Merck-Data Mine Documentation

Merck and Data Mine Corporate Partnership Team

2020-09-23

Contents

Chapter 1

Introduction

This book will serve as tutorial based documentation for the Merck - Data Mine Coporate Partnership team for the 2020-2021 academic year.

1.1 How to contribute

Contributing to this book is simple:

1.1.1 Larger changes or additions

If you have larger changes or additions you'd like to make to the book, the easiest way is to edit the contents of the book on your local machine.

1.1.1.1 Using `git` in the terminal

1. Setup `git` following the directions here.
2. Start by opening up a terminal and configuring `git` to work with GitHub.
3. Navigate to the directory in which you would like to clone the-examples-book repository. For example, if I wanted to clone the repository in my `~/projects` folder, I'd first execute: `cd ~/projects`.
4. Clone the repository. In this example, let's assume I've cloned the repository into my `~/projects` folder.
5. Navigate into the project folder:

```
cd ~/projects/the-examples-book
```

6. At this point in time your current branch should be the `master` branch. You can verify by running:

```
git branch
```

Note: The highlighted branch starting with "*" is the current branch.

or if you'd like just the name of the branch:

```
git rev-parse --abbrev-ref HEAD
```

7. Create a new branch with whatever name you'd like, and check that branch out. For example, `fix-spelling-errors-01`.
8. Open up RStudio. In the "Files" tab in RStudio, navigate to the repository. In this example, we would navigate to `/Users/kamstut/Documents/GitHub/the-examples-b` Click on the "More" dropdown and select "Set As Working Directory".
9. If you do not already have `renv` installed, install it by running the following commands in the console:

```
install.packages("renv")
```

10. Restore the environment by running the following commands in the console:

```
renv::restore()
```

11. In order to compile this book, you must have LaTeX installed. The easiest way to accomplish this is to run the following in the R console:

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

12. In addition, make sure to install both `pandoc` and `pandoc-citeproc` by following the instructions here.
13. Modify the `.Rmd` files to your liking.
14. Click the "Knit" button to compile the book. The resulting "book" is within the "docs" folder.

Important note: If at any point in time you receive an error saying something similar to "there is no package called `my_package`, simply install the missing package, and try to knit again:

```
install.packages("my_package")
library(my_package)
```

15. To test the book out, navigate to the "docs" folder and open the `index.html` in the browser of your choice.
16. When you are happy with the modifications you've made, commit your changes to the repository.
17. You can continue to make modifications and commit your changes locally. When you are ready, you can push your branch to the remote repository (github.com).
18. At this point in time, you can confirm that the branch has been succesfully pushed to github.com by navigating to the repository on github, and click on the "branches" tab:

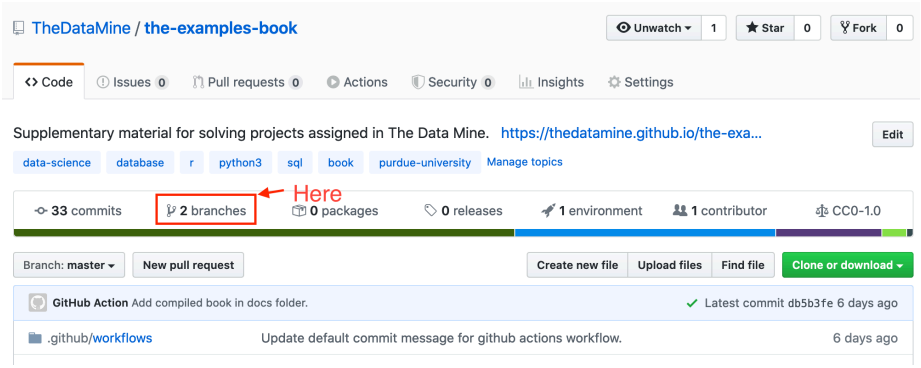


Figure 1.1:

19. Next, create a pull request. Note that a "Pull Request" is a GitHub-specific concept. You cannot create a pull request using `git`. Navigate to the repository https://github.com/thedatamine/the-examples-book, and you should see a message asking if you'd like to create a pull request:

20. Leave a detailed comment about what you've modified or added to the book. You can click on "Preview" to see what your comment will look like. GitHub's markdown applies here. Once satisfied, click "Create pull request".

21. At this point in time, the repository owners will receive a notification and will check and potentially merge the changes into the `master` branch.

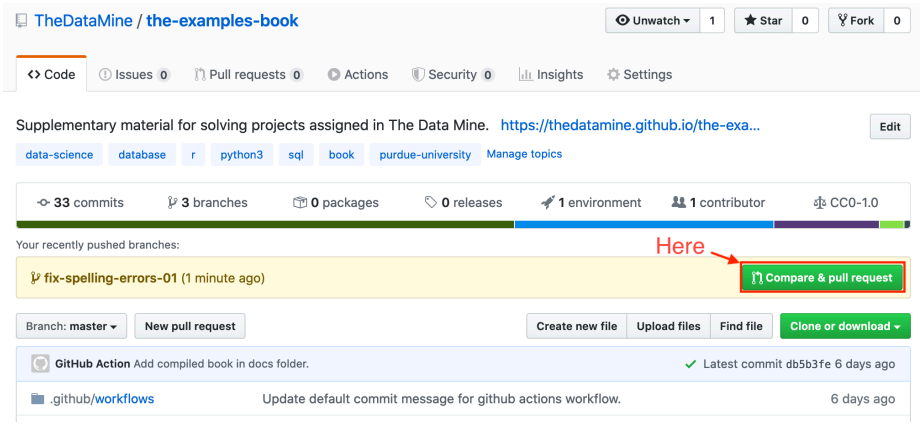
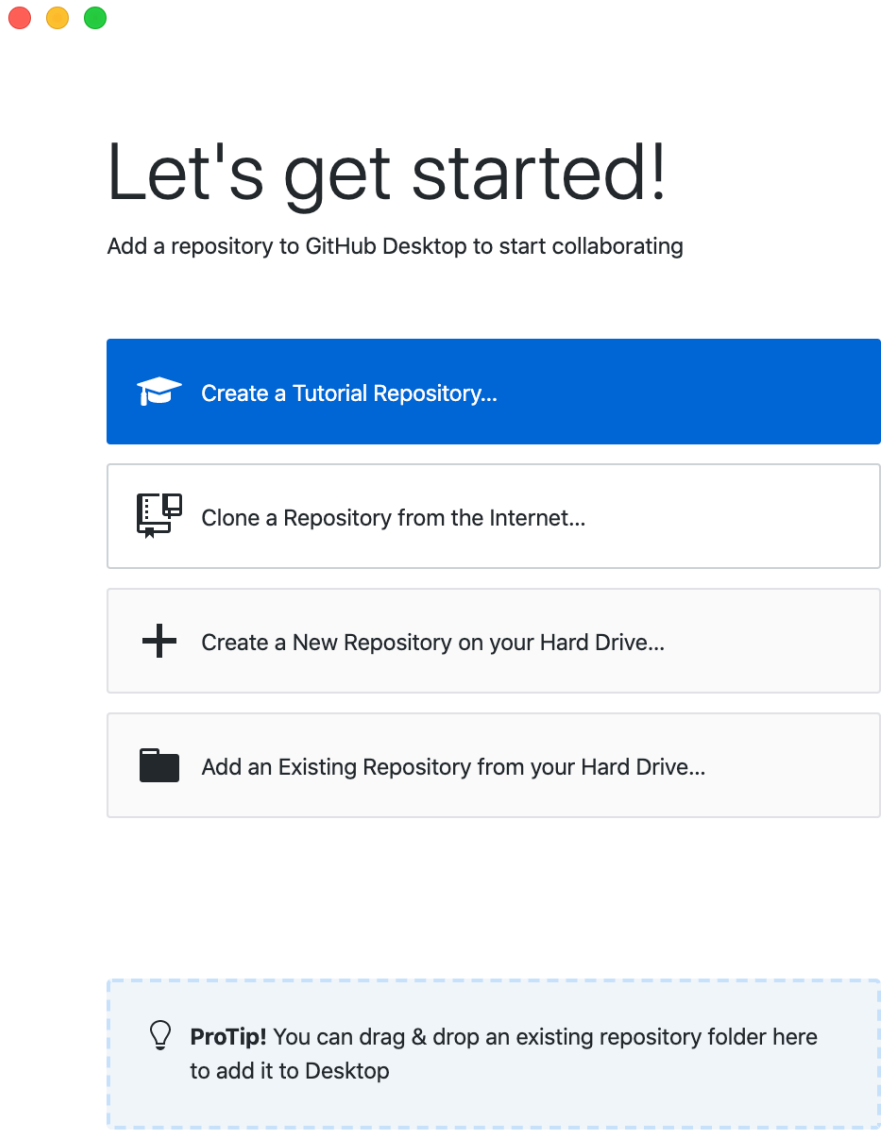


Figure 1.2:

1.1.1.2 Using GitHub Desktop

1. Setup GitHub Desktop following the directions here.
2. When you are presented with the following screen, select "Clone a Repository from the Internet…":



3. Click on the "URL" tab:

4. In the first field, enter "TheDataMine/the-examples-book". This is the repository for this book.
5. In the second field, enter the location in which you'd like the repository to be cloned to. In this example, the repository will be cloned into `/Users/kamstut/Documents/GitHub`. The result will be a new folder

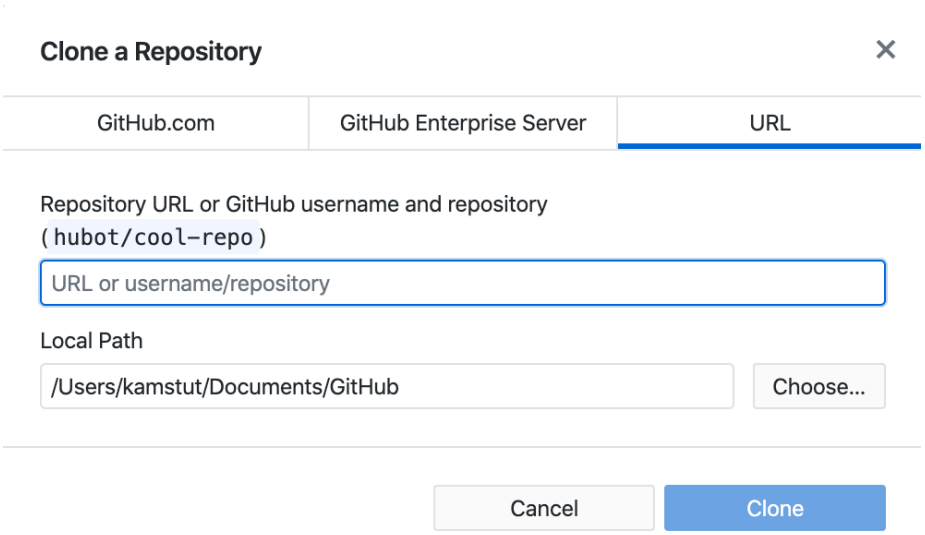


Figure 1.3:

called `the-examples-book` in `/Users/kamstut/Documents/GitHub`.
6. Click "Clone".
7. Upon completion, you will be presented with a screen similar to this:

8. At this point in time, your current branch will be the `master` branch. Create a new branch with whatever name you'd like. For example, `fix-spelling-errors-01`.
9. Open up RStudio. In the "Files" tab in RStudio, navigate to the repository. In this example, we would navigate to `/Users/kamstut/Documents/GitHub/the-examples-b` Click on the "More" dropdown and select "Set As Working Directory".
10. If you do not already have `renv` installed, install it by running the following commands in the console:

```
install.packages("renv")
```

11. Restore the environment by running the following commands in the console:

```
renv::restore()
```

12. In order to compile this book, you must have LaTeX installed. The easiest way to accomplish this is to run the following in the R console:

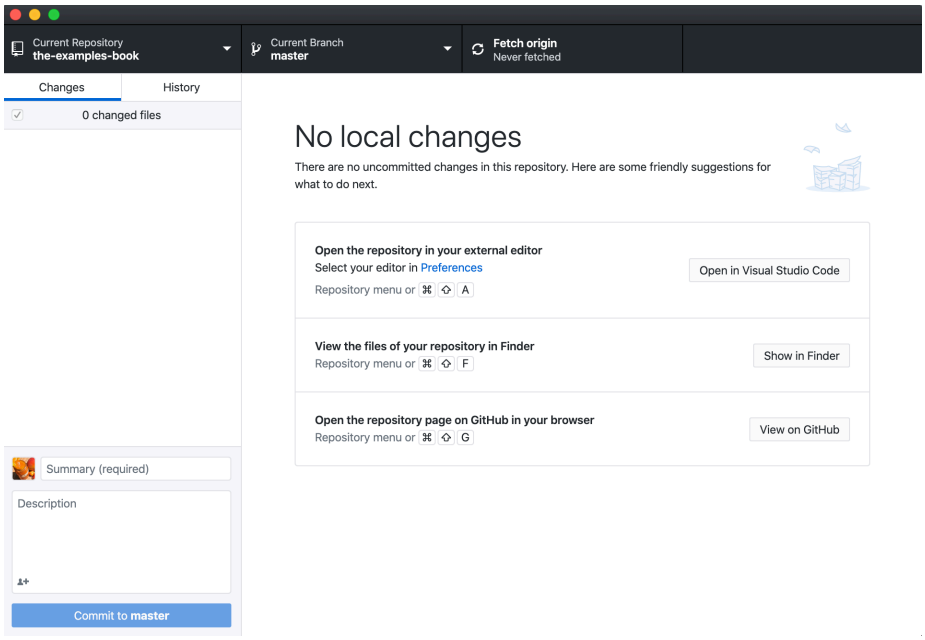


Figure 1.4:

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

13. In addition, make sure to install both `pandoc` and `pandoc-citeproc` by following the instructions here.

14. Modify the `.Rmd` files to your liking.

15. Click the "Knit" button to compile the book. The resulting "book" is within the "docs" folder.

Important note: If at any point in time you receive an error saying something similar to "there is no package called `my_package`, simply install the missing package, and try to knit again:

```
install.packages("my_package")
library(my_package)
```

16. To test the book out, navigate to the "docs" folder and open the `index.html` in the browser of your choice.

17. When you are happy with the modifications you've made, commit your changes to the repository.
18. You can continue to make modifications and commit your changes locally. When you are ready, you can publish your branch:

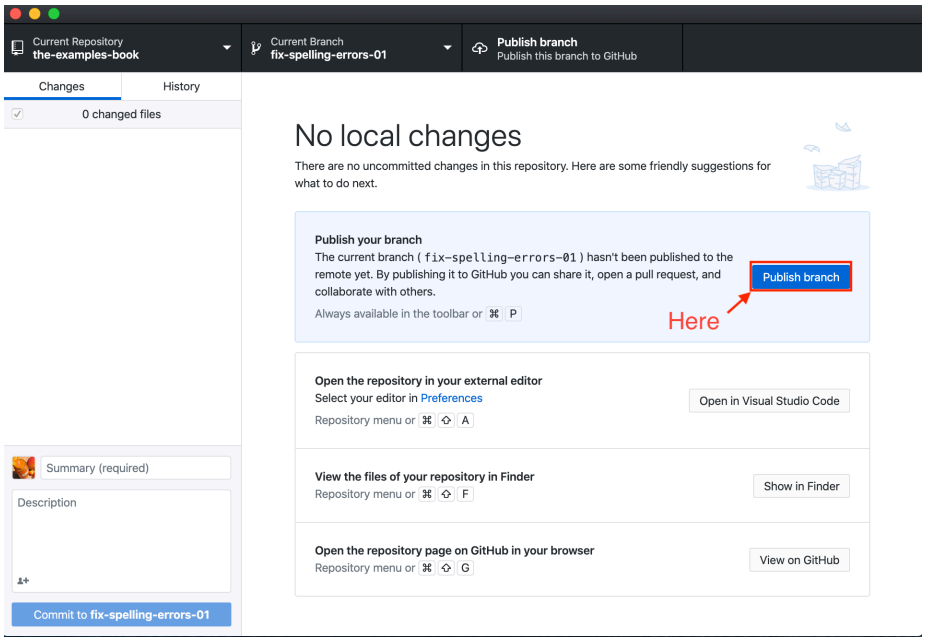


Figure 1.5:

19. Upon publishing your branch, within GitHub Desktop, you'll be presented with the option to create a pull request:

20. At this point in time, the repository owners will receive a notification and will check and potentially merge the changes into the `master` branch. © 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

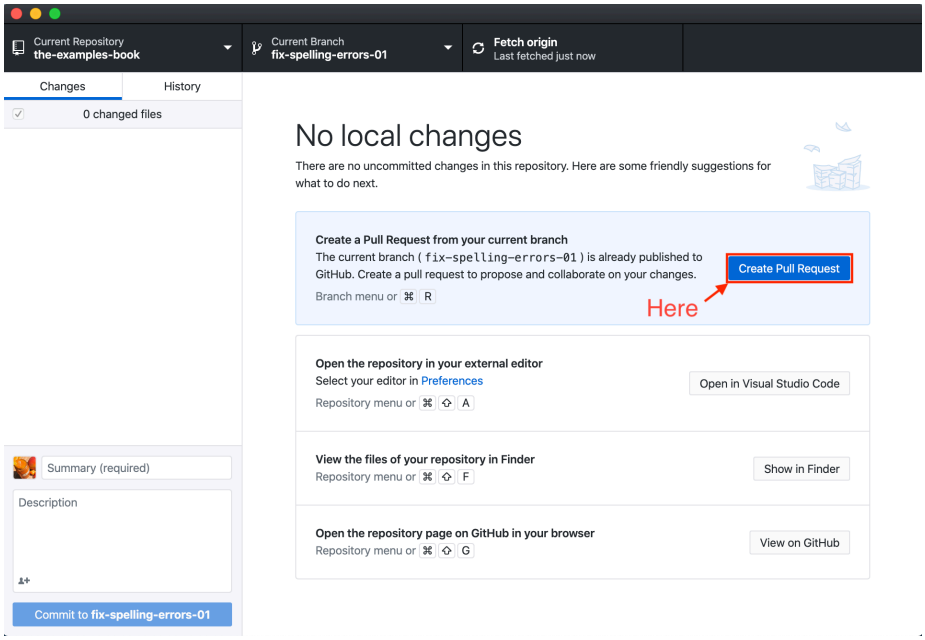

Current Repository
the-examples-book
Current Branch
fix-spelling-errors-01
Fetch origin
Last fetched just now
Changes
History
0 changed files
No local changes
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.
Create a Pull Request from your current branch
The current branch (fix-spelling-errors-01) is already published to GitHub. Create a pull request to propose and collaborate on your changes.
Branch menu or ⌘ R
Create Pull Request
Here
Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⌥ A
Open in Visual Studio Code
View the files of your repository in Finder
Repository menu or ⌘ ⌥ F
Show in Finder
Open the repository page on GitHub in your browser
Repository menu or ⌘ ⌥ G
View on GitHub
Summary (required)
Description
Commit to fix-spelling-errors-01

Figure 1.6:

Chapter 2

Tips for Writing Great Documentation and Walkthroughs

2.1 Find a Good Topic

While writing documentation for a project, it would nearly impossible to include every piece work completed or every line of code written. This makes it important to pick important topics to write about. The goal is to include as much specificty as possible while also remembering the project as its entirety.

Here are a few helpful concepts to consider when choosing a topic:

1. Step by step guides – perfect for readers to learn quickly and implement in their own projects
2. In depth discussions of a specific topic – great for readers who are looking for deeper knowledge in a topic
3. Numbered lists of useful facts about a common topic – lightweight readings that readers can consume in bits and pieces.

2.2 Make Goals and Audience Clear

For these writings, our team will have a dual purpose of writing them.

1. Record the work we have completed

2. Create a centralized location for tutorial based learning

The audience to consider is future team members of the Merck-Data Mine Corporate Partnership and Merck scientists looking to read and learn from our work.

Remember to keep the audience and goal of our documentation in mind as writing your entries.This will help the book keep continuiuty and give readers the best chance to get exposed to the work that we have completed.

2.3 Have a Begining, Middle, and End

It is important to have an introduction, body, and conclusion while writing the documentation. This helps with fluidity within each document and allows for easier comprehension.

2.3.1 Introduction

The introduction should encourage the reader to continue reading. Start with information about what will be covered in the read and how it applies to the project. Try to keep the introduction less technical so readers aren't discouraged by the complexity of the document.

2.3.2 Body

The body is where you elaborate on all that you discussed in the introduction. Provide depth and instruction while still relating to the project as a whole. Use headings, photos, numbered lists, bullet points, and formatting to help provide small bits of information at a time. This is where you can facilitate a technical discussion with code.

2.3.3 Conclusion

Always finish the read with a conclusion, providing assurance of what was just learned and include possible resources for more information (i.e. academic papers, blog posts, youtube videos). It is also appropriate to give the reader a domain in which to use the skills they have learned from your documentation.

2.4 Getting Feedback and Iterate

Everyone on the team is encouraged to follow the documents that are added to this book. Read through them and provide helpful feedback to your teammate on how to improve.

Some common things to look out for:

1. Formatting – Does the document flow properly? Are there enough images, code, headers, etc…?
2. Formality – Is the document written well? Is the language approiate?
3. Goal and Audience – Does the document relate to the goals and target audiences of this book?
4. Attention – Was the reading interesting? Is there opportunity to learn from the read?

2.5 Practice, Practice, Practice

Writing the entries in this book will undoubtably get better over time. You are welcome to write as many entries as you'd like. They can be simple or deep, long or short, imformational or technical, etc. As long as the information in this book is informative and relevant to the projects we hope to complete, it is encouraged for all members of the team to write what they want!

Chapter 3

Documentation Example - The Basics of R Markdown

3.1 Overview of R Markdown

R Markdown is a file format for making dynamic documents with R. An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code, text, images, headers, and more.

In this walkthrough, we'll discuss some of the functionality of R Markdown Documents and how to add images, code, and other features to the document.

Throughout this tutorial, we'll be reviewing the contents of the R Markdown cheat sheet that has most important information for writing in Markdown. The cheat sheet can be found at https://rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf.

3.2 Workflow

One of the great features of Markdown files is that they can be rendered to PDF files, Word documents, HTML content, and more. This allows the writer to easily write in R and export the document how they see best.

Take a look at a common lifecylce of R Markdown documents in th following picture:

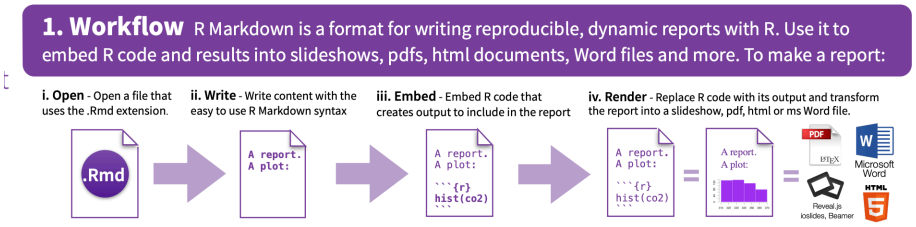


Figure 3.1:

3.3 Opening a New File

Writing R Markdown files is easiest within R Studio. Navigate to File > New File > R Markdown to create a new file.

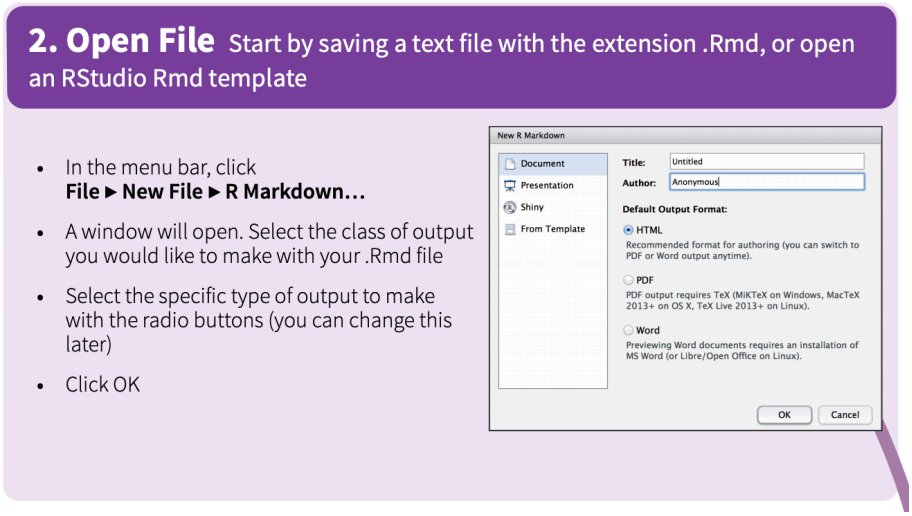


Figure 3.2:

3.4 Helpful Syntax

Take a look through helpful sytax in this photo:

Here are some examples to vie. Take a look at the raw R Markdown to view the syntax in pratice.

italics **bold** ~~strikethrough~~

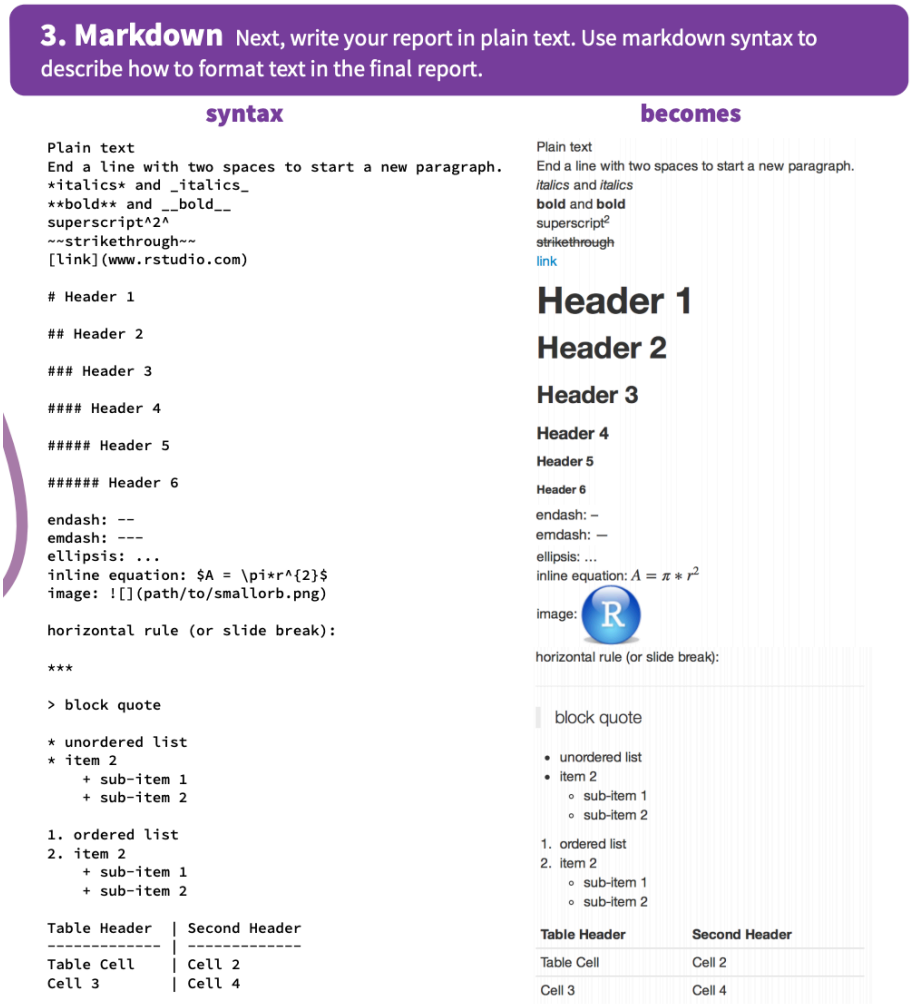

3. Markdown Next, write your report in plain text. Use markdown syntax to describe how to format text in the final report.
syntax
becomes
Plain text
End a line with two spaces to start a new paragraph.
italics and _italics_
bold and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)
Header 1
Header 2
Header 3
Header 4
Header 5
Header 6
endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image:
horizontal rule (or slide break):
***
> block quote
* unordered list
* item 2
+ sub-item 1
+ sub-item 2
1. ordered list
2. item 2
+ sub-item 1
+ sub-item 2
Table Header | Second Header
Table Cell | Cell 2
Cell 3 | Cell 4
Plain text
End a line with two spaces to start a new paragraph.
italics and italics
bold and bold
superscript2
strikethrough
link
Header 1
Header 2
Header 3
Header 4
Header 5
Header 6
endash: –
emdash: —
ellipsis: …
inline equation: A = π ∗ r2
image:
horizontal rule (or slide break):
block quote
unordered list
item 2
sub-item 1
sub-item 2
ordered list
item 2
sub-item 1
sub-item 2
Table Header
Second Header
Table Cell
Cell 2
Cell 3
Cell 4

Figure 3.3:

> Block Quote

- Bullets
- Bullets
- subitem

1. Lists
2. Lists

- subitem

3.5 Embed Code

One of the best features of R Markdown is the ability to add code to your document. To add a code snippet, click on the green *insert* button in your R Studio tool bar and choose which language you would like to use!

```
print("This is an R code snippet")
```

```
## [1] "This is an R code snippet"
```

```
print("This is a python code snippet with eval=FALSE")
```

```
echo this a bash code snippet
```

```
## this a bash code snippet
```

There are also many options for your code snippets. Take a look:

3.6 Wrapping (or knitting) it Up

To knit your Markdown file you click the blue *knit* button in your R Studio toolbar. You can choose which file format you would like to knit to as well!

For the purposes of our Merck-Data Mine documentation book, however, we will not have to knit anything because we are placing the individual documents in one bookdown book. To learn more about bookdown take a look at https://bookdown.org/yihui/bookdown/introduction.html for more information.

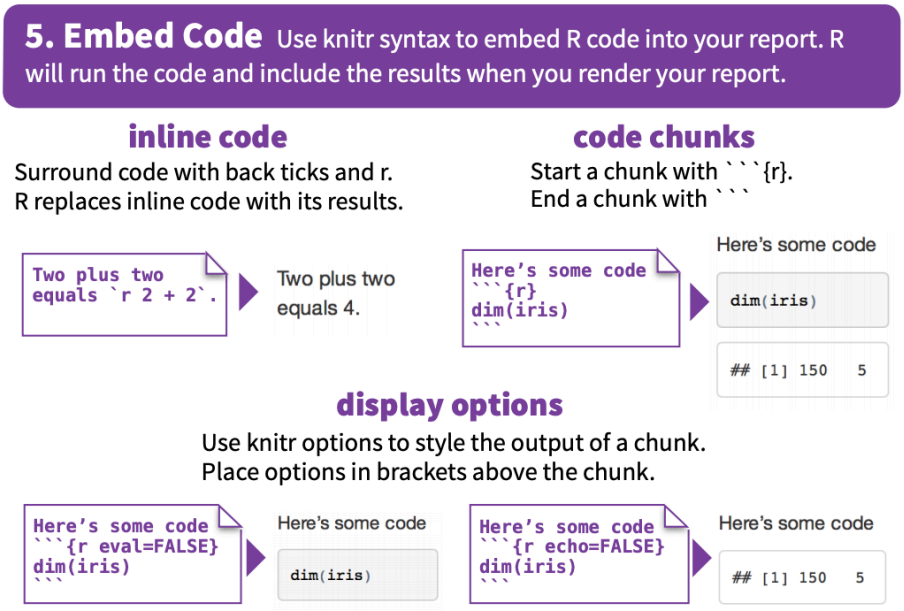


Figure 3.4:

option	default	effect
eval	TRUE	Whether to evaluate the code and include its results
echo	TRUE	Whether to display code along with its results
warning	TRUE	Whether to display warnings
error	FALSE	Whether to display errors
message	TRUE	Whether to display messages
tidy	FALSE	Whether to reformat code in a tidy way when displaying it
results	"markup"	"markup", "asis", "hold", or "hide"
cache	FALSE	Whether to cache results for future renders
comment	"##"	Comment character to preface results with
fig.width	7	Width in inches for plots created in chunk
fig.height	7	Height in inches for plots created in chunk

For more details visit yihui.name/knitr/

Figure 3.5:

Chapter 4

Documentation Template

4.1 Introduction

This section will give an overview of what was accomplished during the previous sprint.

4.2 Code

This section is where important code can be documented and commented on. Some teams do not need to include all their code here as this would get excessive. However, important functions or classes should be documented and discussed here.

4.3 Flow Diagrams / Visualizations

This is where teams should showcase flow diagrams of functions, data pipeline, etc. This might also be a good place for visualization/screenshots.

4.4 White Paper

When important decision are made, such as software specifications or product developments, they should be documented in this section. Teams should explain how they came to the conclusion and key takeaways about the decision.

4.5 Technical Report

This section is where teams can highlight the processes of their work. This is a more in depth look into what was accomplised during the sprint where teams should describe step by step development.