data.table)

setDT(filtered_wages)
setDT(filtered_states)
setDT(filtered_demographics)

wage_trends <- filtered_wages[,
      list(mean_union_wage = mean(union_wage,
                                 na.rm = TRUE),
      mean_nonunion_wage = mean(nonunion_wage,
                               na.rm = TRUE)), by = list(year)]

union_wage_premium <- filtered_wages[,
      list(raw_premium = mean(union_wage_premium_raw, na.rm = TRUE),
      adjusted_premium = mean(union_wage_premium_adjusted,
                             na.rm = TRUE)), by = year]

membership_trends <- filtered_demographics[,
        list(average_p_members = mean(p_members, na.rm = TRUE)), by = year]

average_wage_by_year <- filtered_wages[,
      list(average_wage = (mean(union_wage, na.rm = TRUE) +
      mean(nonunion_wage, na.rm = TRUE)) / 2),
        by = list(year)]

merged_data$members_scaled <- merged_data$members * 100
```

When rendered, this chunk will execute in the background, ensuring the data is processed, but no code, output, warnings, or messages will be visible in your final document, this way you can ensure it gets run every time your book is outputted.

3.5 Adding Plots and Images to Your Bookdown Project

Plots and images can enhance your document by providing visual context. In Bookdown, you can add plots using R code, use Markdown or by embedding images using R code chunks.

3.5.1 Adding Images with RStudio GGPLOT

The easiest way to add images more relevant to your needs is with the built in plotting functions of RStudio, or the ggplot package. Simply add an R code chunk and enter the code you wish to run. For example, say you you want to show a plot showing the wage increases for Union Members compared to non-Union members.

```
ggplot(wages, aes(x = year)) +
  geom_smooth(aes(y = union_wage, color = "Union Wage"),
              se = FALSE, method = "loess") +
  geom_smooth(aes(y = nonunion_wage, color = "Nonunion Wage"),
              se = FALSE, method = "loess") +
  labs(x = "Year", y = "Wage",
       color = "Type of Wage") +
  theme_minimal(base_size = 14) +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("Union Wage" = "blue", "Nonunion Wage" = "green")) +
  guides(color = guide_legend(title = "Wage Type"))
```

As you can see the above code then creates a plot and places it in your document for you. This way you can fully customize the plot from within RStudio.

3.5.2 Adding Images Using Markdown

To add an image using Markdown, use the following syntax:

```
![Alt text for the image](images/wagesector.png)
```

- **Alt text**: A description of the image that is useful for accessibility.
- **Path to the image**: This can be a relative path (e.g., `images/wagesector.png`) or a URL.

Example:

```
![A sample image](images/wagesector.png)
```

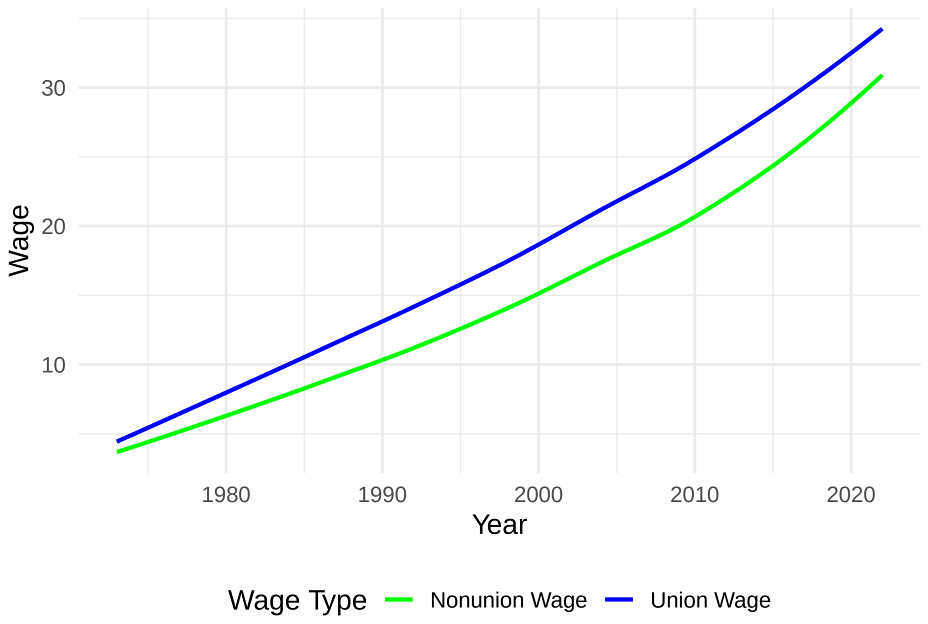


Figure 3.1: Union Vs Non-Union Wages

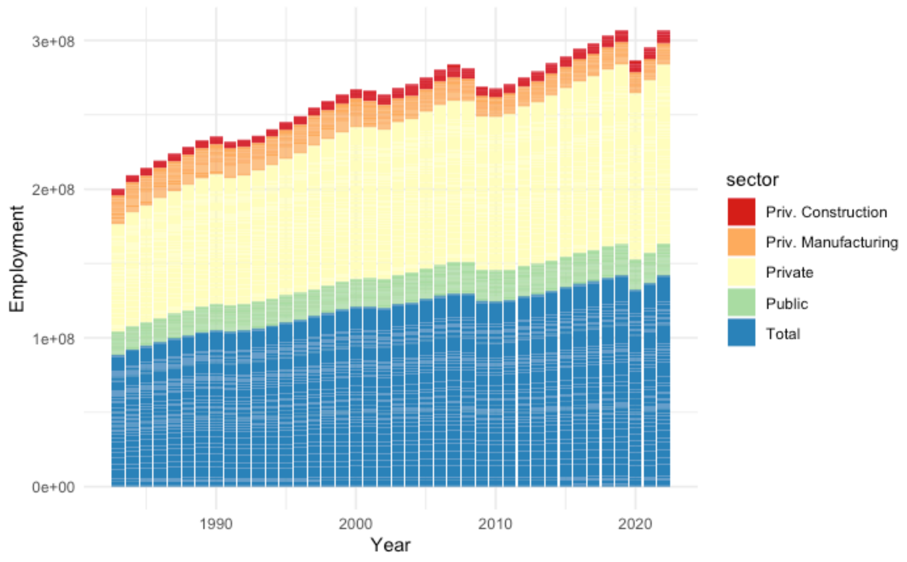
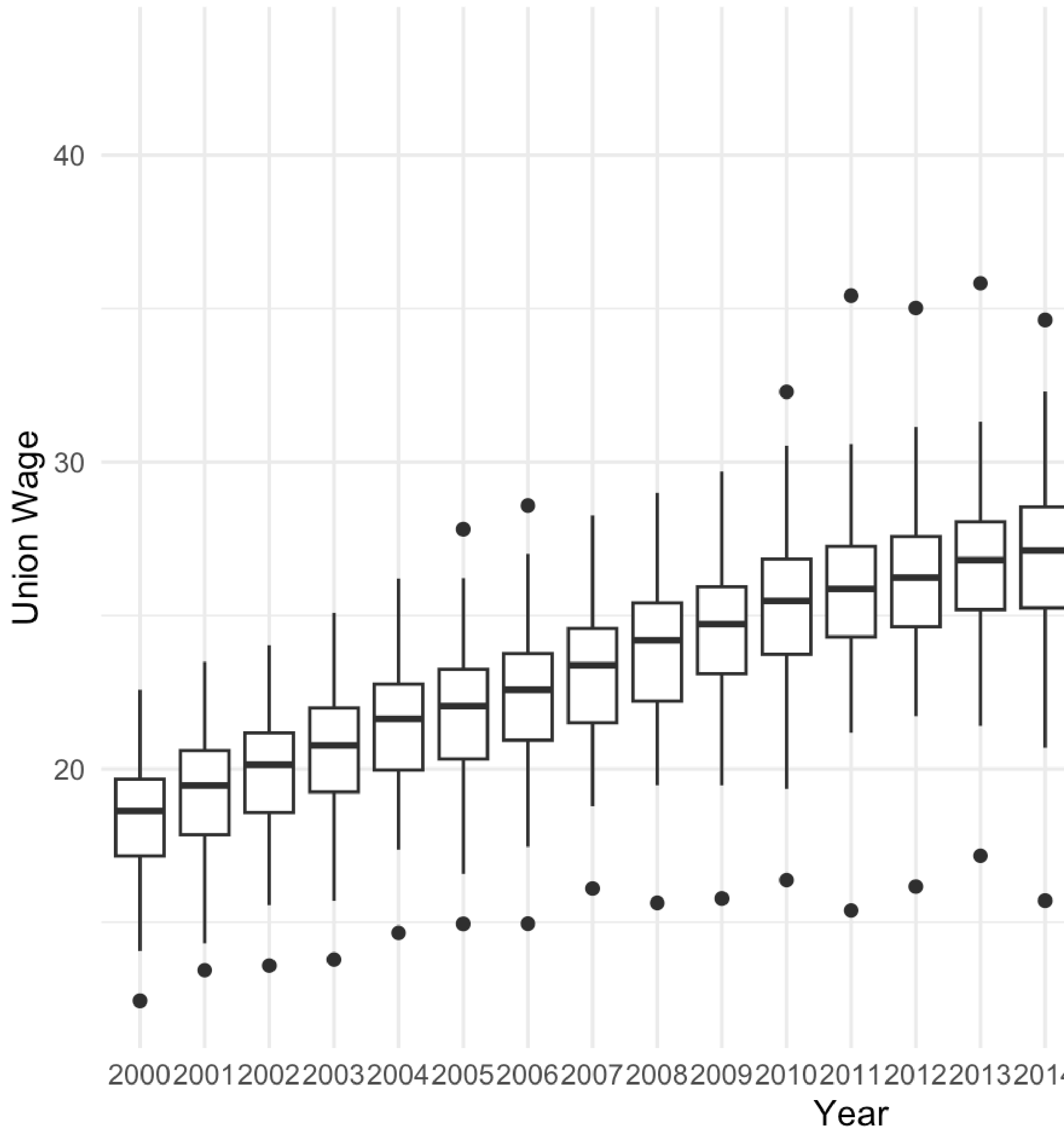


Figure 3.2: A sample image

3.5.3 Adding Images Using Code Chunks

You can also add images using an R code chunk, which can be useful when the image is generated programmatically. Use the `knitr::include_graphics()` function:

```r
knitr::include_graphics("images/wageyear-1.png")
```


40
30
20
Union Wage
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
Year

3.6 Chunk Options for Images

When adding images through code chunks, you can customize their appearance using chunk options such as `fig.cap` for captions and `out.width` for sizing.

For example the following has the caption `fig.cap="An example image"` and has an output width of 25% with `out.width="25%`. As you can see the image is much smaller than allowing the output width to not be adjusted.

```
knitr::include_graphics("images/wageyear-1.png")
```

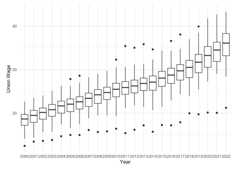


Figure 3.3: wageyear

EXERCISE TIME!

Create a new .Rmd file and write a short chapter that includes different types of Markdown elements (e.g., headers, tables, and images). Experiment with adding code chunks, and text formatting.

Chapter 4

Cross Referencing and Citations

Cross-references make it easier for readers to find and link to elements in your book. In Bookdown, you can create cross-references for sections, figures, tables, and equations. This chapter explains how to use cross-references effectively.

4.1 Cross-Referencing Sections

To reference a section, first add an ID or tag to the heading by including `{#your-id}` at the end of the section header. As an example here is the last chapter of the book that is an analysis I created within bookdown set as a single chapter:

```
# Example Document: Union Earnings Analysis (#examplepaper)
```

You can then refer to it later, with a clickable link that will take your reader directly to it, in your document as follows:

```
See Section \@ref(examplepaper) for more information.
```

4.2 See Section 10 for more information.

4.2.1 Cross-Referencing Text

You can assign some text to a label and reference the text using the label elsewhere in your document. This can be particularly useful for long figure/table captions or when you need to reuse text fragments in multiple places.

The syntax for a text reference is `(ref:label) text`, where `label` is a unique label throughout the document. It must be in a separate paragraph with empty lines above and below it. The paragraph must not be wrapped into multiple lines, and should not end with whitespace.

Additional Guidelines: - Ensure that text references are in a separate paragraph with empty lines above and below. - Avoid wrapping the paragraph containing the text reference into multiple lines. - Make sure there is no trailing whitespace at the end of the paragraph.

Example:

```
Then you can reference this text in your document using \texttt{(ref:textreference)}.

For example:

Then we reference it:
This is the text we are referencing
```

.

4.2.2 Cross-Referencing Figures

To cross-reference a figure, set a chunk label and use the `fig.cap` option to add a caption. Bookdown automatically labels the figure with `fig:chunk-label`.

For example in the previous chapter we showed a plot comparing Union Wages vs Non-Union Wages. It had the following in the code chunk heading:

`{r wagecomparison, fig.cap="Union Vs Non-Union Wages"}`

We can easily create a reference back to this plot, that is clickable so users can easily go back to it, with the following:

```
See Figure \@ref(fig:wagecomparison) for details.
```

See Figure 3.1 for details.

4.2.3 Cross-Referencing Tables

To cross-reference a table, use `knitr::kable()` to create a table with a caption. Bookdown automatically labels the table with `tab:chunk-label`.

Table 4.1: Table of the first rows of the cars dataset

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

Example:

```
knitr::kable(head(cars), caption = "Table of the first rows of the cars dataset")
```

Reference this table using:

```
See Table \@ref(tab:cars-table) for details.
```

See Table 4.1 for details.

4.3 Cross-Referencing Equations

For equations, use `(\#eq:label)` to label the equation and `\@ref(eq:label)` to reference it. This way when you need to reference the equation at a later point in your document you can.

4.3.1 Creating and Labeling an Equation

First you write the equation in LaTeX in the equation environment and include the label as `(\#eq:label)`:

```
\begin{equation}
  E = mc^2
  (\#eq:einstein)
\end{equation}
```

Which outputs as such:

$$E = mc^2 \tag{4.1}$$

Then at a later point when you need to reference it you simply type in `\@ref(eq:label)`. As we know Einsten discovered that (5.1).

4.4 Adding Citations and Managing References

Bookdown makes it easy to manage references and add citations by using BibTeX files. Here's how to set up and include references in your Bookdown project.

4.4.1 Step 1: Create a `.bib` File

First, create a `.bib` file for your references (e.g., `references.bib`). You can add references in BibTeX format. Here's an example entry:

```
@Book{Bookdown,
  title     = {Bookdown: Authoring Books and Technical Documents with R Markdown},
  author    = {Yihui Xie},
  publisher = {Chapman; Hall/CRC},
  year      = 2024,
  edition   = {2nd},
  note      = {ISBN 9780367142568},
  url       = {https://bookdown.org/yihui/bookdown/}
}
```

4.4.2 Step 2: Link the `.bib` File in `index.Rmd`

In your `index.Rmd` file, include the `.bib` file in the YAML header:

```
bibliography: [references.bib]
link-citations: yes
```

4.4.3 Step 3: Cite Sources in Your Text

To cite a source, use `[@citation-key]` in your text, where `citation-key` matches the key in your `.bib` file (e.g., `[@Bookdown]`). For example:

```
This tutorial was written thanks to Bookdown [@Bookdown].
```

This tutorial was written thanks to Bookdown [Xie, 2024].

Bookdown will automatically format your citation based on the output style.

4.4.4 Step 4: Customize Citation Style (Optional)

If you need a specific citation style, you can add a `.csl` (Citation Style Language) file in your project and reference it in the YAML header:

```
csl: "chicago-author-date.csl"
```

Download `.csl` files from sources like Zotero.

4.5 Using Cross-References with Citations

In addition to referencing external sources, Bookdown allows you to cross-reference sections, figures, and tables, as discussed earlier. As a reminder:

```
As shown in Figure \@ref(fig:example-figure), the trend is evident.
```

4.6 Using cross-references with citations helps keep your document organized and easy to navigate.

EXERCISE TIME!

Go ahead and place a few entries in your .bib file. Once you've done that go through and reference these in your previously written content.

Chapter 5

LaTeX in Bookdown

Bookdown offers powerful support for LaTeX, allowing you to seamlessly integrate any LaTeX packages you need into your documents. Whether you're working with mathematical equations, theorems, lemmas, proofs, or other advanced features, this tutorial will guide you through the essentials of using LaTeX in Bookdown, showing you how to effectively incorporate and reference these elements.

5.1 Including LaTeX Packages in Bookdown

One of the powerful features of Bookdown is its seamless integration with LaTeX, allowing you to include any LaTeX package that suits your needs. This flexibility is especially useful when working with advanced mathematical notations, custom formatting, or specialized content.

5.2 Using a `preamble.tex` File

To include LaTeX packages, you need to create a `preamble.tex` file and link it in your `_output.yml` file. Bookdown created this `preamble.tex` file for you when you started this tutorial so you don't need to create it again. The `preamble.tex` file is processed before the document is rendered allowing you to load any additional LaTeX packages you may need. The `preamble.tex` file also allows you to fully customize the output of your PDF using your existing LaTeX knowledge.

1. **Create a `preamble.tex` File** Create a file on your computer with the name `preamble.tex` and add it to the directory of your Bookdown project if it does not already exist. If the file already exists simply open the file within RStudi for easy editing. You can include as many packages as you

need in the `preamble.tex` file. For example here is a list of packages and small changes I made to output the final example document in this tutorial as an a6 sized document:

```
\usepackage{booktabs}
\usepackage{anyfontsize}
\usepackage{titlesec}
\usepackage{fancyhdr}
\usepackage[paperwidth=4.13in, paperheight=5.83in]{geometry}
\usepackage[font=small, labelfont=bf]{caption}
\geometry{left=.5in}
\geometry{right=.5in}
\geometry{bottom=.5in}
\geometry{top=.75in}
```

2. **Link `preamble.tex` in `_output.yml`**
   You need to make sure that the preamble is included in your _output file. Update your `_output.yml` file to include the `preamble.tex` file for PDF output:

```
bookdown::pdf_book:
  includes:
    in_header: preamble.tex
  latex_engine: xelatex
  citation_package: natbib
  keep_tex: yes
```

5.3 Mathematical Equations

Bookdown makes it easy to include both inline and display-style equations while also allowing you to reference these equations for easy reference at any point within your project.

5.4 Inline Equations

Use `$...$` to include inline math equations within your text. The following code:

```
The formula for the area of a circle is $( A = \pi r^2 )$, where $( r )$ is the radius
```

Then is displayed as follows:

The formula for the area of a circle is $(A = \pi r^2)$, where (r) is the radius.

5.5 Display Equations

For equations that you may want to have on their own line you have two options. You can surround your LaTeX code `$$...$$` or alternatively use the LaTeX `equation` environment.

Here we see the usage of `$$...$$`

```
$$
E = mc^2
$$
```

Which gives us:

$$E = mc^2$$

Here we are using the `equation` environment with a label:

```
\begin{equation}
  E = mc^2
  (\#eq:einstein)
\end{equation}
```

Which outputs as such:

$$E = mc^2 \tag{5.1}$$

Reminder for referencing equations: For easy reference within your document we are also able to add a reference to any equation by using `\#eq:your-label` as such:

`\label{eq:einstein}`

So that you can easily reference it later with `\@ref(eq:einstein)`. As an example, we labeled the above equation as `(\#eq:einstein)`. So we can reference it by using `\@ref(eq:einstein)`:

```
As shown in Equation \@ref(eq:einstein), energy is proportional to mass.
```

As shown in Equation (5.1), energy is proportional to mass.

5.6 Theorems, Lemmas, and Proofs

Bookdown supports theorems, lemmas, and proofs, for structured mathematical writing.

5.7 Adding a Theorem

Define a theorem using the following syntax:

```
::: {.theorem #theoremlabel}
This is a `theorem` environment that can contain **any**
_Markdown_ syntax.
For any integer $n \geq 1$, the sum of the first $n$ positive integers is given by
$$
S = \frac{n(n + 1)}{2}
$$
:::
```

Reference the theorem in your text by add `#theoremlabel` to your code block as such:

```
::: {.theorem #theoremlabel}
This is a `theorem` environment that can contain **any**
_Markdown_ syntax.
For any integer $n \geq 1$, the sum of the first $n$ positive integers is given by
$$
S = \frac{n(n + 1)}{2}
$$
:::
```

Which will cause your theorem to display as follows:

Theorem 5.1. *This is a* `theorem` *environment that can contain* ***any*** Markdown *syntax. For any integer $n \geq 1$, the sum of the first n positive integers is given by*

$$S = \frac{n(n+1)}{2}$$

If this is a theorem that you need to reference in other parts of your document you can label it as we did above with `#theoremlabel`. Then when you need to reference it you can do so using `\@ref(thm:theoremlabel)` wherever needed. As an example, lets refer back to the above Theorem 5.1.

5.8 Adding a Lemma

Define a lemma similarly:

```
::: {.lemma #lemmalabel}
Let $x\in \mathbb Z$.If $5x-7$ is odd,then $x$ is even.
:::
```

Lemma 5.1. *Let $x \in \mathbb{Z}$. If $5x-7$ is odd,then x is even.*

Reference the lemma:

```
Lemma \@ref(lem:lemmalabel) confirms that the sum of two even integers is even.
```

Lemma 5.1 confirms that the sum of two even integers is even.

5.9 Adding a Proof

Proofs can be added using the `proof` environment.

A

```
::: {.proof}
Let $x \in \mathbb Z$. Suppose $7x+5$ is odd. Then $7x+5=2k+1$ for some $k \in \mathbb Z$. Then
\begin{align*}
    7x+5&=2k+1\\
    7x&=2k-4\\
    x&=2k-6x-4\\
    x&=2(k-3x-2).
\end{align*}
Since $k-3x-2 \in \mathbb Z$, $x$ is even.
:::
```

Proof. Let $x \in \mathbb{Z}$. Suppose $7x+5$ is odd. Then $7x+5=2k+1$ for some $k \in \mathbb{Z}$. Then

$$\begin{aligned}
7x+5&=2k+1\\
7x&=2k-4\\
x&=2k-6x-4\\
x&=2(k-3x-2).
\end{aligned}$$

Since $k-3x-2 \in \mathbb{Z}$, x is even. □

5.10 Referencing Various Math Environments

When you need to refer to create a different environment you just need to change what's between the brackets to the type of environment you are using.

Table 5.1: Theorem environments in Bookdown.

Environment	Printed Name	Label Prefix
theorem	Theorem	thm
lemma	Lemma	lem
corollary	Corollary	cor
proposition	Proposition	prp
conjecture	Conjecture	cnj
definition	Definition	def
example	Example	exm
exercise	Exercise	exr
hypothesis	Hypothesis	hyp

```
::: {.environment}
Your information goes here
:::
```

When you are referencing the environment you would just ensure that you use the label prefix from the below table for the environment you wish to reference, changing the word prefix in:`\@ref(prefix:label)` to the appropriate environment.

5.11 Tips for LaTeX in Bookdown

- **Use Labels Consistently**: Use meaningful and unique labels for cross-referencing.
- **Use Math Mode**: Always enclose mathematical symbols in `$...$` or `$$...$$` to render correctly.
- **Add Theorem Styles**: Customize theorem environments in `_bookdown.yml` for specific needs: `markdown theorem: lab: "Theorem " lem: "Lemma "`

EXERCISE TIME!

Add an equation and a theorem to your document using LaTeX. Alternatively, copy over some LaTeX code you've written before and place it in one of your chapters.

Chapter 6

Advanced Features

What we have covered before was just the basics. Now lets get into some of the more advanced features that one can perform in Bookdown to customize the look of your document.

6.1 Using LaTeX for Advanced Formatting

Bookdown supports advanced formatting using LaTeX when PDf is chosen as an output option. This allows you to create custom environments and modify your PDF output document to fit your needs. This means that if you want to add custom boxes to draw users attention to specific items or text, change how the PDF is outputted, or anything else you can think of with LaTeX

6.1.1 Example: Adding a Custom Box

Lets say you want to create a note box that is black so that you can ensure users are drawn to specific bits of information. You could then define a custom LaTeX environment for a “black note box” in the `preamble.tex` file as follows:

```
\setlength{\fboxsep}{.8em}

\newenvironment{blackbox}{
  \definecolor{shadecolor}{rgb}{0, 0, 0}
  \color{white}
  \begin{shaded}
  }
 {\end{shaded}}
```

Lets review each part:

First we have: `\setlength{\fboxsep}{.8em}`. This adjusts the padding (separation) inside the box. Here we specifically set the distance between the content and the box edges to `0.8em`.

Then we define a new environment and named it `blackbox`: `\newenvironment{blackbox}{...}{...}`. The first part (`{...}`) is executed at the start of the environment. The second part (`{...}`) is executed at the end of the environment.

Within the first set of curly brackets we have defined the background color to black using the RGB values of 0,0,0:`\definecolor{shadecolor}{rgb}{0, 0, 0}`.

We then changed the color of the text inside this environment with: `\color{white}`

To then create the shading around the boxes border we added `\begin{shaded}` within the first set of curly brackets. - The background color is determined by `shadecolor`.

Then you can call this new custom created environment using the following:

```
:::: {.blackbox data-latex=""}
::: {.center data-latex=""}
**EXAMPLE!**
:::
Giving you this as an output.
::::
```

Which gives you the following output in your PDF document. Below we will also cover how to do the same for your HTML, EPUB, and Word output.

EXAMPLE!

Giving you this as an output.

6.2 Customizing Document with LaTeX

Thanks to LaTeX you can even customize the entire look and output of your document by modifying the content in the `premable.txt` file. For example, say you wish to create an a6 size booklet (4.25” x 5.5”) and so need to modify the size of the font for headers, subheaders, and text to ensure that the words fit on the page but are not too small. Below is the code I used to cutomize the output of the example document at the end of this tutorial. In the preamble would look as follows:

```
\usepackage{booktabs}
\usepackage{anyfontsize}
\usepackage{titlesec}
\usepackage{fancyhdr}
\usepackage[paperwidth=4.25in, paperheight=5.5in]{geometry}
\usepackage[font=small, labelfont=bf]{caption}
\geometry{left=.5in}
\geometry{right=.5in}
\geometry{bottom=.5in}
\geometry{top=.75in}
\renewcommand{\normalsize}{\fontsize{6pt}{8pt}\selectfont}
\titleformat{\section}{\fontsize{8pt}{10pt}\bfseries}{\thesection}{1em}{}
\titleformat{\subsection}{\fontsize{6pt}{8pt}\bfseries}{\thesubsection}{1em}{}
\titleformat{\subsubsection}{\fontsize{7pt}{9pt}\bfseries}{\thesubsubsection}{1em}{}
\usepackage{tocloft}
\renewcommand{\cftsecfont}{\fontsize{6pt}{8pt}}
\renewcommand{\cftsubsecfont}{\fontsize{7pt}{9pt}}
\renewcommand{\cftsubsubsecfont}{\fontsize{6pt}{8pt}}
\usepackage{listings}
\lstset{
  basicstyle=\ttfamily\fontsize{2pt}{4pt},
  breaklines=false,
  frame=single,
}
\usepackage{titlesec}
\titleformat{\chapter}[display]
  {\fontsize{14pt}{18pt}\bfseries}
  {\chaptername~\thechapter}
  {1em}
  {}
\titleformat{\section}
  {\fontsize{10pt}{12pt}\bfseries}
  {\thesection}
  {1em}
  {}
\titleformat{\subsection}
  {\fontsize{10pt}{12pt}\bfseries}
  {\thesubsection}
  {1em}
  {}
\setlength{\fboxsep}{.8em}
```

6.2.1 Code Explanation

Let us review each part:

First, we have `\usepackage{booktabs}`. This package is used to create high-quality tables with professional-looking horizontal rules. It helps improve the overall appearance of tables.

Next is `\usepackage{anyfontsize`. This package allows for the use of arbitrary font sizes beyond LaTeX's default options, giving you finer control over typography.

Following that, we have `\usepackage{titlesec}`, which customizes the formatting of section titles, including subsections and subsubsections. This allows you to adjust font size, style, and alignment for headings.

Then comes `\usepackage{fancyhdr}`, which is used to customize headers and footers in your document. This adds a layer of personalization and improves the document's usability.

After that is `\usepackage[paperwidth=4.13in, paperheight=5.83in]{geometry}`. This sets the page size to A6 dimensions (4.13 inches by 5.83 inches), making it suitable for smaller-format documents.

In the line `\geometry{left=.5in}`, we adjust the page margins. Here, the left and right margins are set to 0.5 inches, while the bottom margin is also 0.5 inches, and the top margin is slightly larger at 0.75 inches.

Now, let us look at `\renewcommand{\normalsize}{\fontsize{6pt}{8pt}\selectfont}`. This command redefines the default font size (`\normalsize`) to 6 points, with line spacing set to 8 points, ensuring the text fits compact page formats.

Next, we see several `\titleformat` commands. These customize how titles for different sections appear: - `\section` is formatted with 8pt font size and bold text. - `\subsection` uses a 6pt font size and bold text. - `\subsubsection` is set to a 7pt font size, also in bold. - `\chapter` is customized with a larger 14pt font size and bold text.

The package `\usepackage{tocloft}` is used to format the table of contents. It allows precise control over how section entries, subsections, and subsubsections appear in the contents list.

The command `\renewcommand{\cftsecfont}{...}` customizes the font sizes of section, subsection, and subsubsection entries in the table of contents. For example: - `\cftsecfont` sets section entries to a 6pt font size. - `\cftsubsecfont` sets subsection entries to a slightly larger 7pt font size. - `\cftsubsubsecfont` ensures subsubsection entries are consistent with 6pt font size.

The next part is `\usepackage{listings}`, which enables syntax highlighting for code listings. It is particularly useful for displaying programming or markup language code.

Within the `\lstset{...}` block, we configure the appearance of the code listings: - `basicstyle=\ttfamily\fontsize{2pt}{4pt}` sets a very small monospace font. - `breaklines=false` ensures that long lines of code do not wrap onto the next line. - `frame=single` adds a border around the code block.

Finally, the `\titleformat{\chapter}[display]` command customizes how chapter titles are displayed: - Chapter titles use a 14pt font size with bold text. - The format includes "Chapter [number]" followed by the title. - A 1em space is added between the chapter number and the title text.

Together, these configurations provide a compact and professional layout tailored for smaller documents while allowing customization options for headings, tables, and code listings.

6.3 Adding Custom CSS

You can enhance the appearance and functionality of your Bookdown project by adding custom CSS or Javascript. So if you want your black box to appear not only in the PDF output of your bookdown project, but also in your HTML, EPUB and Word file this is where you would add the code for it. If a `.css` file doesn't exist yet you first need to create one. By default Bookdown has already created the file for you.

6.3.1 Step 1: Create a Custom CSS File

Create a `styles.css` file with then fill it with some basic stlying as shown:

```
body {
  font-family: "Arial", sans-serif;
}
h1 {
  color: #4CAF50;
}
```

6.3.2 Step 2: Include the CSS File in the YAML Header

Add the following to your `index.Rmd` YAML header. This has already been done when the Bookdown project was created:

```
output:
  bookdown::gitbook:
    css: styles.css
```

6.3.3 Step 3: Adding JavaScript (Optional)

For dynamic behavior, include JavaScript files similarly:

```
output:
  bookdown::gitbook:
    includes:
      in_header: "scripts.js"
```

6.4 Adding Custom Elements

So say you want to make sure the blackbox from earlier appears not only in your PDF file but in your HTML, EPUB, and WORD file. To do this you would add the following to your `style.css` file. The `.blackbox` CSS class is used to style elements with a visually distinct box design.

```
.blackbox {
  padding: 1em;
  background: black;
  color: white;
  border: 2px solid orange;
  border-radius: 10px;
}
```

Let us review the CSS for `.blackbox` step by step:

First, we have: `.blackbox`. This targets elements with the `class="blackbox"` in your HTML, applying the styles defined within this block.
- The `padding: 1em;` adds space between the content and the edges of the box, ensuring the text does not touch the border. This creates a cleaner, more readable layout.
- The `background: black;` sets the background color of the element to black, making the box visually distinct and suitable for highlighting content.
- The `color: white;` changes the text color inside the box to white, ensuring a strong contrast against the black background for readability.

Next, we have: `border: 2px solid orange;`. This creates a border around the box:
- The thickness of the border is set to 2px, providing a clear but not overly dominant outline.
- The `solid` style ensures the border appears as a continuous line, and the orange color gives it a bright, attention-grabbing appearance.

Finally, the `border-radius: 10px;` rounds the corners of the box:
- This style softens the sharp edges of the box, giving it a smoother, more modern look.
- The 10px radius is subtle enough to maintain a professional appearance while adding a touch of design flair.

6.5 Customizing Document with CSS

Through CSS style editing you can customize the HTML output of your project to meet your needs. Here are the CSS edits that were made to this project:

```
p.caption {
  color: #777;
  margin-top: 10px;
}
p code {
  white-space: inherit;
}
pre {
  word-break: normal;
  word-wrap: normal;
}
pre code {
  white-space: inherit;
}
.center {
  text-align: center;
}
h1, h2, h3, h4, h5, h6 {
    font-family: "Times New Roman", Times, serif;
    color: darkblue;
}
hr {
    border: 1px solid black;
    width: 100%;Z
}
```

6.5.1 Explanation of the CSS Code

Let us go over the CSS code step by step:

First, we have: `p.caption`. This targets paragraphs with the class `caption` (e.g., `<p class="caption">`).
- The `color: #777;` style sets the text color to a medium gray, providing a subtle, less prominent look.
- The `margin-top: 10px;` adds 10 pixels of space above the paragraph for better spacing between content.

Next, we have: `p code`. This targets inline `<code>` elements within a paragraph (e.g., `<p><code>code here</code></p>`).
- The `white-space: inherit;` ensures that the inline code adheres to the surrounding text's white-space behavior, allowing for consistent layout.

Then, there is: `pre`. This targets `<pre>` tags, which typically enclose preformat-

ted blocks of text like code snippets.
- The `word-break: normal;` prevents breaking words in unnatural places.
- The `word-wrap: normal;` avoids wrapping text to fit the container width, maintaining horizontal scrolling for long lines.

Within the `pre` block, we also have: `pre code`. This targets `<code>` elements specifically inside `<pre>` tags.
- The `white-space: inherit;` ensures that the `<code>` inside `<pre>` behaves consistently with the preformatted block's layout.

The next selector is: `.center`. This targets elements with the class `center`.
- The `text-align: center;` aligns the text or content of the element to the center of its container.

Moving on to: `h1, h2, h3, h4, h5, h6`. These styles are applied to all heading levels from `<h1>` to `<h6>`.
- The `font-family: "Times New Roman", Times, serif;` changes the heading text to a serif font for a traditional, formal look.
- The `color: darkblue;` gives the headings a distinct dark blue color for emphasis.

Finally, we have: `hr`. This targets horizontal rules (`<hr>`).
- The `border: 1px solid black;` sets the horizontal line to a solid black with a thickness of 1 pixel.
- The `width: 100%;` ensures the rule spans the entire width of its container, creating a full-length divider.

EXERCISE TIME!

Use custom LaTeX to add a unique visual element to your content (e.g., a shaded box or note). Play around with the settings and see what you like best. If you have more experience with CSS go ahead and try changing some items in your project.

Chapter 7

Customizing Output

In this chapter, we'll explore how to customize the output of your Bookdown project. Bookdown supports several output formats, such as HTML, PDF, and EPUB, and allows you to customize each format to match your project's needs. We'll cover choosing output formats, modifying appearance, and configuring output settings.

7.1 Choosing an Output Format

Bookdown provides multiple output formats that allow you to publish your document in various ways:

- **HTML**: Ideal for online documentation or sharing on the web.
- **PDF**: Useful for print-ready documents, especially for academic or professional reports.
- **EPUB**: E-book format, compatible with e-readers for mobile access.
- **Word**: Generates `.docx` files for easy editing or sharing in Microsoft Word.

To specify output formats, edit the `_output.yml` file in your project directory. Here's an example configuration:

```
bookdown::gitbook:
  css: "style.css"
  split_by: "chapter"

bookdown::pdf_book:
  latex_engine: xelatex
  citation_package: natbib
```

```
bookdown::epub_book: default

bookdown::word_document2: default
```

7.2 Available Output Options for Bookdown

Users are able to customize the `_output.yml` file to output only the formats that you want to use.

7.2.1 1. GitBook (`bookdown::gitbook`)

This is one of the most popular output formats, producing an HTML book with interactive features such as search and navigation. Infact this is how the book has been outputted.

Example:

```
output:
  bookdown::gitbook:
    css: style.css  # Optional: Custom CSS for styling
    config:
      toc:
        collapse: section  # Controls the collapsing of the table of contents
```

7.2.2 2. PDF Book (`bookdown::pdf_book`)

Generates a PDF version of your book using LaTeX. This format is often used for printed versions.

Example:

```
output:
  bookdown::pdf_book:
    latex_engine: xelatex  # Specify the LaTeX engine to use (e.g., pdflatex, xelatex,
    includes:
      in_header: preamble.tex  # Include additional LaTeX setup files
    citation_package: natbib  # Citation package to use
```

7.2.3 3. EPUB Book (`bookdown::epub_book`)

Creates an EPUB file, which is a format commonly used for eBooks.

Example:

```yaml
output:
  bookdown::epub_book:
    toc: yes  # Include table of contents
    css: style.css  # Optional: Custom CSS for EPUB styling
```

7.2.4 4. HTML Document (`rmarkdown::html_document`)

Produces a single HTML document, rather than a book format.

Example:

```yaml
output:
  rmarkdown::html_document:
    toc: true  # Include table of contents
    toc_depth: 3  # Depth of the table of contents
    number_sections: true  # Number sections in the output
```

7.2.5 5. Word Document (`bookdown::word_document2`)

Generates a Word document (`.docx`) version of your book, useful for sharing with editors.

Example:

```yaml
output:
  bookdown::word_document2:
    toc: yes  # Include table of contents
    toc_depth: 2  # Depth of table of contents
```

7.2.6 6. Tufte Handout (`bookdown::tufte_html_book` / `bookdown::tufte_pdf_book`)

Generates a book in Tufte style, known for its distinctive design, which is suitable for visually focused content.

Example:

```yaml
output:
  bookdown::tufte_html_book: default
```

For PDF:

```yaml
output:
  bookdown::tufte_pdf_book: default
```

7.2.7 7. HTML with Bookdown-Specific Features (`bookdown::html_document2`)

Similar to `rmarkdown::html_document` but includes additional cross-referencing features from Bookdown.

Example:

```
output:
  bookdown::html_document2:
    toc: true
    number_sections: true
```

7.2.8 8. GitHub Document (`rmarkdown::github_document`)

Produces a document suitable for rendering on GitHub, best for single page documents.

Example:

```
output:
  rmarkdown::github_document:
    toc: true  # Include table of contents
```

7.3 Customizing HTML Output

To customize the HTML format, Bookdown offers the `bookdown::gitbook` and `bookdown::html_document2` options:

- **GitBook**: The default HTML style, which includes a side navigation bar and a search function. This format is ideal for online documentation.
- **HTML Document**: A simpler format without the sidebar, suitable for single-page reports.

You can adjust HTML settings in `_output.yml`:

```
bookdown::gitbook:
  css: "style.css"
  config:
    toc:
      collapse: section
    download: ["pdf", "epub"]
```

7.4 Customizing PDF Output

To generate a high-quality PDF, you'll need to install a LaTeX distribution like TinyTeX.

In `_output.yml`, you can specify options to control PDF formatting:

```
bookdown::pdf_book:
  latex_engine: xelatex
  citation_package: natbib
```

- **`latex_engine`**: Specifies the LaTeX engine (e.g., `xelatex`, `pdflatex`). Using `xelatex` improves font compatibility.
- **`citation_package`**: Choose between `natbib` or `biblatex` for handling citations.

7.4.1 Document Size

To customize the size of your PDF document, you can modify the `geometry` option in the LaTeX preamble. For example, to set the paper size to A4 and customize the margins:

```
output:
  bookdown::pdf_book:
    includes:
      in_header: preamble.tex
```

In the `preamble.tex` file, add:

```
\usepackage[paperwidth=5.5in, paperheight=8.5in, margin=0.75in]{geometry}
```

This sets the document size to A4 and the margins to 1 inch.

7.4.2 Document Type

You can change the document type by modifying the LaTeX class used in the `preamble.tex`. For example, use the `article` class instead of the default `book` class:

```
\documentclass{article}
```

This is useful for a more compact layout, like that of a research paper or report.

7.4.3 LaTeX Engine

You can specify the LaTeX engine used to compile your PDF. Bookdown supports different engines, such as `pdflatex`, `xelatex`, and `lualatex`.

Example:

```
output:
  bookdown::pdf_book:
    latex_engine: xelatex
```

Using `xelatex` or `lualatex` provides better font support, especially for special characters or non-Latin scripts.

7.4.4 Custom LaTeX Packages

Include custom LaTeX packages in the `preamble.tex` file to extend the functionality of your PDF output.

Example:

```
\usepackage{amsmath}
```

This includes the `amsmath` package for advanced mathematical formatting. You can also use `\newcommand` to define custom commands used throughout your document.

7.4.5 Page Layout Customization

To customize the page layout, use packages like `fancyhdr` to modify headers and footers.

Example:

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[LE,RO]{Your Book Title}
\fancyfoot[CE,CO]{\thepage}
```

This sets custom headers and footers, including the book title and page numbers.

7.5 Customizing EPUB Output

To create an EPUB e-book, use `bookdown::epub_book` in `_output.yml`. Bookdown handles most EPUB formatting automatically, but you can make some modifications:

```
bookdown::epub_book:
  stylesheet: "style.css"
  cover_image: "images/cover.jpg"
  toc: yes
```

This configuration adds a cover image, applies the CSS stylesheet, and includes a table of contents.

7.6 Specifying Global Settings in `_bookdown.yml`

The `_bookdown.yml` file allows you to set global configurations, such as the order of chapters, the naming convention for output files, and the label format for figures and tables. Here's an example:

```
book_filename: "my-book"
rmd_files: ["index.Rmd", "01-introduction.Rmd", "02-writing-structuring-content.Rmd", "03-custom
language:
  label:
    fig: "Figure "
    tab: "Table "
delete_merged_file: true
```

- **`book_filename`**: Sets the base filename for output files.
- **`rmd_files`**: Specifies the order of chapters.
- **`language.label`**: Customizes labels for figures and tables.
- **`delete_merged_file`**: Deletes intermediary files after rendering, keeping the directory clean.

7.7 Example Output

To render all formats simultaneously, you can use the `render_book()` function in the R console:

```
bookdown::render_book("index.Rmd", output_format = "all")
```

This command generates HTML, PDF, and EPUB files as specified in `_output.yml`.

EXERCISE TIME!

Modify your Bookdown project's _output.yml to customize the output format. Try changing the appearance of the HTML and PDF outputs. Write a paragraph explaining how the customizations improved the document's look and usability.

When you're ready go ahead and render your book into HTML, PDF, and EPUB formats. Publish it on GitHub Pages or share it with someone else. Write a paragraph reflecting on what you learned about the publishing process and any challenges you encountered.

Chapter 8

LaTeX Distributions

To render PDF outputs with Bookdown, you need to install a LaTeX distribution. Below is a list of popular options, categorized by operating system and user preferences:

8.1 Recommended LaTeX Distribution

8.1.1 1. TinyTeX (Recommended)

- **Description**: A lightweight, cross-platform LaTeX distribution designed to work seamlessly with R and Bookdown.
- **Installation**: Run the following commands in R:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

- **Advantages**:
  - Minimal installation size.
  - Automatically installs missing packages when rendering.
- **Website**: TinyTeX Documentation

8.2 Additional LaTeX Distributions

8.2.1 2. TeX Live

- **Description**: A comprehensive LaTeX distribution suitable for Linux and cross-platform users.

- **Installation**:
  - **Linux**:

    ```
    sudo apt-get install texlive-full
    ```

  - **macOS and Windows**: Download from TeX Live.
- **Advantages**:
  - Full-featured with a vast collection of LaTeX packages.
  - Stable and widely used.
- **Website**: TeX Live Documentation

8.2.2 3. MikTeX

- **Description**: A user-friendly LaTeX distribution popular among Windows users.
- **Installation**: Download and install from MikTeX.
- **Advantages**:
  - On-demand installation of missing packages.
  - Easy-to-use package manager.
- **Website**: MikTeX Documentation

8.2.3 4. MacTeX (for macOS)

- **Description**: A macOS-specific version of TeX Live with additional tools for macOS users.
- **Installation**: Download and install from MacTeX.
- **Advantages**:
  - Tailored for macOS with GUI tools like TeXShop.
  - Includes a full TeX Live distribution.
- **Website**: MacTeX Documentation

8.2.4 5. ProTeXt (for Windows)

- **Description**: A Windows-specific distribution that combines MikTeX with a user-friendly installer.
- **Installation**: Download and install from ProTeXt.
- **Advantages**:
  - Streamlined setup for beginners.
  - Integrates LaTeX editors like TeXworks.
- **Website**: ProTeXt Documentation

Choose the distribution that best fits your operating system and needs. For most users, TinyTeX is the easiest to install and manage, especially if you're using R and Bookdown.

Chapter 9

Advanced Text Formatting Options

Markdown and it's enchanged version Pandoc included with Bookdown allow for a wide variety of text formatting, making it easy to structure documents and highlight important content. Below, you'll find a comprehensive guide to formatting options you can use in Markdown and Pandoc.

9.1 1. Headers

Headers are used to create section headings, and the number of `#` symbols represents the level of the header.

- `# Header 1`
- `## Header 2`
- `### Header 3`
- `#### Header 4`
- `##### Header 5`
- `###### Header 6`

9.2 2. Emphasis

- **Bold**: `**bold text**` or `__bold text__`
- *Italics*: `*italic text*` or `_italic text_`
- ***Bold and Italic***: `***bold and italic***` or `___bold and italic___`

9.3 3. Strikethrough

- ~~Strikethrough~~: `~~strikethrough text~~`

9.4 4. Lists

9.4.1 4.1 Bullet Lists

- `-` or `*` creates a bullet list.

```
- First item
- Second item
  - Subitem
```

9.4.2 4.2 Numbered Lists

- `1.` creates a numbered list.

```
1. First item
2. Second item
   1. Subitem
```

9.4.3 4.3 Task Lists

- `[ ]` creates a task list.

```
- [x] Completed task
- [ ] Incomplete task
```

It also allows the box to be clickable.

☒ Completed task

☒ Incomplete task

9.5 5. Blockquotes

- Blockquote: Use `>` for blockquotes.

```
> This is a blockquote.
```

9.6 > This is a blockquote.

9.7 6. Code

9.7.1 6.1 Inline Code

- Wrap code with backticks: `` `code` ``

9.7.2 6.2 Code Blocks

- Use triple backticks () `for code blocks`.markdown “‘python print(“Hello, World!”) “‘ “‘

9.8 7. Horizontal Rule

- Use three or more `-`, `*`, or `_` to create a horizontal line.

```
---
```

9.9 8. Links

- Inline link: `[link text](URL)`
- Reference link:

```
[link text][reference]

[reference]: http://example.com
```

9.10 9. Images

- Inline image: `![alt text](image-url)`
- Reference image:

```
![alt text][image-ref]

[image-ref]: http://example.com/image.png
```

9.11 10. Tables

Tables can be created using pipes (|) and hyphens (-).

```
/ Header 1 / Header 2 /
/----------/----------/
/ Cell 1   / Cell 2   /
/ Cell 3   / Cell 4   /
```

9.12 11. Footnotes

- Footnote syntax: `Here is a footnote reference[^1].`
- Define the footnote elsewhere: `[^1]: This is the footnote content.`

9.13 12. Definition Lists (Pandoc)

Pandoc extends Markdown by supporting definition lists.

```
Term 1
: Definition 1

Term 2
: Definition 2
```

9.14 13. Math

9.14.1 13.1 Inline Math

- Use single dollar signs: `$E = mc^2$`

9.14.2 13.2 Block Math

- Use double dollar signs:

  ```
  $$
  E = mc^2
  $$
  ```

9.14.3 13.3 LaTeX Environment

```
\[
E = mc^2
\]
```

9.15 14. Custom Blocks (Pandoc)

Pandoc supports custom blocks for HTML and LaTeX conversion.

```
::: {.custom-class}
Custom content here.
:::
```

9.16 15. HTML Tags

You can directly use HTML tags for additional formatting.

```
<span style="color: red;">Red text</span>
```

9.17 16. Metadata Blocks (Pandoc)

You can add metadata at the beginning of the document.

```
---
title: "Document Title"
author: "Author Name"
date: "2024-11-17"
---
```

9.18 17. Line Breaks

- Add two spaces at the end of a line to create a line break.
- Alternatively, use `<br>` for a line break in HTML.

9.19 18. Escaping Characters

- Use a backslash (\) before a character to escape it.

  ```
  \*Not italic\*
  ```

9.20 19. Highlight (Pandoc)

To highlight text, use `==highlighted text==`. This is only available in ==Pandoc.==

```
This is ==highlighted== text.
```

9.21 20. Superscript and Subscript

9.21.1 20.1 Superscript

- `X^2^` becomes X^2.

9.21.2 20.2 Subscript

- `H~2~O` becomes H O.

Chapter 10

Example Document: Union Earnings Analysis

Also a zine!

Download a printable zine version here!

10.1 What's a Union?

So, what exactly is a union, and why should you care? Basically, unions are groups of workers who come together to protect their rights and fight for better working conditions. Whether it's fair wages, benefits, or safer workplaces, unions have got your back.

In 2024, with inflation and the cost of living getting out of control, unions are stepping up big time. They're out here negotiating fair pay and benefits so we don't get left in the dust. This zine will show you how unions are making a difference right now and why they matter to you, especially your paycheck, and your future.

10.2 Quick History

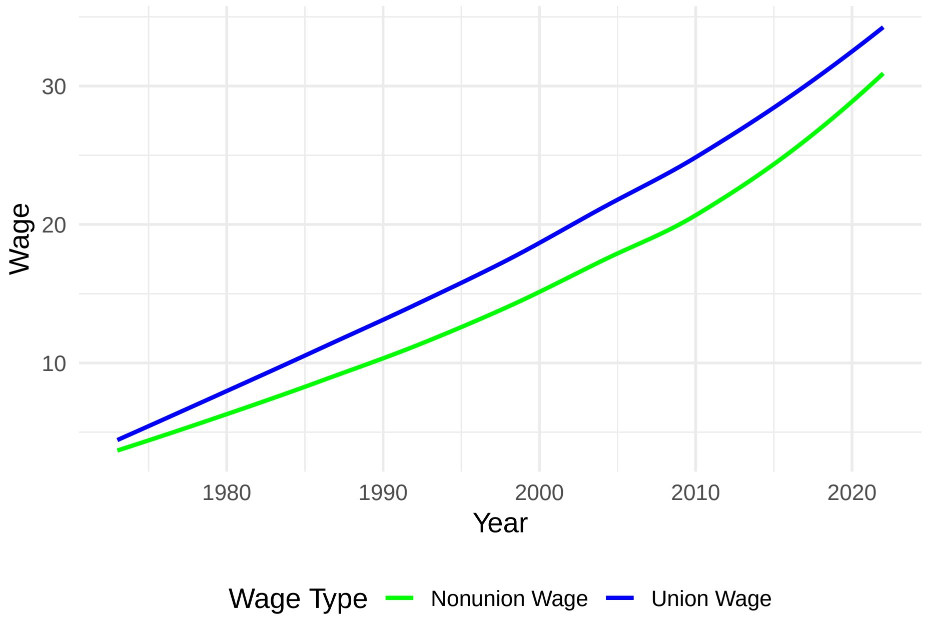
- The story of unions in the United States began in the early 19th century. The first organized unions were formed by skilled tradespeople like shoemakers and carpenters. These early unions set the groundwork for collective bargaining by demanding better wages and shorter working hours.
- By the 1880s, major unions like the Knights of Labor emerged, representing workers from various industries and advocating for an eight-hour

workday. The American Federation of Labor (AFL), founded in 1886, focused on improving the wages and working conditions of its members through direct negotiations with employers.

- These first unions showed that workers could stand up to powerful industrialists and win meaningful victories. Their legacy paved the way for the labor rights we enjoy today, proving that when workers unite, they can achieve lasting change.

10.3 Why a Union?

Ever wonder if being in a union actually makes a difference in your paycheck? Spoiler alert: it totally does. As you can see in the plot Union Wages consistently track better than non-union wages. That's money left on the table if you aren't in a Union!



But it's not just about the money. Unions also fight for safer working conditions, better benefits like healthcare and paid time off, and they give workers a voice. Being part of a union means having real power to push back against unfair treatment and make sure you're getting what you need to thrive—not just survive. Lets focus just on earnings though.

10.4 Recent Union Wins

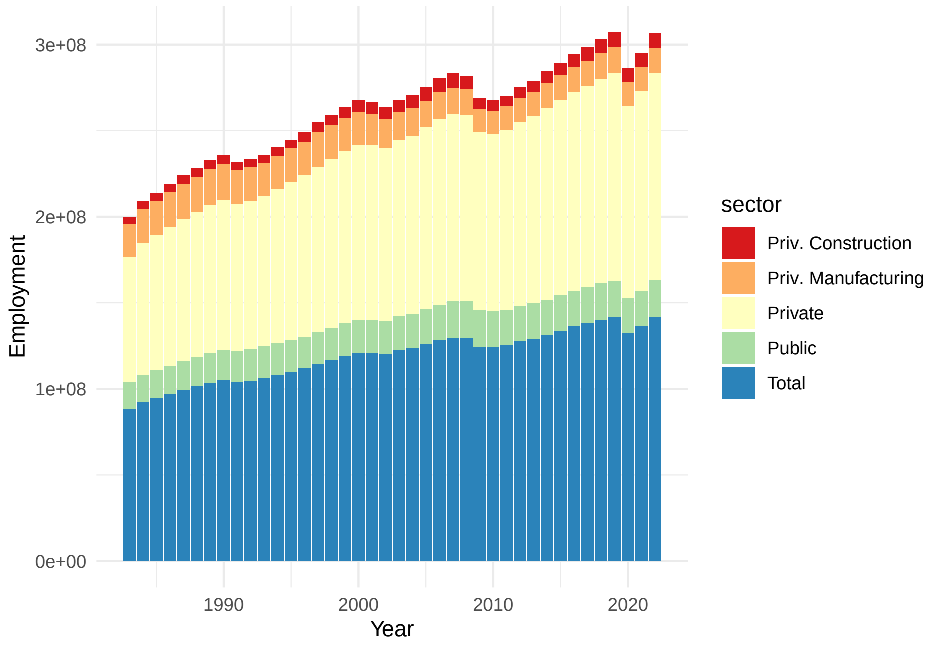
- **Restoration of Salary Steps for California State University (CSU) Workers (2024):**

Teamsters Local 2010 secured an agreement with CSU, reinstating a salary step system that had been absent for decades. The agreement included an immediate 5% salary increase retroactive to July 1, 2023, and all members reaching their target salary step by July 2025.[?]

- **United Auto Workers (UAW) Strike Achievements (2023):** In fall 2023, the UAW strike against major automakers resulted in substantial pay raises and improved working conditions. It was one of the most significant contract victories since the sit-down strikes of the 1930s. [?]
- **Unionization of Volkswagen's Chattanooga Plant (2024):** The UAW successfully unionized Volkswagen's Chattanooga, Tennessee, plant in April 2024. This was the first successful unionization at a foreign manufacturer's U.S. plant in the southern states. [?]
- **Unionization of Blue Bird Bus Factory Workers (2023):** The United Steelworkers won an election at a Blue Bird bus factory in Georgia, unionizing nearly 1,500 predominantly Black workers in 2023.

10.5 Unions are Growing!

As the economy changes so too does Union membership in certain job sectors. What your take away should be is that no matter the type of job you have, it can be unionized!



So join a union or start forming one of your own!

10.6 Getting Involved!

- **Join a Union**: Check if there is an existing union that represents workers in your industry. The AFL-CIO website (https://aflcio.org) is a great starting point to find affiliated unions.
- **Form a Union**: If there isn't one, consider forming your own. The National Labor Relations Board (https://www.nlrb.gov) provides resources to help workers understand their rights and the steps involved in unionizing.
- **Connect with Organizers**: The United Electrical, Radio, and Machine Workers of America (UE) has guides on forming a union and can connect you with experienced organizers (https://www.ueunion.org).
- **Get Educated**: Websites like Labor Notes (https://labornotes.org) offer valuable information about union organizing, labor rights, and how to take action.
- **Reach Out to Local Chapters**: Look for local worker centers or community organizations that support unionization efforts. They can be a great source of advice and resources.
- **Communication Workers of America (CWA)** – CWA represents tech and telecommunications workers and has been actively organizing workers in the tech sector, including campaigns at Google and other tech companies. More information can be found here: https://cwa-union.org.

Bibliography

Yihui Xie. *Bookdown: Authoring Books and Technical Documents with R Markdown.* Chapman; Hall/CRC, 2nd edition, 2024. URL https://bookdown.org/yihui/bookdown/. ISBN 9780367142568.